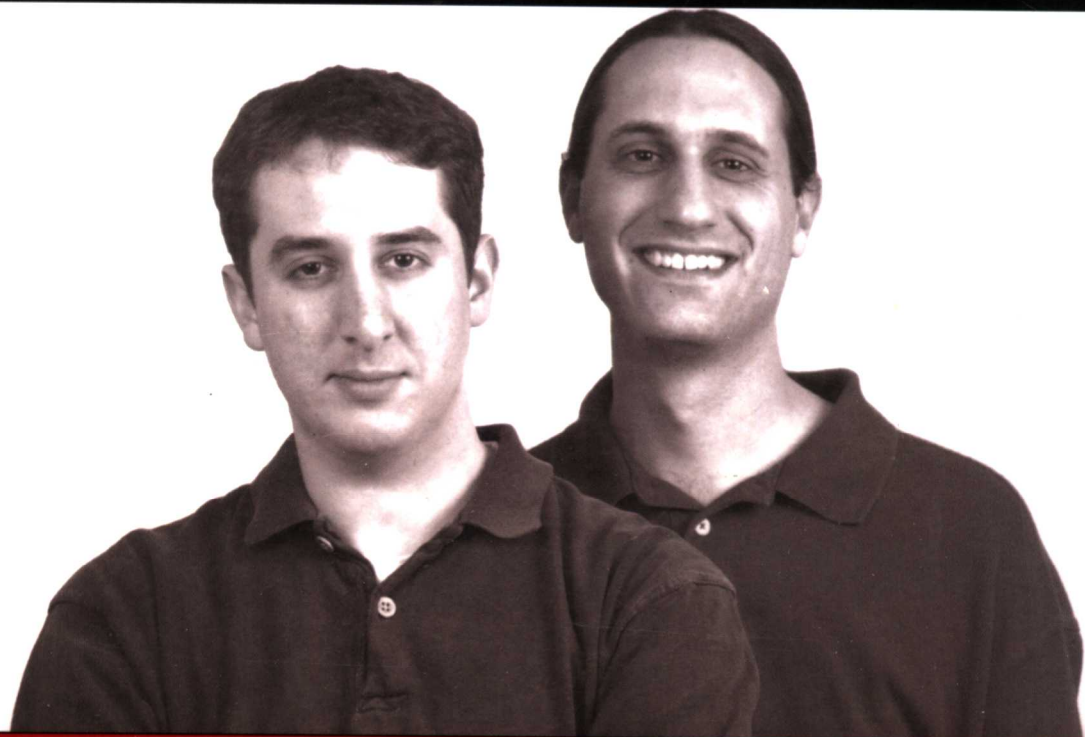




Professional C++

C++高级编程

(美) Nicholas A. Solter 著
Scott J. Kleper

刘鑫 杨健康 等译



机械工业出版社
China Machine Press



Professional C++

C++高级编程

(美) Nicholas A. Solter 著
Scott J. Kleper

刘鑫 杨健康 等译



机械工业出版社
China Machine Press

本书既系统全面又突出重点, 作者从 C++ 基础知识讲起, 始终着眼于 C++ 语言的编程实践, 提供了大量实践示例和解决方案, 包括如何更好地实现重用、如何有效地测试和调试等 C++ 专业人员常用的一些技术与方法, 还提供了一些鲜为人知的、能大大简化工作的 C++ 语言特性; 最后, 还配有大量可重用的编码模式, 并在附录中提供 C++ 面试宝典作为开发人员的实用指南。

本书面向进阶 C++ 的初学者, 以及那些想把 C++ 水平提高到专业水准的程序员和开发人员。

Nicholas A. Solter, Scott J. Kleper: Professional C++ (ISBN: 0-7645-7484-1)

Authorized translation from the English language edition published by John Wiley & Sons, Inc.

Copyright ©2005 by Wiley publishing, Inc.

All rights reserved.

本书中文简体字版由约翰-威利父子公司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。

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: 01-2005-1309

图书在版编目 (CIP) 数据

C++ 高级编程/ (美) 索尔特 (Nicholas A. S.), (美) 凯乐普 (Scott J. K.) 著; 刘鑫等译. —北京: 机械工业出版社, 2006. 1

书名原文: Professional C++

ISBN 7-111-17778-9

I. C… II. ①索… ②凯… ③刘… III. C 语言-程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2005) 第 144910 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: 孙笑竹 姜淑欣

北京瑞德印刷有限公司印刷·新华书店北京发行所发行

2006 年 1 月第 1 版第 1 次印刷

787mm×1092mm 1/16·44 印张

印数: 0 001-4 000 册

定价: 88.00 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

译者序

市场上的 C++ 书籍可谓不少，但面向的读者大多是那些对 C++ 已经很了解的人，还有一些 C++ 书籍更像是参考手册，而不是真正的编程书，并没有真正教我们如何使用 C++。如果读者还不精通 C++，但是想利用它来解决实际问题，这本书就非常适合。它深入浅出地系统介绍了 C++ 的各项高级主题，可以很好地帮助你成为 C++ 专家，而且不要求你先前对 C++ 有太多了解。

本书除了系统、全面的内容介绍外，还讲述了程序设计实践和软件工程，这也是它的另一个闪光点。并不是每个程序员都受过软件工程和软件开发方面的培训，这本书介绍了一些非常好的实践解决方案，告诉我们如何更好地实现重用、如何更快地调试等，这对我们的实际编程尤其有意义。尽早地掌握这些编程实践经验，将有助于编程新手养成良好的编程习惯，即使是具备相当编程经验的人也可以从本书了解到使用 C++ 的更有效方法。

