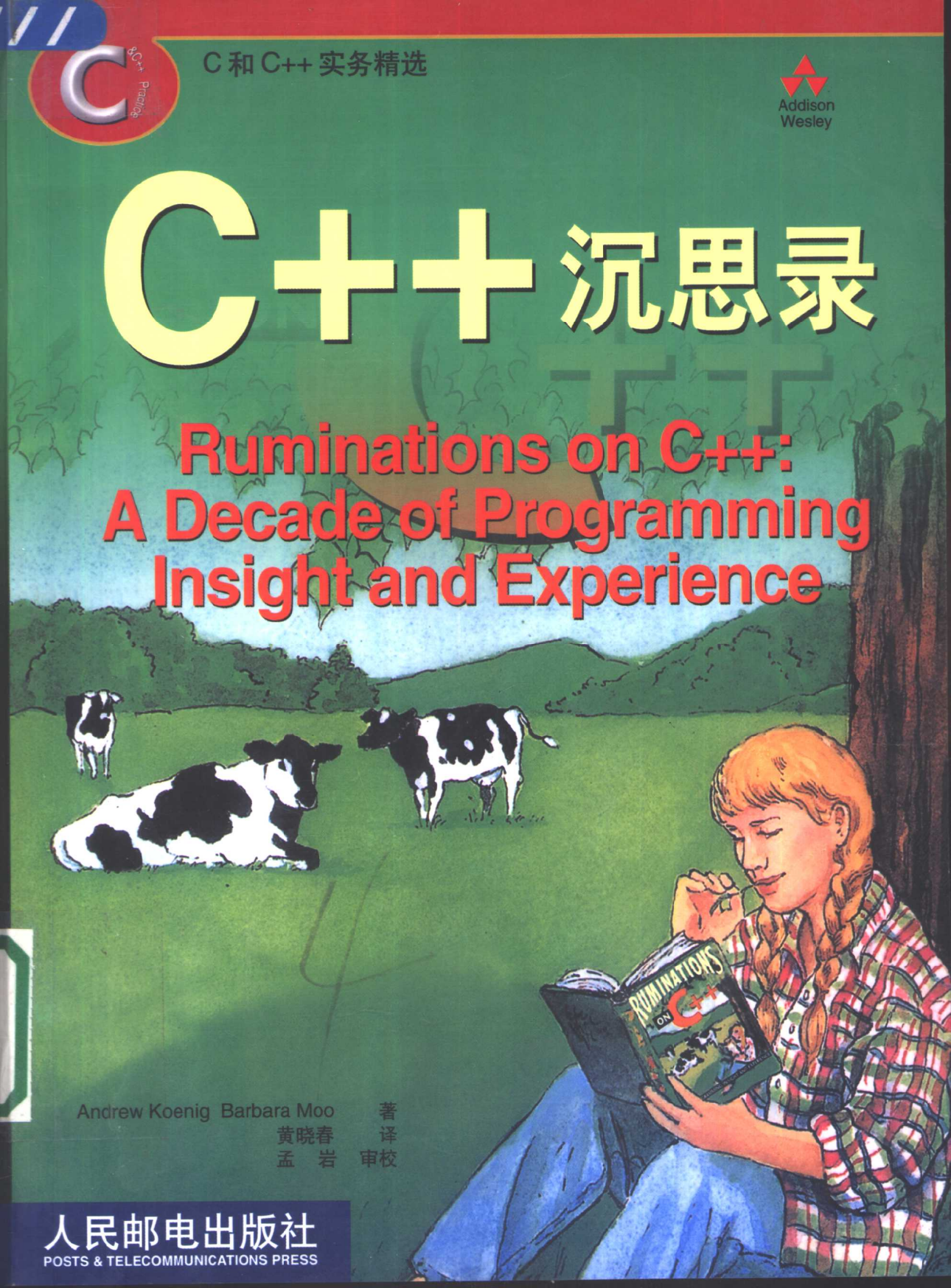
C 和 C++ 实务精选

Addison Wesley

C++沉思录

Ruminations on C++: A Decade of Programming Insight and Experience



Andrew Koenig Barbara Moo 著
黄晓春 译
孟 岩 审校

人民邮电出版社
POSTS & TELECOMMUNICATIONS PRESS

C和C++实务精选

C++ 沉思录

Andrew Koenig　Barbara Moo 著
黄晓春　译
孟　岩　审校

人民邮电出版社

C 和 C++实务精选

C++ 沉思录

◆ 著　　Andrew Koenig　Barbara Moo
译　　黄晓春
审　　校　孟　岩
责任编辑　陈冀康

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-67132705
北京汉魂图文设计有限公司制作
北京朝阳展望印刷厂印刷
新华书店总店北京发行所经销

