HZ BOOKS
华章教育
PEARSON

计 算 机 科 学 丛 书

特别版

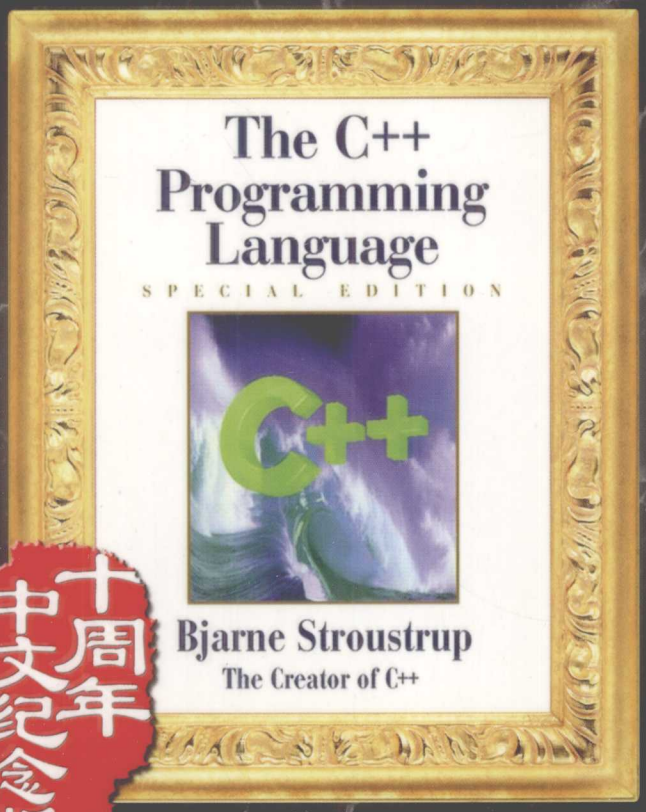
C++之父的经典之作

C++程序设计语言

(美) Bjarne Stroustrup 著 裘宗燕 译

1979年，Bjarne Stroustrup博士开始进行一项现在看来具有深远意义的工作—在C语言的基础上实现了内建支持面向对象程序设计方法的C with Classes，这就是震铄当代、让全世界数百万程序员如痴如狂的C++语言的前身。

1998年，ANSI/ISO C++标准建立，C++的标准化标志着Stroustrup博士倾注多年心血的伟大构想终于实现。



十周年中文纪念版

2000年，Stroustrup博士推出其经典著作The C++ Programming Language的特别版，这位C++之父将其对C++语言要义的理解、对编程精髓的把握、致C++程序员的箴言融会在这本书中，使本书自面世以来便成为C++编程领域最重要的著作，对全世界C++程序员产生了广泛而深刻的影响。

十年后，让我们重温经典，体味C++语言的精妙与魅力，享受与大师的心灵对话……

The C++ Programming Language

Special Edition

机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

特别版

C++程序设计语言

(美) Bjarne Stroustrup 著　裘宗燕 译

The C++ Programming Language

SPECIAL EDITION

十周年中文纪念版

The C++ Programming Language

Special Edition

机械工业出版社
China Machine Press

本书是在C++语言和程序设计领域具有深远影响、畅销不衰的著作，由C++语言的设计者编写，对C++语言进行了最全面、最权威的论述，覆盖标准C++以及由C++所支持的关键性编程技术和设计技术。本书英文原版一经面世，即引起业内人士的高度评价和热烈欢迎，先后被翻译成德、希、匈、西、荷、法、日、俄、中、韩等近20种语言，数以百万计的程序员从中获益，是无可取代的C++经典力作。

在本书英文原版面世10年后的今天，特别奉上十周年中文纪念版，希望众多具有丰富实战经验的C++开发人员能够温故而知新，印证学习心得，了解更加本质的C++知识，让获得的理论应用得更加灵活，也期望新的C++程序员从中认识到这本书的价值所在，从更高的起点出发，书写更加精彩的程序设计人生。

本书版权登记号：图字：01-2001-2195

图书在版编目（CIP）数据

C++程序设计语言（特别版）：十周年中文纪念版/（美）斯特朗斯特鲁普（Stroustrup, B.）著；裘宗燕译. —北京：机械工业出版社，2010.3
（计算机科学丛书）
书名原文：The C++ Programming Language, Special Edition