本书有以下特点：

- **重视风格。**如果不注意编程风格，尽管你完全了解 C++，仍有可能写出极糟糕的 C++ 程序，所以这本书中风格问题贯穿始终。
- **突出重点。**本书明确指出了哪些特性很难用、哪些方面很少用。由于 C++ 是一个如此庞大的语言，所以读者要想真正掌握，必须切中要害，强调重点。
- **强调实战。**本书没有太多“玩具型”的小例子，而是提供尽可能多的实践示例，这些示例的代码都可以真正用在你的实际工作中。
- **关注模式。**利用可重用的模式可以编写出更好的代码。本书特别强调了 C++ 程序中反复出现的一些好技术，尤其着很多笔墨来介绍一些可以重用的设计模式。

尽管本书篇幅不短，但是读者读起来一点儿都不会吃力。另外，书中最后还附了一个面试宝典，这是一般的编程书所没有的，这也充分体现出这本书的实用价值。

译者认为，无论本书是作为正式教材还是自学用书，都非常适合。如果你想改进代码质量，提高编程效率，成为一个专业的 C++ 程序员，就千万不要错过这本书。

我们衷心地感谢我们的家人和朋友。在翻译过程中，他们给予了我们莫大的关心、支持和帮助。

全书由刘鑫、杨健康、王林绪、孙健、阎慧、熊伟、朱涛江、王宇、谢剑薇、王树春、韦群、林华君、刘名臣、赵蓓、潘森、刘立强、龚雪晶、王志淋、刘跃邦、蔡洪量、王三梅、何跃强、苏金国、丁小峰、孙春娟、阎文丽、林琪、周兴汉、张练达等进行翻译，其中，刘鑫、杨健康担任主要翻译，王林绪、孙健等进行全书术语的审核，刘名臣、赵蓓等提供技术问题支持，全体工作人员共同完成了本书的翻译工作，最后由刘鑫统稿。

由于时间仓促，且译者的水平有限，在翻译过程中难免会出现一些错误，请读者批评指正。

前言

多年以来，在编写速度快、功能强的企业级面向对象程序时，C++已经成为事实上的标准语言。令人惊讶的是，尽管C++变得如此普及，我们却很难全面地掌握这种语言。一些专业C++程序员会使用一些简单但功能很强大的技术，但以往传统的资料中对此都未曾提及；另外，C++中还有一些有用的部分，这些内容即使是对经验丰富的C++程序员来说可能也很神秘。

通常，编程方面的书更多地强调语言的语法，而不注重讲述如何实际使用这种语言来编程。一般的C++书都会分章介绍C++语言的各个主要部分，来解释相关的语法，并提供一个例子。本书不打算落入这种“俗套”。一般的图书只介绍这种语言方方面面的具体细节，而不关注实践内容，本书则不同，我们的目的很明确，就是要教你如何在实际工作中使用C++。你会从书中了解到一些鲜为人知的特性，这些特性能使你的开发更为轻松；另外这里还提供了一些可重用的编码模式，专业的程序员就是因为掌握了这些模式而从初学者中脱颖而出。

本书读者对象

即使你用C++已经很多年了，对这种语言的一些更为高级的特性可能还是不太熟悉，或者并没有充分利用到C++的全部功能。也许你编写的C++代码确实也能完成任务，但是你还想更多地了解如何完成C++设计，以及怎样才是好的编程风格。也许你是刚刚接触C++的初学者，想有一个好的起点，希望了解怎样才能“正确地”编写程序。本书将使你的C++水平更上一个台阶，达到专业水准。

因为这本书的目的是让你进阶，从对C++只有基本或初步的了解，转变成一名专业的C++程序员，因此我们假设你对这种语言已经有一定的认识了。第1章相当于一个复习，其中介绍了C++的基本知识，不过仅凭这一章，并不能取代踏踏实实的培训和具体地使用这种语言。即使你刚开始学习C++，但C编程的经验很丰富，阅读第1章应该也够了，你需要的大多数知识都能从中找到。无论如何，你都应当有牢固的编程基础，除了应该对循环、函数和变量等内容了如指掌外，还应该知道如何组织程序的结构，对诸如递归等基本技术应该也不陌生。另外，你应当对散列表和队列等常用的数据结构有一定了解，还应该知道排序和查找等有用的算法。当然，你可以不了解面向对象编程，这部分内容将在第3章介绍。

你可以采用任何编译器来开发代码，但必须熟悉所用的编译器。本书不会提供各种编译器的具体用法说明，你可以参考编译器随附的文档来回顾有关的内容。

本书内容

本书提供了一种C++编程方法，这种方法不仅可以改进你的代码质量，还可以提高编程效率。本书不单单讲述C++的语法和语言特性，它还强调了一些编程方法、可重用的设计模式以及好的编程风格。其中，编程方法涵盖了整个软件开发过程，从开始设计和编写代码，到测试、调试和分组工作都有涉及。学完本书，你将掌握C++语言和它的诸多特性，并能充分利用C++的强大功能来完成大规模软件开发。

假设有人已经学过C++的所有语法，但没有见过任何一个简单实例，这就很危险了！没有做过或看

过具体的例子，他可能会认为所有代码都应当放在程序的 `main()` 函数中，或者认为所有变量都应当是全局变量，而通常这些做法都是不好的编程实践。

专业的 C++ 程序员除了了解 C++ 的语法之外，还知道如何正确地使用这种语言。他们认识到好的设计极其重要，并了解面向对象编程理论，知道有哪些最佳的方法来使用现有的库。这些专业的程序员已经开发了大量有用的代码，并提出了许多可重用的思想。