◆ 开本：720×980　1/16
印张：26.25
字数：504 千字　　2002 年 11 月第 1 版
印数：1-5 000 册　　2002 年 11 月北京第 1 次印刷

著作权合同登记　图字：01－2002－1080 号
ISBN 7-115-10622-3/TP・3079

定价：50.00 元

内容提要

本书基于作者在知名技术杂志发表的技术文章、世界各地发表的演讲以及斯坦福大学的课程讲义整理、写作而成，融聚了作者 10 多年 C++程序生涯的真知灼见。

全书分为 6 大部分，共 32 章，分别对 C++语言的历史和特点、类和继承、STL 与泛型编程、库的设计等几大技术话题进行了详细而深入的讨论，细微之处几乎涵盖了 C++所有的设计思想和技术细节。全书通过精心挑选的实例，向读者传达先进的程序设计的方法和理念。

本书适合有一定经验的C++程序员阅读学习，可以帮助你加强提高技术能力，成为 C++程序设计的高手。

作者简介

Andrew Koenig

AT&T 大规模程序研发部（前贝尔实验室）成员。他从 1986 年开始从事 C 语言的研究，1977 年加入贝尔实验室。他编写了一些早期的类库，并在 1988 年组织召开了第一个具有相当规模的 C++会议。在 ISO/ANSI C++委员会成立的 1989 年，他就加入了该委员会，并一直担任项目编辑，他已经发表了 C++方面的 100 多篇论文，在 Addsion-Wesley 出版了 *C Traps and Pitfalls* 一书（中文版名为《C缺陷与陷阱》，由人民邮电出版社出版），还应邀到世界各地演讲。

Barbara Moo

现任 AT&T 网络体系结构部门负责人。在 1983 年加入贝尔实验室不久，她开始从事 Fortran77 编译器的研究工作，这是第一个用 C++编写的商业产品。她负责 AT&T 的 C++编译器项目直到 AT&T 卖掉它的软件业务。她还为 SIGS 会议、Lund 技术学院和 Stanford 大学提供辅导课程。

Anderw Koenig 和 Barbara Moo 不仅有着多年的 C++开发、研究和教学经验，而且还亲身参与了 C++的演化和变革，对 C++的变化和发展起到重要的影响。

Preface to the Chinese Edition

This is a book about programming in C++. In particular, it is first a book about programming and then a book about C++. In that sense, it is very different from most books about C++, which concentrate on what the language is rather than on how to use it.

Is the best author the one who knows the most words? Is the best musician the one who can play the most notes? Is the best painting the one with the most brush strokes? Of course not--these very ideas are absurd. Yet too often, we think that the best programmers are the ones who know the most language features. This idea is equally absurd: The hard part of programming is not learning the details of language features——it is understanding how to solve problems.

This book doesn't talk much about language features. Instead, it talks about programming techniques. Just as a writer must learn how to tell a story, a programmer must learn how to analyze a problem. Accordingly, this book is full of problems and their solutions. Studying them is one way to become a better programmer.

Most of the solutions in this book share the idea of abstraction——concentrating one's attention on just the parts of a problem that are important in the current context. It is impossible to write computer programs without using abstraction in one way or another, a fact that makes abstraction the most important single idea in programming. C++ supports abstraction in several forms, the best known of which are abstract data types, object-oriented programming, and generic programming.

Generic programming was not a widely known idea when we published this book. The idea became popular only a few years later, when the STL (Standard Template Library), a library designed to support generic programming, became part of the C++ standard library. As a result, the ideas in this book have become more important with time.

We hope that you can use these ideas to understand the difference between a pile of code and an abstraction——a difference that is as important as the difference between a pile of words and a story, or a pile of notes and a song, or a pile of brush strokes and a painting.