ISBN 978-7-111-29885-4

I. C… II. ①斯… ②裘… III. C语言—程序设计 IV. TP312

中国版本图书馆CIP数字核字（2010）第031474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：姚　蕾
北京瑞德印刷有限公司印刷
2010年3月第1版第1次印刷
184mm×260mm · 58.25印张
标准书号：ISBN 978-7-111-29885-4
定价：99.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991，88361066
购书热线：（010）68326294，88379649，68995259
投稿热线：（010）88379604
读者信箱：hzjsj@hzbook.com

出版者的话

写在《C++程序设计语言（特别版）》十周年中文纪念版付梓之前

在C++领域，《C++程序设计语言》是除了C++标准文献之外最权威的C++参考手册，作者Stroustrup博士是C++语言的设计师和最早的实现者，书中对C++程序设计语言的分析和描述非常完整和透彻，其他C++著作的内容莫不可以追溯至此。因此，本书自出版以来，先后被翻译成德、希、匈、西、荷、法、日、俄、中、韩等近20种语言，是拥有最多读者、使用最广泛的C++著作，在程序设计领域产生了深远的影响，数以百万计的程序员从中获益。

经典的著作是经得起岁月的沉淀和时间的考验的，这样的著作一经诞生，就已经独立于作者、独立于时代，而属于每个读者。本书就是这样一本经得起时间考验，值得一读再读、常读常新、用心体会、反复咀嚼的书。

为此，我们在本书特别版原版面世10年后的今天，特别奉上十周年中文纪念版，希望众多具有丰富实战经验的C++开发人员能够温故而知新，印证学习心得，了解更加本质的C++知识，让获得的理论应用得更加灵活，也期望新的C++程序员从中认识到这本书的价值所在，从更高的起点出发，书写更加精彩的程序设计人生。

“这是一个最好的时代，这是一个最坏的时代”。狄更斯的经典名言在当今繁荣与萧条并存、希望与失望共生、机遇与挑战互现的社会大环境下，依然掷地有声，发人深省。这个时代的机会之门向每一个人敞开，但获得机会的竞争也前所未有的激烈。与其临渊羡鱼，不如退而结网。读经典的著作，读大师的著作，在阅读中实现与大师思想的沟通和碰撞，分享大师的见地与感悟，惟有这样深入的思考才能积累创新的力量，获得超越前人的源动力和可持续发展的软实力。

把时间花在“经典”上，或许生命的份量将从此不同。

机械工业出版社华章公司

2010年3月

Preface to the Chinese Edition

This book is the most complete and up-to-date book on Standard C++. It is also the most widely read and most widely available. By my current best count, it is now available in 17 languages (see <http://www.research.att.com/~bs/covers.html>). This translation is based on a text that has been improved by thousands of suggestions from readers.

I am particularly pleased that this book is now more easily accessible to Chinese programmers and students. Chinese colleagues have pointed out the need for a translation to me, as has many Chinese programmers in email. Not being a native English speaker myself, I appreciate the need - and anyway, I just love my collection of translations as a graphic example of the widespread use of C++.

Naturally, some people prefer the original English text whereas others feel that a translation into their native language removes a barrier to understanding. I have known programmers who used both the original version and a translation to have the advantage of using their native language and also of having English available to communicate with programmers worldwide.

This book covers Standard C++, its standard library, and the fundamental programming techniques supported by C++, such as Object-Oriented Programming and Generic Programming. The aim is not just to explain the language facilities, but to provide sufficient information of their effective use for a programmer to cope with major projects. For that discussion of design is essential.

In August 1998 the ISO standard for C++ (ISO/IEC 14882 *Standard for the C++ Programming Language*) was ratified (by a 22-0 vote of national standards committees). This was a milestone for C++ inaugurating a new era of stability and advances in tools and techniques.

For me, the bottom line is that Standard C++ is a better approximation to my aims for C++ than any previous version. Standard C++ and its standard library allows me to write better, more elegant, and more efficient C++ programs than I have been able to in the past.

The aim of the standards effort was to specify a language and a library that will serve every part of the user community well without favoring any particular group of users, any particular company, or any particular country over others. It was an open and fair process aimed at quality and consensus.

One potential problem with an open and democratic standards process is "design by committee." This was largely avoided for C++. One reason was that I served as the chairman of the working group for language extensions. In that capacity, I evaluated all suggestions for major extensions and wrote the final version of the ones I, the working group, and the committee deemed both worthwhile and feasible. Thus, the primary activity of the committee was discussion of relatively complete designs presented to it, rather than design itself. Similarly, the major new part of the standard library - the "STL" providing a general, efficient, type-safe, and extensible framework of containers, iterators, and algorithms was primarily the work of one man: Alexander Stepanov.