通过阅读本书，你将成为一个专业的 C++ 程序员。你对 C++ 的了解将更为深入，会掌握一些鲜为人知而且通常被误解的语言特性。除此以外，你将学习面向对象程序设计的内容，并获得一些高超的调试技巧。最重要的是，读过这本书后，你的脑海中会留下许多可重用思想，这些思想能够用于实际日常工作当中。

为什么费心尽力地想要成为一个专业的 C++ 程序员，而不是一个只了解 C++ 皮毛的程序员，原因有很多。如果能通晓 C++ 语言的实际工作原理，将大大改善你的代码质量。通过了解不同的编程方法和过程，将有助于你更好地与你的开发小组协作；若能发现可重用的库和常用的设计模式，将有助于提高你的日常工作效率，并避免重蹈覆辙。所有这些，都将使你成为一个更好的程序员和一个更有价值的员工。不过，就算本书没有带给你升迁之喜，多了解一些总不是坏事吧！

本书的组织结构

本书包括 6 大部分。

第一部分，“专业 C++ 程序设计概述”，先提供 C++ 基础知识的快速入门课程，为你奠定一定的 C++ 基础。在入门课程之后，将分析 C++ 设计方法。你会了解到设计的重要性、面向对象方法、库和模式的使用、代码重用的重要性，以及当前为众多编程机构所用的工程实践方法。

第二部分，“编写 C++ 代码方式”，这一部分从专业角度为读者提供了一次 C++ 技术之旅。从中可了解到如何编写可读的 C++ 代码，如何创建可重用的类，以及如何充分利用诸如继承和模板等重要的语言特性。

第三部分，“掌握 C++ 高级特性”，在此介绍了如何更充分地利用 C++。本书这一部分展示了 C++ 的诸多神秘之处，并介绍了如何使用这样一些更高级的特性。在这一部分中你将看到 C++ 语言中一些不常用甚至有些古怪的部分，并了解 C++ 中管理内存有哪些好方法，此外还将学习输入输出技术、专业级错误处理、高级的操作符重载、如何编写高效的 C++ 代码，以及如何编写跨语言和跨平台的代码。

第四部分，“确保无错代码”，这一部分的重点是如何编写企业质量的（enterprise-quality）软件。你将了解一些软件测试概念，如单元测试和回归测试，还将学习调试 C++ 程序时会用到的一些技术。

第五部分，“使用库和模式”，这一部分介绍了库和模式的使用，基于库和模式的编程，不仅可以使你更省力，还可帮助你编写出更好的代码。你将了解 C++ 提供的标准库，包括诸如扩展标准库的一些高级主题。你还将学习分布式对象、可重用 C++ 设计技术和概念上的面向对象设计模式的有关内容。

本书最后一部分对各章提供了一个实用指南，以方便查阅有关的 C++ 技术。在本书相关网站上（www.wrox.com），还能找到 C++ 标准库的一个实用参考指南。

使用本书的前提

要使用这本书，只要有一个安装了 C++ 编译器的计算机就足够了。不同的编译器在对 C++ 语言的解释上往往存在差别，不过本书只关注 C++ 中已经标准化的部分。本书中的所有程序已经在 Windows、Solaris 和 Linux 等平台上成功地通过了测试。

本书约定

为了帮助你更充分地利用这本书，并了解会出现什么情况，我们将采用如下约定：

诸如此类的方框提供了一些不容忘记的重要信息，这些信息与方框前后的文字有很直接的关系。

对当前讨论的主题可能有一些提示、技巧和旁注，这些都将如此缩进并用楷体显示。

正文中还包括以下样式：

- 在初次介绍一些重要的词时，我们会用楷体突出强调。
- 按键采用如下形式表示：Ctrl+A。
- 文件名、URL 和正文中出现的代码表示为：monkey.cpp。
- 代码的表示分为两种：

代码示例中，新出现的代码或重要代码用灰色背景突出强调。

对当前讨论不太重要的代码，或者是前面已经出现过的代码不用灰色背景强调。

源代码

在使用本书中的例子时，你可以手工输入所有代码，也可以直接使用本书随附的源代码文件。本书中用到的所有源代码文件都可以从 www.wrox.com 下载。访问该网站时，只要找到本书的书名（可以使用搜索（Search）框，也可以使用某个书目列表），并点击该书详细信息网页上的下载代码（Download Code）链接，就可以得到本书的所有源代码。

由于会有许多书名雷同，最快捷的方法是利用 ISBN 搜索，本书的 ISBN 是 0-7645-7484-1。

下载了代码之后，只需用你最习惯的压缩工具解压即可。另外也可以前往 Wrox 主站的代码下载网页（www.wrox.com/dynamic/books/download.aspx），在此可以找到这本书以及所有其他 Wrox 书的可用代码。

勘误

尽管我们竭尽所能力争文字和代码不出错，但人无完人，错误在所难免。如果你在书中发现了错误（如拼写错误或者某段代码有误），希望你能及时反馈给我们，对此我们表示深深的谢意。你所提供的勘误可能会帮助另一个读者节省大量时间，否则他可能会因为同一处错误迷惑不解，而徒劳地花费数小时想去搞清楚；与此同时，你的勘误还将有助于我们提供更高质量的书。

要找到本书的勘误页面，请访问 www.wrox.com，并使用搜索（Search）框或者某个书目列表找到这本书的书名。然后，在本书的详细信息页面上，单击本书勘误（Book Errata）链接。在这个页面上，你将看到针对这本书提交并由 Wrox 编辑确认发布的所有勘误。在 www.wrox.com/misc-pages/book-list.shtml 上还可以得到一个完整的水目列表，其中也提供了每本书的勘误。