Andrew Koenig
Barbara Moo
Gillette, New Jersey, USA
October, 2002

中文版序

这是一本关于C++程序设计的书。说得具体些，它首先是一本关于程序设计的书，其次才是一本关于C++的书。从这个意义上讲，这本书与坊间大部分C++书籍都不一样，那些书所关注的是语言本身，而不是如何运用这种语言。

识字最多的人一定是最好的作家吗？能演奏最多音符的人一定是最好的音乐家吗？最勤于挥舞画笔的人一定是最好的画家吗？显然不是——这些观点极其荒谬。然而，我们却经常认为，那些了解最多语言特性的人就是最好的程序员。这一看法同样是荒谬的：编程工作中最困难的部分并不是去学习语言细节，而是理解问题的解决之道。

这本书对于语言本身并没有说太多。相反，我们谈了很多关于程序设计技术方面的话题。一个作家必须学习如何讲述故事，同样，一个程序员也必须学习如何分析问题。这本书中包含了大量的问题，以及针对这些问题的解决方案。认真地研习这些内容*，将会有助于你成为更出色的程序员。

这本书中所展示的解决方案有一个共同的思想，那就是抽象——集中注意力，只关注问题中那些在当前背景下最为重要的部分。可以说，如果不以某种方式进行抽象，你就不可能编写任何计算机程序，只此一点已经足以使“抽象”成为程序设计中最重要的单个思想。C++支持好几种不同的抽象形式，其中最著名的有抽象数据结构（Abstract Data Type，ADT）、面向对象程序设计和泛型程序设计。

我们出版这本书的时候，泛型程序设计还没有得到广泛的认知。短短几年后，STL（Standard Template Library，标准模板库）成为了C++标准库的一部分，这一思想也已经非常流行。所有这些使得本书中的思想随着时间的推移而越来越重要。

---

* 译者注：这里“研习”一词，原文是 study。这个词在英文里意义是比较重的，ACCU 主席 Francis Glassborow 曾经说过，所谓 study，就是“阅读，学习，再阅读，再学习，反复阅读和学习，直到彻底理解”。

我们希望您能运用这些思想去理解一堆拼凑的代码与一个抽象之间的差别——这种差别，就好像一堆词藻与一篇文章，一堆音符与一支歌曲，一纸涂鸦与一幅图画之间的差别一样。

Andrew Koenig

Barbara Moo

2002 年 10 月

于美国新泽西州吉列

大师的沉思

——读C++经典著作 *Ruminations on C++*有感

人民邮电出版社即将推出 C++编程领域的又一部经典著作 *Ruminations on C++*中文版——《C++沉思录》。作为一个普通的C++程序员，我很荣幸能有机会成为本书中文版的第一个读者，先饱眼福。原书英文版我也有，虽然也不时拿出来翻看，但是随意的摘读与通篇的浏览不同，通篇的浏览与技术审校又不同，此番对照中英文，从头到尾把此书读过一遍，想过一遍，确实是收获丰厚，感慨良多。

本书作者，不需要我过多的介绍。虽然我们不赞成论资排辈的习气，但是所谓公道自在人心，Andy Koenig 在C++发展历史中不可置疑的权威地位，是勿庸置疑的。作为 Bjarne Stroustrup 的亲密朋友，ANSI C++标准委员会的项目编辑，Koenig 在C++的整个发展过程中都发挥了及其重要的作用，是C++社群中最受尊敬的大师之一。特别值得一提的是，在C++大师中，Koenig 的教学实践和文字能力历来备受好评，在前后十几年的时间里，他在各大技术刊物上发表了近百篇C++技术文章。这些文章长时间以来以其朴实而又精深的思想，准确而又权威的论述，高屋建瓴而又平易近人的表达方式，成为业界公认的“正统 C++之声”。本书第二作者 Barbara Moo,是 Koenig 的夫人，也是他在贝尔实验室的同事，曾经领导 AT&T 的 Fortran77 和 CFont 编译器项目，可谓计算机科学领域中的巾帼英雄。这本书正是在 Barbara Moo 的建议下，由二人共同从 Koenig 所发表的文章中精选、编修、升华而成的一本结集之作。由于源自杂志的专栏文章，因此书中的内容具有高度的可读性，知识密度高，表现力强。更重要的是，这些文章是在发表之后若干年，由原作者挑选出来，经过了多年的沉淀和反思，重新编辑整理，加上自己多年的心得与思考，自然有一种千锤百炼的韧性和纯度。也正因为如此，作者当仁不让地把这本书命名为 *Ruminations on C++*，rumination 一词，充分显现出作者的自信和对这本书的珍爱。