Importantly, the C++ standard is not just a document. It is embodied in C++ implementations. All the major implementations now implement the standard with very minor exceptions. To help keep vendors honest, at least two companies now offer validation suites for Standard C++. Thus, the code I write now use — where appropriate — most of the facilities offered by Standard C++ and described in this edition of *The C++ Programming Language*.

The improvements to C++ language and the addition of a standard library have made a significant difference in the way I am able to write my code. My programs are now shorter, cleaner, and more efficient than they used to be. This is a direct result of Standard C++'s better, more systematic, and cleaner support for abstraction. Better support for facilities such as templates and exceptions reduce the need to deal with lower-level and messier facilities. Also, the last few years have seen the emergance of many new design and programming techniques, which are reflected in the presentation approach and examples of this book.

C++ can now be taught as a higher-level language. That is, the initial emphasis can be on algorithms and containers, rather than fiddling with bits, unions, C-style strings, arrays, etc. Naturally, lower-level concepts (such as arrays, non-trivial uses of pointers, and casts) must eventually be taught/learned. However, the presentation of such facilities can be postponed until a novice C++ programmer, reader, or student has gained the maturity to see them in the context of the higher-level concepts that they are used to implement.

In particular, I cannot overemphasize the importance of the statically type-safe strings and containers over programming styles involving lots of macros, casts, and arrays. In *The C++ Programming Language* I have been able to eliminate essentially all uses of macros and to reduce the use of casts to the handful of cases where they are essential. I consider the C/C++ form of macros a serious deficiency — which now has been made largely redundant by proper language facilities such as templates, namespaces, inline functions, and constants. Similarly, extensive use of casting is — in any language — a sign of weak design. Both macros and casts are major sources of errors. Being able to do without them makes C++ programming much safer and more elegant.

Standard C++ is meant to change the way we program in C++, to change the way we design our programs, and to change the way we teach C++ programming. Such changes do not happen overnight. I encourage you to take a long hard look at Standard C++, at the design and programming techniques used in *The C++ Programming Language*, and the way you program. I suggest that major improvements are possible. Do keep a cool head, though. There are no miracles, and using a language facility or a technique that you only partially understand in production code is dangerous. Now is the time to explore and experiment — the really major benefits of Standard C++ you reap only through the understanding of new concepts and new techniques.

Enjoy!

Bjarne Stroustrup

中文版序

本书是讲述标准C++的最完整和最新的著作，它拥有最多的读者，使用也最为广泛。按我目前的统计，本书已经被翻译成17种语言（参见http://www.research.att.com/~bs/covers.html）。所以，这个译本所依据的原文，已经从成千上万的读者建议中获益匪浅。

现在，中国的程序员和莘莘学子能够更容易地读到本书，对此我尤感欣慰。我的中国同事，还有许许多多中国的程序员（通过电子邮件）早就向我建议有必要将本书译为中文。因为自己的母语也不是英语，我当然也认识到了这种必要性——何况，我还非常喜欢拿本书译本的总数作为C++得到广泛应用的活生生的例子。

自然了，所谓“仁者乐山，智者乐水”，有人会更喜欢英文原版，而另一些人则会感觉阅读翻译成母语的版本更能消除理解上的障碍。我认识许多程序员同时使用原版和译本，这样既能发挥母语的优势，又能用英语与全世界的程序员进行交流。

本书涵盖了标准C++、它的标准库和C++所支持的基本技术，如面向对象程序设计和通用型程序设计。其目的不仅仅是阐述语言的功能，还要提供如何行之有效地使用这些功能的信息，使程序员足以应付大多数开发项目。因此其中对设计的讨论非常重要。

1998年，ISO的C++标准（ISO/IEC 14882 *Standard for the C++ Programming Language*）得到了批准（各国标准委员会以22-0全票通过）。这是C++发展史上的一个里程碑，开创了C++工具和技术稳定发展的新纪元。

对我本人而言，其中关键在于，标准C++相对于以前的任何版本，更接近于我对C++的目标。标准C++及其标准库使我能够编写出比过去更好、更优雅、更高效的C++程序。

标准化的目的是为一种语言和一个库制定规范，使其能够服务于所有用户群体，而不至偏向于某个用户群、某个公司或某个国家。这是一个以保证质量和达成共识为目的的开放、公正的过程。