如果你在本书勘误（Book Errata）页面上没有找到你发现的错误，请访问 www.wrox.com/contact/techsupport.shtml，填写表单，把找到的错误发送给我们。我们将检查你提供的信息，如果错误属实，我们会在本书勘误页面上发布这条信息，并在本书的后续版本中修正这个问题。

p2p.wrox.com

要与作者或其他人讨论有关 C++ 技术问题，请加入 P2P 论坛（p2p.wrox.com）。这个论坛是一个基

于 Web 的系统, 你可以在此发表有关 Wrox 图书和相关技术的消息, 并与其他读者和技术用户交流。论坛提供了一个订购功能, 针对你感兴趣的主题, 如果论坛上新发布了相关的消息, 会通过电子邮件通知你。Wrox 作者、编辑、其他行业专家以及其他读者也会访问这些论坛。

在 <http://p2p.wrox.com> 上, 你会看到许多论坛, 这些论坛不仅可以帮助你阅读本书, 还有助于你开发自己的应用软件。要想加入论坛, 只需遵循以下几个步骤:

1. 访问 p2p.wrox.com, 并单击“注册”(Register) 链接。
2. 阅读使用条文, 并单击“同意”(Agree)。
3. 填写加入论坛的必要信息, 如果想提供其他可选信息, 也可以相应填写, 单击“提交”(Submit)。
4. 你将收到一个电子邮件, 其中将说明如何验证你的账户, 并完成加入过程。

如果你只是阅读论坛中的消息, 无须加入 P2P, 不过, 如果你想发布自己的消息, 就必须加入论坛。

一旦加入, 你就可以发布新的消息了, 还可以对其他用户发布的消息做出响应。任何时刻你都可以在 Web 上阅读消息。如果你希望某个论坛能通过电子邮件向你发送新发布的消息, 请点击论坛列表中该论坛名旁边的“订购此论坛”(Subscribe to this Forum) 图标。

要了解如何使用 Wrox P2P 的更多信息, 请阅读 P2P FAQs, 在此对这个论坛软件工作的问题做了解释, 另外还回答了与 P2P 和 Wrox 书有关的许多常用问题。读者若要阅读 FAQs, 单击任何 P2P 页面上的 FAQ 链接都可以。

致谢

本书能成功问世, 离不开许多人的帮助, 为此我们也欠下了很多人情。这里要感谢 Waterside Productions 的 David Fugate 对我们的谆谆教导, 还要感谢 Wiley 的 Robert Elliot, 是他让一文不名的我们有机会写出这本书, 以一种全新的方式讲述 C++。如果没有策划编辑 Adaobi Obi Tulton 的大力协助, 这本书的出版可能不会像现在这么好。另外要感谢 Kathryn Malm Bourgoine 在编辑方面提供的帮助。封面上的照片巧妙地对我们有所美化, 感谢 Adam Tow 拍摄的这张照片。

还要感谢我们的所有同事和老师, 是诸位多年以来一直在鼓励我们以正确的方法编写代码。特别地, 我们要感谢 Mike Hanson、Maggie Johnson、Adam Nash、Nick Parlante、Bob Plummer、Eric Roberts、Mehran Sahami、Bill Walker、Dan Walkowski、Patrick Young 和 Julie Zelenski。还要向 Jerry Cain 致以诚挚的谢意, 是他最早教我们学 C++, 不仅如此, 他还担任了本书的技术编辑, 一丝不苟地分析了本书中的代码, 认真的程度就好像在批阅我们的期末试卷一样。

另外要感谢: Rob Baesman、Aaron Bradley、Elaine Cheung、Marni Kleper、Toli Kuznets、Akshay Rangnekar、Eltefaat Shokri、Aletha Solter、Ken Solter 和 Sonja Solter, 各位都审阅了书中的一章甚至多章。当然, 如果书中还存在错误, 这都要归咎于我们自己。在此还要向各位的家人表示感谢, 谢谢大家的耐心和支持。

最后, 我们还要感谢你, 亲爱的读者, 谢谢你打算采用我们提供的方法踏上专业 C++ 开发的道路。

作者简介

Nicholas A. Solter 曾在斯坦福大学攻读计算机专业，获得了理学学士和理学硕士学位，并致力于多个系统的开发。他还是一个学生时，就担任了多门计算机课程的助教，这些课程门类众多，下至为非计算机专业开设的入门性课程，上至有关项目组织和软件工程的高级课程。

Nick 现在是一名软件工程师，任职于 Sun Microsystems 公司，在工作中他主要用 C 和 C++ 编程来开发高可用性软件。他还在计算机游戏行业有过工作经历。在 Digital Media International 公司工作期间，他曾担任多媒体教学游戏“Land Before Time Math Adventure”的首席程序员。在 Electronic Arts 公司实习时，他协助开发了 Course Architect 2000 高尔夫教程，高尔夫界大名鼎鼎的“老虎”伍兹参加 PGA（美国职业高尔夫球协会）Tour 2000 比赛时就使用了这个工具。

除了行业开发经验外，Nick 还作为兼职教授在 Fullerton 学院讲授了一年 C++ 课程。闲暇时，Nick 很喜欢看书、玩篮球、照顾他的儿子 Kai，以及和家人在一起享受天伦之乐。

Scott J. Kleper 在初中时就开始了他的编程生涯，那时他用 BASIC 为 Tandy TRS-80 编写了一些冒险游戏。由于他所在的高中 Mac 大行其道，Scott 也因此开始转向更高级的语言，并发布过许多荣获大奖的共享应用软件。

Scott 也考入了斯坦福大学，在那里他获得了计算机科学专业的理学学士和理学硕士学位，并致力于人机交互领域。上大学期间，Scott 担任了一些课程的助教，包括程序设计入门、面向对象程序设计、数据结构、GUI 框架、项目组织和 Internet 编程等。

