HZ BOOKS
华章教育

计 算 机 科 学 丛 书

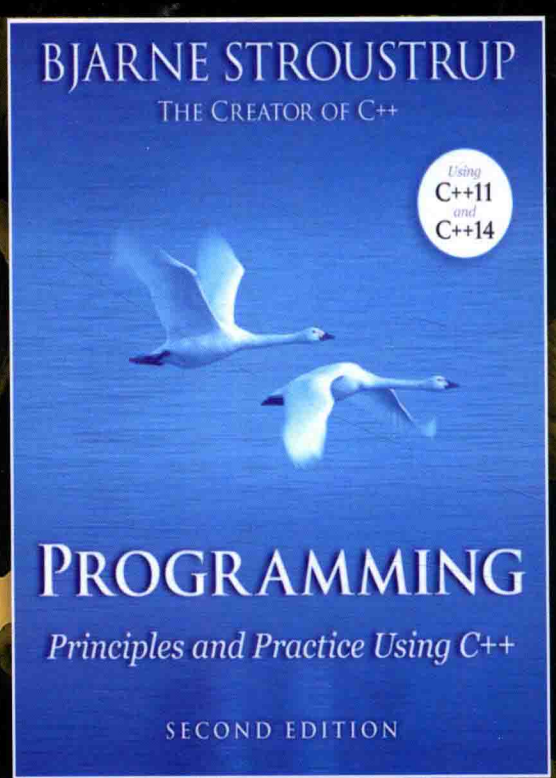
Pearson

原书第2版

C++程序设计
原理与实践
（基础篇）

[美] 本贾尼·斯特劳斯特鲁普（Bjarne Stroustrup）著 任明明 王刚 李忠伟 译

Programming
Principles and Practice Using C++ Second Edition



机械工业出版社
China Machine Press

计 算 机 科 学 丛 书

原书第2版

C++[redacted]设计

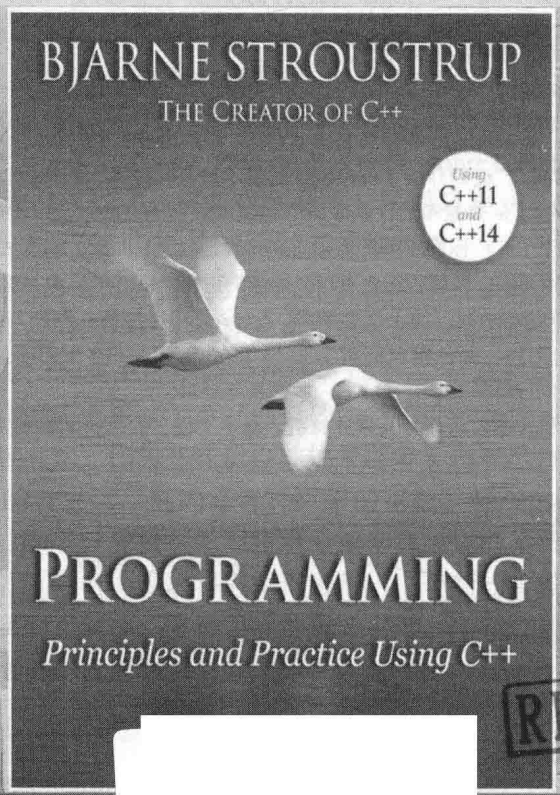
原理与实践

（基础篇）

[美] 本贾尼·斯特劳斯特鲁普（Bjarne Stroustrup）著 任明明 王刚 李忠伟 译

Programming

Principles and Practice Using C++ Second Edition



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

C++ 程序设计：原理与实践（基础篇）（原书第 2 版）/（美）本贾尼·斯特劳斯特鲁普（Bjarne Stroustrup）著；任明明，王刚，李忠伟译．—北京：机械工业出版社，2017.3
（计算机科学丛书）
书名原文：Programming：Principles and Practice Using C++，Second Edition