开放和民主的标准化过程存在一个潜在的问题：所谓“由委员会设计”。这在C++的标准化中基本上被避免了。原因之一在于，我担任了语言扩展工作组的主席。在此位置上，我负责评估所有关于主要语言扩展方面的建议，并就那些我本人、工作组和委员会都认为值得和可行的建议撰写最终版本。因此，委员会的主要活动是讨论提交上来的相对完整的设计，而不是自己来设计。与此类似，标准库的主要新增部分——“STL”（为容器、迭代器和算法提供了通用的、高效的、类型安全的和可扩展的框架），主要都源自一个人——Alexander Stepanov的工作成果。

重要的是，C++标准不仅仅是一份文档。它已经在各种C++实现产品中得到了体现。所有主要的C++实现产品现在都实现了标准，只有极少的几个例外。为了帮助厂商更好地实现标准，现在至少有两个公司提供了标准C++的验证套件。因此，我现在写代码，只要合适，都会用到标准C++提供的和本书这一版中讲述的功能。

C++语言的改进和标准库的增加，使我自己编写代码的方式发生了显著变化。现在我的程序比原来更加简洁、更加高效。这直接得益于标准C++对抽象更好、更系统和更纯粹的支持。

对模板和异常等功能更好的支持，使对底层处理和更混乱的功能的需要大大降低了。而且，最近几年出现了许多新的设计和编程技术，这在本书的表达方法和实例中都有所反映。

C++现在可以作为高级语言来讲授了。也就是说，重点一开始就可以放在算法和容器上，而不用再在什么位呀，联合呀，C风格字符串，数组等等东西上纠缠不清了。自然，底层的概念（如数组、重要的指针应用和强制转换）最终还是要教要学的。但是，可以等到作为新手的C++程序员、读者或学生已经成熟，能够在实现这些功能的高级概念的大背景中看待它们的时候，再对这些功能进行阐释。

我想特别强调（怎么强调都不过分）的是，应该多使用静态类型安全的字符串和容器，而不要学那些使用大量宏、强制转换和数组的编程风格。在本书中，我能够根本就不用宏，并且只在很少的非用不可的情况下才使用强制转换。我认为C/C++形式的宏是一种严重的缺陷——现在因为有了模板、名字空间、在线函数和常量这些正确的语言功能，它很大程度上更是一种多余了。同样，在任何语言中，强制转换的大量使用都是设计不良的标志。宏和强制转换是错误的主要渊薮。不用它们也能工作，这一点大大提高了C++编程的安全性和优雅性。

标准C++改变了我们使用C++编程、设计程序以及教授C++编程的方式。这些变化不可能“毕其功于一役”。我鼓励你在标准C++、在本书中所用的设计和编程技术，以及自己的编程方式上好好下一番功夫。我想脱胎换骨是有可能的。但是别太死心眼了。奇迹是不存在的，在产品代码中使用仅仅一知半解的语言功能和技术是相当危险的。现在该开始探索，开始试验了——标准C++真正对你有所裨益的地方，就在理解新概念和新技术的旅程中！

旅途愉快！

Bjarne Stroustrup

译 者 序

Bjarne Stroustrup的《*The C++ Programming Language*》是有关C++语言的第一部著作。毫无疑问，它是关于C++语言及其程序设计的最重要著作，在此领域中的地位是无可替代的。《*The C++ Programming Language*》一书伴随着C++语言的发展演化而不断进步，经过第1版（1985年）、第2版（1991年），第3版（1998年），本书的英文原书是《*The C++ Programming Language*》第3版经过补充和修订后的“特别版（2000）”（对应于国内引进的影印本）。对于这个中译本，我想说的第一句话就是“来得太晚了”。

要学习C++语言和程序设计，要将C++应用于程序设计实践，本书自然是必读之书。这个“特别版”以标准化的C++语言为基础，讨论了C++的各种语言特征和有效使用这一语言的程序设计技术。书中也用了大量的篇幅，在标准库以及一般软件开发的环境下，讨论了使用C++语言编程和组织程序的许多高级技术。本书内容覆盖了C++语言及其程序设计的各个方面，其技术深度与广度是举世公认的。