毕业以后，Scott 担任了多家公司开发小组的首席工程师，目前是 Reactivity 公司的一名高级软件工程师。工作之余，Scott 特别喜欢网上购物和读书，他还是一个不错的吉它手。

特别鸣谢

副总裁、主管集团出版商

Richard Swadley

副总裁、出版商

Joseph B. Wikert

执行编辑

Robert Elliott

编辑经理

Kathryn Malm Bourgoine

高级制作编辑

Geraldine Fahey

高级策划编辑

Adaobi Obi Tulton

制作编辑

Felicia Robinson

媒体开发专家

Richard Graves

技术编辑

Jerry Cain

文字设计与合成

Wiley Composition Services

封面摄影

Adam Tow

目 录

译者序	
前言	
作者简介	

第一部分 专业 C++ 程序设计概述

第 1 章 C++ 快速入门	1
1.1 C++ 基础	1
1.1.1 必不可少的“Hello, World”	1
1.1.2 命名空间	4
1.1.3 变量	5
1.1.4 操作符	7
1.1.5 类型	9
1.1.6 条件语句	10
1.1.7 循环	13
1.1.8 数组	14
1.1.9 函数	14
1.1.10 结束语	15
1.2 C++ 进阶	16
1.2.1 指针和动态内存	16
1.2.2 C++ 中的字符串	18
1.2.3 引用	20
1.2.4 异常	21
1.2.5 const 的多重用途	22
1.3 作为一种面向对象语言的 C++	23
1.4 你的第一个实用的 C++ 程序	25
1.4.1 一个员工记录系统	26
1.4.2 Employee 类	26
1.4.3 Database 类	30
1.4.4 用户界面	34
1.4.5 对程序的评价	36
1.5 小结	36
第 2 章 设计专业的 C++ 程序	37

2.1 什么是编程设计	37
2.2 编程设计的重要性	38
2.3 C++ 设计有什么不同之处	39
2.4 C++ 设计的两个原则	40
2.4.1 抽象	40
2.4.2 重用	41
2.5 设计一个象棋程序	43
2.5.1 需求	43
2.5.2 设计步骤	43
2.6 小结	47
第 3 章 基于对象的设计	48
3.1 面向对象的世界观	48
3.1.1 我是在以过程性思维思考吗	48
3.1.2 面向对象思想	49
3.1.3 身处对象世界中	51
3.1.4 对象关系	52
3.1.5 抽象	61
3.2 小结	63
第 4 章 基于库和模式的设计	64
4.1 重用代码	64
4.1.1 有关术语	64
4.1.2 决定是否重用代码	65
4.1.3 重用代码的策略	67
4.1.4 捆绑第三方应用	70
4.1.5 开源库	70
4.1.6 C++ 标准库	71
4.2 利用模式和技术完成设计	81
4.2.1 设计技术	81
4.2.2 设计模式	82
4.3 小结	83
第 5 章 重用设计	84
5.1 重用方法论	84

5.2 如何设计可重用的代码	85	7.5.1 使用常量	119
5.2.1 使用抽象	85	7.5.2 利用 const 变量	120
5.2.2 适当地建立代码结构以优化重用	86	7.5.3 使用引用而不是指针	120
5.2.3 设计可用的接口	89	7.5.4 使用定制异常	120
5.2.4 协调一般性和易用性	93	7.6 格式化	121
5.3 小结	94	7.6.1 有关大括号对齐的争论	121
第6章 充分利用软件工程方法	95	7.6.2 考虑空格和小括号	122
6.1 为什么需要过程	95	7.6.3 空格和制表符	122
6.2 软件生命期模型	96	7.7 风格方面的难题	122
6.2.1 分阶段模型和瀑布模型	96	7.8 小结	123
6.2.2 螺旋方法	98	第8章 掌握类和对象	124
6.2.3 统一开发过程	100	8.1 电子表格示例	124
6.3 软件工程方法论	101	8.2 编写类	124
6.3.1 极限编程(XP)	101	8.2.1 类定义	124
6.3.2 软件 triage	104	8.2.2 定义方法	127
6.4 建立自己的过程和方法论	104	8.2.3 使用对象	130
6.4.1 以开放的心态接纳新思想	105	8.3 对象生命期	131
6.4.2 汇总新思想	105	8.3.1 对象创建	131
6.4.3 明确哪些可行, 哪些不可行	105	8.3.2 对象撤销	140
6.4.4 不要做叛逆者	105	8.3.3 对象赋值	141
6.5 小结	105	8.3.4 区别复制和赋值	143
第二部分 编写 C++ 代码方式		8.4 小结	144
第7章 好的编码风格	107	第9章 精通类和对象	145
7.1 为什么代码看上去要好	107	9.1 对象中的动态内存分配	145
7.1.1 提前考虑	107	9.1.1 Spreadsheet 类	145
7.1.2 保持清晰	107	9.1.2 用析构函数释放内存	147
7.1.3 好的代码风格包括哪些因素	108	9.1.3 处理复制和赋值	147
7.2 为代码加注释	108	9.2 不同类型的数据成员	154
7.2.1 写注释的原因	108	9.2.1 静态数据成员	154
7.2.2 注释风格	111	9.2.2 const 数据成员	156
7.2.3 本书中的注释	115	9.2.3 引用数据成员	157
7.3 分解	115	9.2.4 const 引用数据成员	158
7.3.1 通过重构来分解	115	9.3 深入了解方法	158
7.3.2 根据设计来分解	116	9.3.1 静态方法	158
7.3.3 本书中的分解	117	9.3.2 const 方法	159
7.4 命名	117	9.3.3 方法重载	161
7.4.1 选择一个好名字	117	9.3.4 默认参数	162
7.4.2 命名约定	118	9.3.5 内联方法	163
7.5 合理地使用语言特性	119	9.4 嵌套类	164
		9.5 友元	166
		9.6 操作符重载	166