这两位C++发展史上的重要人物夫唱妇随，一同出版著作，本身就足以引起整个C++社群的高度重视，而这本书不平凡的来历和出版之后5年间所获得的极高赞誉，更加确立了它在C++技术书籍中的经典地位。Bjarne Stroustrup在他的主页上特别推荐人们去阅读这本书，ACCU的主席Francis Glassborow在书评中慷慨地向读者最热诚地推荐此书，说“我对这本书没什么更多可说的，因为每个C++程序员都应该去读这本书。如果你在阅读的过程中既没有感到快乐，又没学到什么东西，那你可真是罕见的人物”。而著名C++专家Chuck Allison在他自己的书*C & C++ Code Capsules*（本书中文版《C和C++代码精粹》将由人民邮电出版社出版，编者注）中，更是直截了当地说：“对我来说，这是我所有C++藏书中最好的一本。”

对我来说，给这本书一个合适的评价超出了我现在的能力。究竟它能够为我的学习和工作带来怎样的启发，还需要更长时间的实践来验证。不过就目前而言，这本书的一些特色已经给我留下很深刻的印象。

首先，作者对C++有着居高临下的见识，对于C++的设计理念和实际应用有非常清晰的观点。众多纷繁复杂的C++特性如何组合运用，如何有效运作，什么是主流，什么是旁支，哪些是通用技术，哪些是特殊的技巧，在书中都有清晰明白的介绍。我们都知道，C++有自己的一套思想体系，它虽然有庞大的体积，繁多的特性，无穷无尽的技术组合能力，但是其核心理念也是很朴实、很简单的。掌握了C++的核心理念，在实践中就会“有主心骨”，有自己的技术判断力。但是在很多C++书籍，甚至某些经典名著中，C++的核心理念被纷繁的技术细节所遮掩，变得模糊不清，读者很容易偏重于技术细节，最后陷入其中，不能自拔。而在这本书中，作者毫不含糊地把C++的核心观念展现在读者面前，为读者引导方向。全书中多次强调，C++最基本的设计理念就是“用类来表示概念”，C++解决复杂性的基本原则是抽象，面向对象思想是C++的手段之一，而不是全部，等等。这些言论可以说是掷地有声，对我们很多程序员来说都是一剂纠偏良药。

其次，这本书在C++的教学方式上有独到之妙。作者循循善诱，娓娓道来，所举的例子虽然小，但是非常典型，切中要害，让你花费不大的精力就可以掌握相当多的东西。比如本书讲述面向对象编程时先后只讲了几项技术，举了两个例子，但是细细读来，你会对C++面向对象编程有一个基本的正确观念，知道应该用具体类型来隐藏一个派生层次，知道应该如何处理动态内存管理的问题。从这

一点点内容中能够得到的收获，比看一大堆厚书都来得清晰深刻。对于 STL 的介绍，更是独具匠心。作者不是一上来就讲 STL，而是把 STL 之前的那些类似技术一一道来，把优点缺点讲清楚，然后从道理上给你讲清楚 STL 的设计和运用，让你不仅知其然而且知其所以然，胸有成竹。

书毕竟不厚，我想更重要的东西并不是这本书教给了你什么技术。所谓授人以鱼不如授人以渔。这本书最大的特点就在于，不仅仅告诉你什么是答案，更重要的是告诉你思考的方法，解决问题的步骤和方向。书中遍布了大量宝贵的建议，正是这些建议，为这本书增添了永不磨灭的价值。Francis Glassborow 甚至说，仅仅这本书的第 32 章给出的建议，就足以体现全书的价值。

当前，C++面临其发展历史中的一个非常重要的时期。一方面，它受到了不公正的质疑和诋毁，个别新兴语言的狂热拥护者甚至迫不及待地想宣布 C++的死讯。而另一方面，C++在学术界和工业界都在稳定地发展，符合 ISO 标准的 C++编译器呼之欲出，人们对于 C++特性的合理运用的认识也越来越丰富，越来越成熟和全面。事实上，根据我个人从业界了解到的情形，以及从近期 C++的出版物的内容和质量上看，C++经过这么多年的积淀，已经开始真正的成熟发展时期，它的步子越来越稳健，思路越来越清晰，越来越演化成为一种强大而又实用的编程语言。作为工业界的基础技术，C++还将在很长的一段时间里扮演不可替代的重要角色。因此，这本书也会在很长的时间里伴随我们的学习与实践，并且引导我们以正确的观点看待技术的发展，帮助我们中国程序员形成属于我们自己的、成熟的、独立的技术判断力。

孟岩

2002 年 10 月

前言

原由

1988年初，大概是我刚刚写完 *C Traps and Pitfalls*（本书中文版《陷阱与缺陷》由人民邮电出版社出版）的时候，Bjarne Stroustrup 跑来告诉我，他刚刚被邀请参加了一个新杂志的编委会，那个杂志叫做《面向对象编程月刊》(*Journal of Object-Oriented Programming，JOOP*)。该杂志试图在那些面孔冰冷的学术期刊与满是产品介绍和广告的庸俗杂志之间寻求一个折中。他们在找一个C++专栏作家，问我是否感兴趣。