然而，作者讨论的并不仅是C++语言及其程序设计。本书的讨论远远超出这一范围，第四部分用了大量的篇幅去讨论软件开发过程及其问题。即使是在介绍C++语言及其程序设计的具体问题时，作者也常在程序结构、设计和软件开发的大环境下，提出自己的许多认识。作者有很强的计算机科学与技术基础，在系统软件开发方面极富经验，他所提出的观点和意见值得每个在这个领域中工作的人的重视。

当然，重视并不是盲从。在Stroustrup的两本关于C++的重要著作（本书和《C++语言的设计与演化》（已由机械工业出版社出版））中，都有这样一句话使我印象深刻：希望读者带着一种健康的怀疑态度。看来这是作者深深铭刻在心的一种思想，也特别值得国内每个从事信息技术，或者努力向这个方向发展的人注意。从来就没有什么救世主，Stroustrup不是在传道，他只是在总结和论述自己在这个领域中工作的经验。请不要将本书中的东西作为教条，那也一定是本书作者所深恶痛绝的。

许多人说本书比较难读，这种说法有一定道理。真正理解本书的一般性内容需要花一些时间，融会贯通则更需要下功夫。理解本书的内容不仅需要去读它，还需要去实践。问题是，花这个时间值吗？作者在讨论C++语言的设计时提出了一个观点：你从C++语言中的收获大致与在学习实践这个语言的过程中所付出的努力成正比；而且C++是一个可以伴随你成长的语言。同样的话也适用于本书。如果你致力于将自己发展成一个职业的程序员，或者要在计算机方面的技术领域中长期工作下去，我认为，你从本书中的收获大致也会与你所花的时间成正比，这本书也是一本能够伴随你成长的书。

当然，这本书也不是没有缺陷的。由于作者有着极其丰富的实践经验，因此，当他想要论述一个问题、提出一个观点时，常会想到在自己长期实践中最适合说明这个问题的示例，用几句简短的话引述有关的情况。由于作者对C++谙练有加，因此，在讨论中有时会不知不觉地将某些并不显然的东西当做不言自明的事情提出来。而对于许多初学者而言，这些都可能成为学习中的障碍。为了帮助这部分读者，我也在书中一些地方加入了少量注释，解释一些背景性情况。过多过滥的注释会增大书的篇幅，干扰读者阅读，绝不会是大家都喜欢的方

式。因此我在这样做的时候也很有节制，希望不会引起读者的反感。

由于读者的水平有极大差异，对一些人很熟的东西，对另一些人可能就会莫名其妙。我无法解决所有问题，但也希望能为广大读者做一点服务性的工作。为了帮助初学者入门，为使本书（包括其中文译本）能更好地在国内计算机领域中发挥作用，也为了关心本书、学习本书的人们能够有一个交流经验、传播认识的场所，我将在下面地址维护一个有关本书的信息、情况、认识和意见的网页：

http://www.math.pku.edu.cn/teachers/qiuzy/cpp.htm

在其中收集有关的信息，记录朋友们（包括我自己）的认识与意见，提出的问题和相应的认识，有关这一译本的勘误信息，原英文书的更正与更新信息等。欢迎读者提供自己的见解和问题，提供有价值的线索。我没时间去创建与维护一个“纯的”C++语言及其程序设计的讨论组（确实需要这样的场所，而有关VC等的讨论组倒是太多了。如果有人愿做，我乐得坐享其成、积极参与并尽可能提供帮助），只想抽空将接收到的和自己写的东西编辑公布。与我联系可以发email：qzy@math.pku.edu.cn。我还将把有关《C++语言的设计与演化》一书的相关信息也放在那里，供大家参考。

还请读者注意，本书的英文原版书是“特别版”的第1次印刷，即“第3版”的第11次印刷，也是目前国内可买到的影印本的原书。在那以后，作者在重印时不断更正书中的错误，并修改了少量的程序示例。最新的重印是第16次印刷，有关情况可从作者的网页或上面网址找到。由于一些情况，本书无法按最新的重印本翻译，但我还是参考了作者的网页，在译文中尽可能地采纳了有关勘误信息。此外，在翻译过程中我也发现了一些错误。经与作者通过电子邮件讨论取得了一致意见，有关更正反映在本书里。由于这些原因，本书在个别地方的说法可能与读者手头的英文原书有异。如果想确认有关情况，请查看原书的勘误信息。