9.6.1 实现加法	166	11.1 模板概述	219
9.6.2 重载算术操作符	170	11.2 类模板	220
9.6.3 重载比较操作符	172	11.2.1 编写类模板	220
9.6.4 利用操作符重载构建类型	174	11.2.2 编译器如何处理模板	227
9.7 方法和成员指针	174	11.2.3 模板代码在文件之间的分布	228
9.8 构建抽象类	175	11.2.4 模板参数	229
9.9 小结	178	11.2.5 方法模板	231
第 10 章 探索继承技术	179	11.2.6 模板类特殊化	235
10.1 使用继承构建类	179	11.2.7 从模板类派生子类	239
10.1.1 扩展类	179	11.2.8 继承与特殊化的区别	240
10.1.2 覆盖方法	182	11.3 函数模板	240
10.2 继承以实现重用	184	11.3.1 函数模板特殊化	241
10.2.1 类 WeatherPrediction	184	11.3.2 函数模板的重载	242
10.2.2 在子类中增加功能	185	11.3.3 类模板的友元函数模板	243
10.2.3 在子类中进行功能替换	187	11.4 高级模板	244
10.3 考虑父类	187	11.4.1 关于模板参数的更多知识	244
10.3.1 父构造函数	187	11.4.2 模板类的部分特殊化	251
10.3.2 父析构函数	189	11.4.3 用重载模板函数部分特殊化	256
10.3.3 引用父类的数据	191	11.4.4 模板递归	257
10.3.4 向上类型强制转换和向下类型 强制转换	192	11.5 小结	264
10.4 继承以实现多态	193	第 12 章 理解 C++ 疑难问题	265
10.4.1 Spreadsheet 的返回结果	193	12.1 引用	265
10.4.2 设计多态电子表格单元格	194	12.1.1 引用变量	265
10.4.3 电子表格单元格的基类	194	12.1.2 引用数据成员	267
10.4.4 各个子类	196	12.1.3 引用参数	267
10.4.5 充分利用多态	198	12.1.4 引用返回类型	268
10.4.6 将来的考虑	199	12.1.5 采用引用还是指针	268
10.5 多重继承	200	12.2 关键字疑点	270
10.5.1 从多个类继承	200	12.2.1 关键字 const	270
10.5.2 命名冲突与二义基类	201	12.2.2 关键字 static	273
10.6 继承技术中有趣而隐蔽的问题	203	12.2.3 非局部变量的初始化顺序	276
10.6.1 改变覆盖方法的特性	203	12.3 类型与类型强制转换	276
10.6.2 覆盖方法的特殊情况	206	12.3.1 typedef	276
10.6.3 复制构造函数与相等操作符	212	12.3.2 类型强制转换	277
10.6.4 关键字 virtual 的真相	213	12.4 解析作用域	281
10.6.5 运行时类型工具	215	12.5 头文件	282
10.6.6 非公共继承	217	12.6 C 实用工具	283
10.6.7 虚基类	217	12.6.1 变量长度参数列表	283
10.7 小结	218	12.6.2 预处理宏	285
第 11 章 利用模板编写通用代码	219	12.7 小结	285

第三部分 掌握 C++ 高级特性

第 13 章 有效的内存管理	287	14.5.1 宽字符	329
13.1 使用动态内存	287	14.5.2 非西方字符集	329
13.1.1 如何描述内存	287	14.5.3 本地化环境与方面	330
13.1.2 内存的分配与撤销	289	14.6 小结	332
13.1.3 数组	291	第 15 章 处理错误	333
13.1.4 使用指针	297	15.1 错误和异常	333
13.2 数组与指针的对应	299	15.1.1 到底什么是异常	333
13.2.1 数组即指针	299	15.1.2 C++ 中的异常为什么好	334
13.2.2 指针并非都是数组	300	15.1.3 C++ 中的异常为什么不好	335
13.3 动态字符串	300	15.1.4 我们的建议	335
13.3.1 C 风格的字符串	301	15.2 异常机制	335
13.3.2 字符串直接量	302	15.2.1 抛出和捕获异常	336
13.3.3 C++ 的字符串类	303	15.2.2 异常类型	338
13.4 低级的内存操作	304	15.2.3 抛出和捕获多个异常	339
13.4.1 指针运算	304	15.2.4 未捕获的异常	342
13.4.2 自定义内存管理	305	15.2.5 抛出列表	343
13.4.3 垃圾回收	305	15.3 异常和多态	346
13.4.4 对象池	306	15.3.1 标准异常层次体系	346
13.4.5 函数指针	306	15.3.2 按类层次捕获异常	348
13.5 常见的内存陷阱	308	15.3.3 编写自己的异常类	349
13.5.1 字符串空间分配不足	308	15.4 栈展开和清除	351
13.5.2 内存泄漏	309	15.4.1 捕获、清除和重新抛出	353
13.5.3 二次删除与无效指针	311	15.4.2 使用智能指针	353
13.5.4 访问越界指针	312	15.5 常见的错误处理问题	354
13.6 小结	312	15.5.1 内存分配错误	354
第 14 章 揭开 C++ I/O 的神秘		15.5.2 构造函数中的错误	356
面纱	313	15.5.3 析构函数中的错误	357
14.1 使用流	313	15.6 综合	357
14.1.1 到底什么是流	313	15.7 小结	359
14.1.2 流的源与目的	313		
14.1.3 流输出	314	第四部分 确保无错代码	
14.1.4 流输入	317	第 16 章 重载 C++ 操作符	361
14.1.5 输入与输出对象	321	16.1 操作符重载概述	361
14.2 字符串流	323	16.1.1 为什么要重载操作符	362
14.3 文件流	324	16.1.2 操作符重载的限制	362
14.3.1 使用 seek() 与 tell()	325	16.1.3 操作符重载中的选择	362
14.3.2 链接流	327	16.1.4 不应重载的操作符	364
14.4 双向 I/O	327	16.1.5 可重载操作符小结	364
14.5 国际化	329	16.2 重载算术操作符	367
		16.2.1 重载一元减和一元加	367
		16.2.2 重载自增和自减	368