那时，C++对于编程界的重要影响才刚刚开始。Usenix 其时才刚刚在新墨西哥圣达菲举办了第一届 C++交流会。他们预期有 50 人参加，结果到场的有 200 人。更多的人希望搭上 C++快车，这意味着 C++社群急需一个准确而理智的声音，去对抗必然汹涌而至的谣言大潮。需要有个人能够在谣言和实质之间明辨是非，在任何混乱之中保持冷静的头脑。无论如何，我顶了上去。

在写下这些话的时候，我正在构思我为 *JOOP* 撰写的第 63 期专栏。这个专栏每期或者每两期就会刊登。其间，我也有过非常想中断的时候，非常幸运的是，Jonathan Shopiro 接替了我。偶尔，我只是写一些当期专栏的介绍，然后到卓越的丹麦计算机科学家 Bjørn Stavstrup[1]那里去求助。此外，Livleen Singh 曾跟我谈起为季刊 *C++ Journal* 撰写稿件的事，那个杂志在发行 6 期之后停刊了。Stan Lippman 也甜言蜜语地哄着我在 *C++ Report* 上开了个专栏，当时这本杂志刚刚从一分简陋的通信时刊正式成为成熟的杂志。加上我在 *C++ Report* 上发表的 29 篇专栏文章，我一共发表了 98 篇文章。

[1] 就是 C++创造者 Bjarne Stroustrup，这里可能是丹麦文。——译者注

在这么多的杂志刊物里，分布着大量的材料。如果这些文章单独看来是有用的，那么集结起来应该会更有用。所以，Barbara[1]和我（主要是 Barbara）重新回顾了所有的专栏，选择出其中最好的，并根据一致性和连续性的原则增补和重写了这些文章。

本书正是世界所需的又一本 C++书籍

既然你已经知道了本书的由来，我就再讲讲为什么要读这本书，而不是其他的 C++书籍。天知道！C++方面的书籍太多了，为什么要选这一本呢？

第一个原因是，我想你们会喜欢它。大部分 C++书籍都没有顾及到这点：它们应该是基于科目教学式的。吸引人最多不过是次要目标。

杂志专栏则不同。我猜想肯定会有一些人站在书店里，手里拿着一本 *JOOP*，扫一眼我 Koenig 的专栏之后，便立刻决定购买整本杂志。但是要是我自认为这种情况很多的话，就未免太狂妄自大了。绝大多数读者都是在买了书之后读我的专栏的，也就是说他们有绝对的自由来决定是否读我的专栏。所以，我得让我的每期专栏都货真价实。

本书不对那些晦涩生僻的细节进行琐碎烦人的长篇大论。初学者不应该指望只读这本书就能学会 C++。具备了一定基础的人，比如已经知道几种编程语言的人，以及已经体会到如何通过阅读代码推断出一门新语言的规则的人，将能够通过本书对 C++有所了解。大部分从头开始学的读者先读 Bjarne Stroustrup 的 *The C++ Programming Language*（Addison-Wesley 1991）或者 Stan Lippman 的 *C++ Primer*（Addison-Wesley 1991），然后再读这本书，效果可能会更好。[2]

这是一本关于思想和技术的书，不是关于细节的。如果你试图了解怎样用虚基类实现向后翻腾两周半，就请到别处去找吧。这里所能找到的是许多等待你去阅读分析的代码。请试一试这些范例。根据我们的课堂经验，想办法使这些程序运行起来，然后加以改进，能够很好地巩固你的理解。至于那些更愿意从分析代码开始学习的人，我们也从本书中挑选了一些范例，放在 ftp.aw.com 的目录 cseng/authors/koenig/ruminations 下，可以匿名登录获取。

如果你已经对 C++有所了解，那么本书不仅能让你过一把瘾，而且能对你有所启示。这也是你应该阅读本书的第二个原因。我的意图并不是教 C++本身，而是想告诉你用 C++编程时怎样进行思考，以及如何思考问题并用 C++表述解决方案。知识可以通过系统学习获取，智慧则不能。

---

[1] 本书合作者 Barbara Moo 是 Andrew Koenig 的夫人，退休前是 Bell 实验室高级项目管理人员，曾负责 Fortran 和 CFront 编译器的项目管理。——译者注