ISBN 978-7-111-56225-2

I. C… II. ①本… ②任… ③王… ④李… III. C 语言－程序设计－高等学校－教材
IV. TP312.8

中国版本图书馆 CIP 数据核字（2017）第 040462 号

本书版权登记号：图字：01-2014-5459

C++ 之父 Bjarne Stroustrup 的经典著作《C++ 程序设计：原理与实践（原书第 2 版）》基于最新的 C++11 和 C++14，广泛地介绍了程序设计的基本概念和技术，包括类型系统、算术运算、控制结构、错误处理等；介绍了从键盘和文件获取数值和文本数据的方法以及以图形化方式表示数值数据、文本和几何图形；介绍了 C++ 标准库中的容器（如向量、列表、映射）和算法（如排序、查找和内积）的设计和使用。同时还对 C++ 思想和历史进行了详细的讨论，很好地拓宽了读者的视野。

为方便读者循序渐进地学习，加上篇幅所限，《C++ 程序设计：原理与实践（原书第 2 版）》分为基础篇和进阶篇两册出版，基础篇包括第 1 ~ 11 章、第 17 ~ 19 章和附录 A、C，进阶篇包括第 12 ~ 16 章、第 20 ~ 27 章和附录 B、D、E。本书是基础篇。

本书通俗易懂、实例丰富，可作为大学计算机、电子工程、信息科学等相关专业的教材，也可供相关专业人员参考。

出版发行：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：和　静　　责任校对：李秋荣
印　　刷：北京诚信伟业印刷有限公司　　版　　次：2017 年 4 月第 1 版第 1 次印刷
开　　本：185mm×260mm　1/16　　印　　张：27
书　　号：ISBN 978-7-111-56225-2　　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　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson, McGraw-Hill, Elsevier, MIT, John Wiley & Sons, Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S.Tanenbaum, Bjarne Stroustrup, Brian W.Kernighan, Dennis Ritchie, Jim Gray, Afred V.Aho, John E.Hopcroft, Jeffrey D.Ullman, Abraham Silberschatz, William Stallings, Donald E.Knuth, John L.Hennessy, Larry L.Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037

华章科技图书出版中心

译者序

Programming: Principles and Practice Using C++, Second Edition

程序设计是打开计算机世界大门的金钥匙，它使五花八门的软件对你来说不再是“魔法”。C++ 语言则是掌握这把金钥匙的有力武器，它优美、高效，从大洋深处到火星表面，从系统核心到高层应用，从掌中的手机到超级计算机，到处都有 C++ 程序的身影。本书的目标不是作为程序设计语言的简单入门教材，而是成为初学者学习基础实用编程技术的绝佳启蒙。如果你愿意努力学习，本书能帮助你理解使用 C++ 语言进行程序设计的基本原理及大量实践技巧，其中大多数可直接用于其他程序设计语言。基于这一目标，注重实践是本书的明显特点。它希望教会你编写真正能被他人所使用的“有用的程序”，而非“玩具程序”。因此，本书不是机械地介绍各种 C++ 特性，而是针对一些具体问题，不断精化其求解方案，在这个过程中自然地引出基本编程技术及相应的 C++ 程序特性。此外，本书还介绍了大量的求解实际问题的程序设计技术，如语法分析器的设计、图形化程序设计、利用正则表达式处理文本、数值计算程序设计以及嵌入式程序设计等。在其他大多数程序设计入门书籍中，是找不到这些内容的。像调试技术、测试技术等其他程序设计书籍着墨不多的话题，本书也有详细的介绍。程序设计远非遵循语法规则和阅读手册那么简单，而在于理解基本思想、原理和技术，并进行大量实践。本书阐述了这一理念，为如何才能达到编写有用的、优美的程序这一最终目标指引了明确的方向。