16.3 重载位操作符和二元逻辑操作符	369	18.1.2 实现问题	413
16.4 重载插入和析取操作符	369	18.1.3 特定于平台的特性	414
16.5 重载下标操作符	371	18.2 跨语言开发	415
16.5.1 利用 operator[] 提供只读 访问	373	18.2.1 混合 C 和 C++	415
16.5.2 非整数数组索引	375	18.2.2 转换模式	415
16.6 重载函数调用操作符	375	18.2.3 与 C 代码的链接	418
16.7 重载解除引用操作符	377	18.2.4 利用 JNI 混合 Java 和 C++	419
16.7.1 实现 operator*	378	18.2.5 C++ 与 Perl 和 Shell 脚本的 混合	421
16.7.2 实现 operator->	379	18.2.6 C++ 与汇编代码的混合	424
16.7.3 到底什么是 operator->*	379	18.3 小结	424
16.8 编写转换操作符	380	第 19 章 熟练地测试	425
16.8.1 转换操作符的二义性问题	381	19.1 质量控制	425
16.8.2 布尔表达式的转换	382	19.1.1 谁来负责测试	425
16.9 重载内存分配和撤销操作符	383	19.1.2 bug 的生命期	425
16.9.1 new 和 delete 究竟如何工作	384	19.1.3 bug 跟踪工具	427
16.9.2 重载 operator new 和 operator delete	385	19.2 单元测试	428
16.9.3 重载带额外参数的 operator new 和 operator delete	387	19.2.1 单元测试的方法	428
16.10 小结	389	19.2.2 单元测试过程	429
第 17 章 编写高效的 C++ 程序	390	19.2.3 实战单元测试	431
17.1 性能和效率概述	390	19.3 高级测试	439
17.1.1 实现高效的两种方法	390	19.3.1 集成测试	439
17.1.2 两类程序	390	19.3.2 系统测试	440
17.1.3 C++ 是一种低效语言吗	391	19.3.3 回归测试	441
17.2 语言级效率	391	19.4 成功测试的提示	441
17.2.1 高效地处理对象	392	19.5 小结	442
17.2.2 不要过度使用高开销的语言 特性	394	第 20 章 征服调试	443
17.2.3 使用内联方法和函数	395	20.1 调试基本法则	443
17.3 设计级效率	396	20.2 bug 分类	443
17.3.1 尽可能缓存	396	20.3 避免 bug	443
17.3.2 使用对象池	397	20.4 找出 bug 的方法	444
17.3.3 使用线程池	402	20.4.1 错误日志	444
17.4 测评分析	402	20.4.2 调试轨迹	445
17.5 小结	410	20.4.3 断言	455
第 18 章 开发跨平台和跨语言的应用	411	20.5 调试技术	456
18.1 跨平台开发	411	20.5.1 再生 bug	456
18.1.1 体系结构问题	411	20.5.2 调试可再生 bug	456
		20.5.3 调试不可再生 bug	457
		20.5.4 调试内存问题	457
		20.5.5 调试多线程程序	461

20.5.6 调试示例：文章引用	461	22.2.5 编写自己的函数对象	534
20.5.7 从 ArticleCitations 示例学到的 教训	472	22.3 算法细节	535
20.6 小结	472	22.3.1 工具算法	536
第 21 章 深入 STL：容器和迭代器	473	22.3.2 非修改算法	536
21.1 容器概述	473	22.3.3 修改算法	542
21.1.1 元素需求	474	22.3.4 排序算法	545
21.1.2 异常和错误检查	474	22.3.5 集合算法	547
21.1.3 迭代器	475	22.4 算法和函数对象示例：选民注册 审计	549
21.2 顺序容器	476	22.4.1 选民注册审计问题描述	549
21.2.1 vector	476	22.4.2 auditVoterRolls() 函数	549
21.2.2 vector<bool> 特殊化	492	22.4.3 getDuplicates() 函数	550
21.2.3 deque	493	22.4.4 RemoveNames 函数对象	551
21.2.4 list	493	22.4.5 NameInList 函数对象	552
21.3 容器适配器	496	22.4.6 测试 auditVoterRolls() 函数	553
21.3.1 queue	496	22.5 小结	554
21.3.2 priority_queue	499	第 23 章 定制和扩展 STL	555
21.3.3 stack	502	23.1 分配器	555
21.4 关联容器	503	23.2 迭代器适配器	556
21.4.1 pair 工具类	503	23.2.1 逆序迭代器	556
21.4.2 map	504	23.2.2 流迭代器	557
21.4.3 multimap	511	23.2.3 插入迭代器	557
21.4.4 set	515	23.3 扩展 STL	559
21.4.5 multiset	517	23.3.1 为什么要扩展 STL	559
21.5 其他容器	517	23.3.2 编写 STL 算法	559
21.5.1 数组作为 STL 容器	517	23.3.3 编写一个 STL 容器	561
21.5.2 string 作为 STL 容器	518	23.4 小结	587
21.5.3 流作为 STL 容器	519	第 24 章 探讨分布式对象	588
21.5.4 bitset	519	24.1 分布式计算的魔力	588
21.6 小结	524	24.1.1 分布以获得可扩展性	588
24.1.2 分布以获得可靠性	589	24.1.3 分布以获得集中性	589
24.1.4 分布式内容	589	24.1.5 分布式 vs 网络式	589
24.2 分布式对象	590	24.2 串行化和编组	590
24.2.1 串行化和编组	590	24.2.2 远程过程调用	593
24.3 CORBA	595	24.3 接口定义语言	595
24.3.1 接口定义语言	595	24.3.2 实现类	597
24.3.2 实现类	597	24.3.3 使用对象	598
24.3.3 使用对象	598		