裘宗燕

2002年2月于北京大学数学学院信息科学系

译者简介

裘宗燕，北京大学数学学院信息科学系教授。长期从事计算机软件与理论、程序设计语言和符号计算方面的研究和教学工作。已出版多部著作和译著，包括：《程序设计语言基础》（译著，北京大学出版社，1990），《Mathematica数学软件系统的应用与程序设计》（编著，北京大学出版社，1994），《计算概论（上）》（合著，高等教育出版社，1997），《从问题到程序——程序设计与C语言引论》（编著，北京大学出版社，1999），《程序设计实践》（译著，机械工业出版社，2000），《C++语言的设计和演化》（译著，机械工业出版社，2002），《程序设计语言——概念和结构》（合译，机械工业出版社，2002）等。

序

去编程就是去理解。
——Kristen Nyggard

我觉得用C++ 编程序比以往更令人感到愉快。在过去这些年里，C++ 在支持设计和编程方面取得了令人振奋的进步，针对其使用的大量新技术已经被开发出来了。然而，C++ 并不就是好玩。普通的实际程序员在几乎所有种类和规模的开发项目上，在生产率、可维护性、灵活性和质量方面都取得了显著的进步。到今天为止，C++ 已经实现了我当初对它的期望中的绝大部分，还在许多我原来根本没有梦想过的工作中取得了成功。

本书介绍的是标准C++[⊖] 以及由C++ 所支持的关键性编程技术和设计技术。与本书第1版所介绍的那个C++ 版本相比，标准C++ 是一个经过了更仔细推敲的更强大的语言。各种新的语言特征，如名字空间、异常、模板，以及运行时类型识别，使人能以比过去更直接的方式使用许多技术，标准库使程序员能够从比基本语言高得多的层面上起步。

本书第2版中大约有三分之一的内容来自第1版。这个第3版则是重写了比例更大的篇幅的结果。它提供的许多东西是大部分有经验的程序员也需要的，与此同时，本书也比它的以前版本更容易供新手入门。C++ 使用的爆炸性增长和作为其结果的海量经验积累使这些成为可能。

一个功能广泛的标准库定义使我能以一种与以前不同的方式介绍C++ 的各种概念。与过去一样，本书对C++ 的介绍与任何特定的实现都没有关系；与过去一样，教材式的各章还是采用“自下而上”的方式，使每种结构都是在定义之后才使用。无论如何，使用一个设计良好的库远比理解其实现细节容易得多。由于这些情况，在假定读者已经理解了标准库的内部工作原理之前，就可以利用它提供许多更实际更有趣的例子。标准库本身也是程序设计实例和设计技术的丰富源泉。

本书将介绍每种主要的C++ 语言特征和这个标准库，它是围绕着语言和库功能组织起来的。当然，各种特征都将在使用它们的环境中介绍。也就是说，这里所关注的是将语言作为一种设计和编程的工具，而不是语言本身。本书将展示那些使C++ 卓有成效的关键性技术，讲述为掌握它们所需要的那些基本概念。除了专门阐释技术细节的那些地方之外，其他示例都取自系统软件领域。另一本与本书配套出版的书《带标注的C++ 语言标准》（*The Annotated C++ Language Standard*），将给出完整的语言定义，所附标注能使它更容易理解。

本书的基本目标就是帮助读者理解C++ 所提供的功能将如何支持关键性的程序设计技术。这里的目标是使读者能远远超越简单地复制示例并使之能够运行，或者模仿来自其他语言的程序设计风格。只有对隐藏在语言背后的思想有了一个很好的理解之后，才能真正掌握这个语言。如果有一些具体实现的文档的辅助，这里所提供的信息就足以对付具有挑战性的真实世界中的重要项目。我的希望是，本书能帮助读者获得新的洞察力，使他们成为更好的程序员和设计师。

⊖ ISO/IEC 14882, C++程序设计语言标准。

致谢

除了第1版和第2版的致谢中所提到的那些人之外，我还要感谢Matt Austern，Hans Boehm，Don Caldwell，Lawrence Crowl，Alan Feuer，Andrew Forrest，David Gay，Tim Griffin，Peter Juhl，Brian Kernighan，Andrew Koenig，Mike Mowbray，Rob Murray，Lee Nackman，Joseph Newcomer，Alex Stepanov，David Vandevoorde，Peter Weinberger和Chris Van Wyk，他们对第3版各章的初稿提出了许多评论和意见。没有他们的帮助和建议，这本书一定会更难理解，包含更多的错误，没有这么完全，当然也可能稍微短一点。