[2] 这两本 C++百科大全类的名著分别于 1997 年和 1998 年推出了各自的第三版，Bjarne Stroustrup 还于 2000 年推出了 *The C++ Programming Language* 特别版。——译者注

组织

就专栏来说，我尽力使每期文章都独立成章，但我相信，对于结集来说，如果能根据概念进行编排，将更易于阅读，也更有趣味。因此，本书划分为6篇。

第一篇是对主题的扩展介绍，这些主题将遍布本书的其余部分中。本部分中没有太多的代码，但是所展现的有关抽象和务实的基本思想贯穿本书，更重要的是，这些思想渗透了C++设计原则和应用策略。

第二篇着眼于继承和面向对象编程，大多数人都认为这些是C++中最重要的思想。你将知道继承的重要性何在，它能做什么。你还会知道为什么将继承对用户隐藏起来是有益的，以及什么时候要避免继承。

第三篇探索模板技术，我认为这才是C++里最重要的思想。我之所以这样认为，是因为这些模板提供了一种特别的强大的抽象机制。它们不仅可以构造对所包含的对象类型一无所知的容器，还可以建立远远超出类型范畴的泛型抽象。

继承和模板之所以重要的另一个原因是，它们能够扩展C++，而不必等待（或者雇佣）人去开发新的语言和编译器。进行扩展的方法之一就是通过类库。第四篇谈到了库——包括库的设计和使用。

对基础有了很好的理解以后，我们可以学习第五篇中的一些特殊编程技术了。在这部分，你可以知道如何把类紧密地组合在一起，或者把它们尽可能地分离开。

最后，在第六篇，我们将返回头来对本书所涉及到的内容做一个回顾。

编译和编辑

这些经年累月写出来的文章有一个缺陷，就是它们通常都没有用到语言的现有特性。这就导致了一个问题：我们是应该在C++标准尚未最终定稿的时候，假装ISO C++已经成熟了，然后重写这些专栏，还是维持古迹，保留老掉牙的过时风格呢？[1]

还有许多这样的问题，我们选择了折中。对那些原来的栏目有错的地方——无论是由于后来语言规则的变化而导致的错误，还是由于我们看待事物的方式改变而导致的错误——我们都做了修正。一个很普遍的例子就是对 const 的使用，自从const加入到语言中以来，它的重要性就在我们的意识中日益加强。

另一方面，例如，尽管标准委员会已经接受bool作为内建数据类型，这里大量的范例还是使用 int 来表示真或者假的值。这是因为这些专栏文章早在这之前就完成了，使用 int 作为真、假值还将继续有效，而且要使绝大多数编译器支持bool还需要一些年头。

---

[1] 本书编写于1996年底，当时C++标准已经发布了草案第二版，非常接近最终标准。次年(1997)，C++标准正式定稿。本书内容是完全符合C++标准的。——译者注

致谢

除了在 *JOOP*、*C++ Report*、*C++ Journal* 中发表我们的观点外，我们还在许多地方通过发表讲演（和听取学生的意见）来对它们进行提炼。尤其值得感谢的是 Usenix Association 和 SIGS Publications 举办的会议，以及 *JOOP* 和 *C++ Report* 的发行人。另外，在 Western Institute in Computer Science 的赞助下，我们俩在斯坦福大学讲授过多次单周课程，在贝尔实验室我们为声学研究实验室和网络服务研究实验室的成员讲过课。还有 Dag Brück 曾为我们在瑞典组织了一系列的课程和讲座。Dag Brück 当时在朗德理工学院自动控制系任教，现在在 Dynasim AB。

我们也非常感谢那些阅读过本书草稿以及那些专栏并对它们发表意见的人：Dag Brück, Jim Coplien, Tony Hansen, Bill Hopkins, Brian Kernighan(他曾笔不离手地认真阅读了两遍), Stan Lippman, Rob Murray, George Otto 和 Bjarne Stroustrup。

如果没有以下人员的帮助，这些专栏永远也成不了书。他们是 Deborah Lafferty, Loren Stevens, Addison-Welsey 的 Tom Stone，以及本书编辑 Lyn Dupre。

我们特别感谢 AT&T 开通的经理们，是他们使得编写这些专栏并编辑成书成为可能。他们是 Dave Belanger, Ron Brachman, Jim Finucane, Sandy Fraser, Wayne Hunt, Brian Kernighan, Rob Murray, Ravi Sethi, Bjarne Stroustrup，以及 Eric Sumner。

Andrew Koenig
Barbara Moo
新泽西州吉列
1996 年 4 月

目录