第五部分 使用库和模式

第 22 章 掌握 STL 算法和函数对象

22.1 算法概述	525
22.1.1 find() 和 find_if() 算法	526
22.1.2 accumulate() 算法	528
22.2 函数对象	529
22.2.1 算术函数对象	529
22.2.2 比较函数对象	530
22.2.3 逻辑函数对象	531
22.2.4 函数对象适配器	531

24.4 XML	602	26.1.3 使用单例	645
24.4.1 XML 快速入门	602	26.2 工厂模式	646
24.4.2 XML 作为一种分布式对象 技术	604	26.2.1 举例：汽车工厂模拟	646
24.4.3 用 C++ 生成和解析 XML	604	26.2.2 实现工厂	648
24.4.4 XML 验证	612	26.2.3 使用工厂	650
24.4.5 用 XML 构建分布式对象	613	26.2.4 工厂的其他使用	651
24.4.6 SOAP（简单对象访问 协议）	616	26.3 代理模式	652
24.5 小结	618	26.3.1 举例：隐藏网络连通性问题	652
第 25 章 结合技术和框架	619	26.3.2 实现代理	652
25.1 “我想不起来如何...”	619	26.3.3 使用代理	653
25.1.1编写一个类	619	26.4 适配器模式	653
25.1.2派生一个现有类	621	26.4.1 举例：适配一个 XML 库	653
25.1.3抛出和捕获异常	622	26.4.2 适配器的实现	654
25.1.4读文件	622	26.4.3 使用适配器	657
25.1.5写文件	623	26.5 装饰器模式	658
25.1.6编写模板类	623	26.5.1 举例：定义网页中的样式	658
25.2 还有更好的办法	625	26.5.2 装饰器的实现	659
25.2.1 带引用计数的智能指针	625	26.5.3 使用装饰器	660
25.2.2 双重分派	630	26.6 职责链模式	661
25.2.3 混合类	635	26.6.1 举例：事件处理	661
25.3 面向对象框架	637	26.6.2 职责链的实现	661
25.3.1 使用框架	637	26.6.3 使用职责链	662
25.3.2 模型-视图-控制器模式	638	26.7 观察者模式	662
25.4 小结	639	26.7.1 举例：事件处理	663
第 26 章 应用设计模式	640	26.7.2 实现观察者	663
26.1 单例模式	640	26.7.3 使用观察者	664
26.1.1 举例：日志机制	640	26.8 小结	665
26.1.2 单例的实现	641		
		附 录	
		附录 A C++ 面试宝典	666
		附录 B 参考书目	681

第一部分

专业 C++ 程序设计概述

第1章 C++ 快速入门

本章的目标很明确，即简要地介绍 C++ 中最重要的一些环节，使读者对 C++ 有一个基本的了解，以便更好地学习书中余下的内容。本章可不是详尽介绍 C++ 程序设计语言的全面教程，例如“程序是什么”、“= 和 == 之间有什么区别”之类的基本问题这里并不涉及。对于一些深奥的问题，比如“什么是 union?”、“volatile 关键字是怎么回事?”，我们也不打算深入探究。C 语言里的某些部分与 C++ 的关系并不大，另外 C++ 的某些部分会在本书的后续章节做深入的介绍，所以这些内容在本章中也不会出现。

本章就是要介绍每天与程序员“打交道”的基本 C++ 知识。如果你好久没有用 C++ 了，忘记了 for 循环的语法，那么可以在这一章中找回“记忆”。如果你是头一次接触 C++，还不知道什么是引用变量，也可以在这里了解到有关内容。

如果你很熟悉 C++，而且已经有相当丰富的经验了，可以快速地浏览一下本章，看看其中哪些关于 C++ 的基本知识是以前没有注意到、而需要掌握的。如果你是一个 C++ 新手，请花些时间仔细阅读本章，一定要完全掌握本章中的所有例子。如果你还需要更多的相关信息，可以参阅附录 B 中列出的参考书目。

1.1 C++ 基础

C++ 语言通常被视作一种“更好的 C”，或者是“C 的超集”。C 语言中许多麻烦的地方、含糊不清的地方在设计 C++ 时得到了解决。由于 C++ 是基于 C 的，如果你是一个熟练的 C 程序员，就会发现这一节中看到的许多语法都似曾相识。不过，这两种语言当然也存在着差别。C++ 之父 Bjarne Stroustrup 所著的《The C++ Programming Language》厚达 911 页，而 Kernighan 和 Ritchie 编写的《The C Programming Language》只有区区 274 页，由此就可见一斑。所以，如果你是一位 C 程序员，就要当心你没见过的、不熟悉的语法！

1.1.1 必不可少的“Hello, World”

可以说，下面这段代码可能是你遇到的最简单的 C++ 程序了。（译者注：介绍语言的书几乎都从“Hello, World”开始举例，所以这一节标题为必不可少的“Hello, World”。）