我还要感谢C++ 标准化委员会的志愿者们，是他们完成了规模宏大的建设性工作，才使C++ 具有它今天这个样子。要罗列出每个人就会有一点不公平，但一个也不提就更不公平，所以我想特别提出Mike Ball，Dag Brück，Sean Corfield，Ted Goldstein，Kim Knuttila，Andrew Koenig，José Lajoie，Dmitry Lenkov，Nathan Myers，Martin O'Riordan，Tom Plum，Jonathan Shopiro，John Spicer，Jerry Schwarz，Alex Stepanov和Mike Vilot，他们中的每个都在C++及其标准库的某些方面直接与我合作过。

在这本书第一次印刷之后，许多人给我发来电子邮件，提出更正和建议。我已经在原书的结构里响应了他们的建议，使后来出版的版本大为改善。将本书翻译到各种语言的译者也提供了许多澄清性的意见。作为对这些读者的回应，我增加了附录D和附录E。让我借这个机会感谢他们之中特别有帮助的几位：Dave Abrahams，Matt Austern，Jan Bielawski，Janina Mincer Daszkiewicz，Andrew Koenig，Dietmar Kühl，Nicolai Josuttis，Nathan Myers，Paul E. Sevinc，Andy Tenne-Sens，Shoichi Uchida，Ping-Fai(Mike) Yang和Dennis Yelle。

Bjarne Stroustrup
Murray Hill，新泽西

第2版序

前路漫漫。
——Bilbo Baggins

正如在本书的第1版所承诺的，C++ 为满足其用户的需要正在不断地演化。这一演化过程得助于许多有着极大的背景差异，在范围广泛的应用领域中工作的用户们的实际经验的指导。在第1版出版后的六年中，C++ 的用户群体扩大了不只百倍，人们学到了许多东西，发现了许多新技术并通过了实践的检验。这些技术中的一些也在这一版中有所反映。

在过去六年里所完成的许多语言扩展，其基本宗旨就是将C++ 提升成为一种服务于一般性的数据抽象和面向对象程序设计的语言，特别是提升为一个可编写高质量的用户定义类型库的工具。一个“高质量的库”是指这样的库，它以一个或几个方便、安全且高效的类的形式，给用户提供了一个概念。在这个环境中，安全意味着这个类在库的使用者与它的供方之间构成了一个特殊的类型安全的界面；高效意味着与手工写出的C代码相比，这种库的使用不会给用户强加明显的运行时间上或空间上的额外开销。

本书介绍的是完整的C++ 语言。从第1章到第10章是一个教材式的导引，第11章到第13章展现的是一个有关设计和软件开发问题的讨论，最后包含了完整的C++ 参考手册。自然，在原来版本之后新加入的特征和变化已成为这个展示的有机组成部分。这些特征包括：经过精化后的重载解析规则和存储管理功能，以及访问控制机制、类型安全的连接、***const***和***static***成员函数、抽象类、多重继承、模板和异常处理。

C++ 是一个通用的程序设计语言，其核心应用领域是最广泛意义上的系统程序设计。此外，C++ 还被成功地用到许多无法称为系统程序设计的应用领域中。从最摩登的小型计算机到最大的超级计算机上，以及几乎所有操作系统上都有C++ 的实现。因此，本书描述的是C++ 语言本身，并不想试着去解释任何特殊的实现、程序设计环境或者库。

本书中给出的许多类的示例虽然都很有用，但也还是应该归到“玩具”一类。与在完整的精益求精的程序中做解释相比，这里所采用的解说风格能更清晰地呈现那些具有普遍意义的原理和极其有用的技术，在实际例子中它们很容易被细节所淹没。这里给出的大部分有用的类，如链接表、数组、字符串、矩阵、图形类、关联数组等，在广泛可用的各种商品的和非商品资源中，都有可用的“防弹”和/或“金盘”版本。那些“具有工业强度”的类和库中的许多东西，实际上不过是在这里可以找到的玩具版本的直接或间接后裔。

与本书的第1版相比，这一版更加强调本书的教学方面的作用。然而，这里的叙述仍然是针对有经验的程序员，并努力不去轻视他们的智慧和经验。有关设计问题的讨论有了很大的扩充，作为对读者在语言特征及其直接应用之外的要求的一种回应。技术细节和精确性也有所增强。特别是，这里的参考手册表现了在这个方向上多年的工作。我的目标是提供一本具有足够深度的书籍，使大部分程序员能在多次阅读中都有所收获。换句话说，这本书给出的是C++ 语言，它的基本原理，以及使用时所需要的关键性技术。欢迎欣赏！

致谢

除了在第1版序中致谢里所提到人们之外，我还要感谢Al Aho，Steve Buroff，Jim Coplien，Ted Goldstein，Tony Hansen，Lorraine Juhl，Peter Juhl，Brian Kernighan，Andrew Koenig，Bill Leggett，Warren Montgomery，Mike Mowbray，Rob Murray，Jonathan Shopiro，Mike Vilot和Peter Weinberger，他们对第2版的初稿提出了许多意见。许多人对C++从1985年到1991年的开发有很大影响，我只能提出其中几个：Andrew Koenig，Brian Kernighan，Doug McIlroy和Johathan Shopiro。还要感谢参考手册“外部评阅”的许多参与者，以及在X3J16的整个第一年里一直在其中受苦的人们。

Bjarne Stroustrup
Murray Hill，新泽西

第1版序

语言磨砺了我们思维的方式，
也决定着我们思考的范围。
——B.L. Whorf

C++ 是一种通用的程序设计语言，其设计就是为了使认真的程序员工作得更愉快。除了一些小细节之外，C++ 是C程序设计语言的一个超集。C++ 提供了C所提供的各种功能，还为定义新类型提供了灵活而有效的功能。程序员可以通过定义新类型，使这些类型与应用中的概念紧密对应，从而把一个应用划分成许多容易管理的片段。这种程序构造技术通常被称为*数据抽象*。某些用户定义类型的对象包含着类型信息，这种对象就可以方便而安全地用在那种对象类型无法在编译时确定的环境中。使用这种类型的对象的程序通常被称为是*基于对象*的。如果用得好，这些技术可以产生出更短、更容易理解，而且也更容易管理的程序。

C++ 里的最关键概念是*类*。一个类就是一个用户定义类型。类提供了对数据的隐藏，数据的初始化保证，用户定义类型的隐式类型转换，动态类型识别，用户控制的存储管理，以及重载运算符的机制等。在类型检查和表述模块性方面，C++提供了比C好得多的功能。它还包含了许多并不直接与类相关的改进，包括符号常量、函数的在线[⊖]替换、默认函数参数、重载函数名、自由存储管理运算符，以及引用类型等。C++ 保持了C高效处理硬件基本对象（位、字节、字、地址等）的能力。这就使用户定义类型能够在相当高的效率水平上实现。

C++ 及其标准库也是为了可移植性而设计的。当前的实现能够在大多数支持C的系统上运行。C的库也能用于C++ 程序，而且大部分支持C程序设计的工具也同样能用于C++。

本书的基本目标就是帮助认真的程序员学习这个语言，并将它用于那些非平凡的项目。书中提供了有关C++ 的完整描述，许多完整的例子，以及更多的程序片段。

致谢

如果没有许多朋友和同事持之以恒的使用、建议和建设性的批评，C++ 绝不会像它现在这样成熟。特别地，Tom Cargill，Jim Coplien，Stu Feldman，Sandy Fraser，Steve Johnson，Brian Kernighan，Bart Locanthi，Doug McIlroy，Dennis Rechie，Larry Rosler，Jerry Schwarz和Jon Shopiro对语言发展提供了重要的思想。Dave Presotto写出了流I/O库的当前实现。

此外，还有成百的人们对C++ 及其编译器的开发做出了贡献：他们给我提出改进的建议，描述他们所遇到的问题，告诉我编译中的错误等。我只能提出其中的很少几位：Gary Bishop，Abdrew Hume，Tom Karzes，Victor Milenkovic，Rob Murray，Leonie Rose，Brian Schmult和Gary Walker。

⊖ inline在本书中统一地直译为“在线”，表示函数代码在调用位置的在线展开（而不是编译为独立的函数代码段）和运行中的在线执行（不经过函数调用）。目前常见的另一译法是“内联”，因其不妥，本书没有采用。——译者注

许多人在本书的撰写过程中为我提供了帮助，特别值得提出的是Jon Bentley，Laura Eaves，Brian Kernighan，Ted Kowalski，Steve Mahaney，Jon Shopiro，以及参加1985年7月26～27日俄亥俄州哥伦布贝尔实验室C++ 课程的人们。

Bjarne Stroustrup
Murray Hill, 新泽西

目　录