本书的作者 Bjarne Stroustrup 是 C++ 语言的设计者和最初的实现者，也是《The C++ Programming Language》(Addison-Wesley 出版社）一书的作者。他现在是摩根斯坦利技术部门的总经理和哥伦比亚大学的客座教授，美国国家工程院的院士，ACM 会士和 IEEE 会士。在进入学术界之前，他为 AT&T 贝尔实验室工作多年。他是 ISO 标准组织 C++ 委员会的创建者，现在是该委员会语言演化工作组的主席。本书第 1 版已成为程序设计领域的经典著作，第 2 版又进行了精心的修订，增加了一些新的内容，包括 C++14 的一些新特性。

虽然是面向初学者，但本书原版仍是大部头。为方便读者循序渐进地学习，我们重新组织了章节顺序，将原版书组织为基础篇和进阶篇两册。基础篇包括第 1 ~ 11 章、第 17 ~ 19 章和附录 A、C，进阶篇包括第 12 ~ 16 章、第 20 ~ 27 章和附录 B、D、E。

基础篇	原　书	进阶篇	原　书
第 1 章	第 1 章	第 15 章	第 20 章
第 2 章	第 2 章	第 16 章	第 21 章
第 3 章	第 3 章	第 17 章	第 12 章
第 4 章	第 4 章	第 18 章	第 13 章
第 5 章	第 5 章	第 19 章	第 14 章
第 6 章	第 6 章	第 20 章	第 15 章
第 7 章	第 7 章	第 21 章	第 16 章
第 8 章	第 8 章	第 22 章	第 22 章
第 9 章	第 9 章	第 23 章	第 23 章
第 10 章	第 10 章	第 24 章	第 24 章

（续）

基础篇	原　书	进阶篇	原　书
第 11 章	第 11 章	第 25 章	第 25 章
第 12 章	第 17 章	第 26 章	第 26 章
第 13 章	第 18 章	第 27 章	第 27 章
第 14 章	第 19 章	附录 C	附录 B
附录 A	附录 A	附录 D	附录 D
附录 B	附录 C	附录 E	附录 E

基础篇逻辑上分成四部分：第一部分介绍基本的 C++ 程序设计知识，包括第一个“Hello, World!”程序、对象、类型、值、计算、错误处理、函数、类等内容，以及一个计算器程序实例；第二部分介绍字符方式输入输出，包括输入输出流的基本概念和格式化输出方法；第三部分介绍数据结构的基本知识，重点介绍向量以及自由内存空间、数组、模板和异常；第四部分为附录，介绍了 C++ 语言概要和 Visual Studio 简要入门。通过基础篇的学习，读者可掌握 C++ 最基本的语言特性，以及运用这些特性编写高质量程序的基本技巧。

在此基础上，进阶篇希望帮助读者学习一些更高级的编程技术及相应的 C++ 语言特性，也逻辑上分成四部分：第一部分为数据结构和算法进阶知识，介绍容器和迭代器以及算法和映射；第二部分深入讨论输入输出，介绍图形 /GUI 类和图形化程序设计；第三部分希望拓宽读者的视野，介绍程序设计语言的理念和历史、文本处理技术、数值计算、嵌入式程序设计技术及测试技术，此外还较为详细地介绍了 C 语言与 C++ 的异同；第四部分为附录，包括标准库概要、FLTK 安装以及 GUI 实现等内容。

本书的引言、第 1 章以及第 2 ~ 9 章由任明明翻译，第 10、11 章和第 17 ~ 21 章由李忠伟翻译，第 22 ~ 27 章由刘晓光翻译，第 12 ~ 16 章以及前言、附录等由王刚翻译。翻译大师经典，难度超乎想象。接受任务之初，诚惶诚恐；翻译过程中，如履薄冰；完成后，忐忑不安。虽然竭尽全力，但肯定还有很多错漏之处，敬请读者批评指正。

感谢机械工业出版社华章公司的温莉芳总编辑将此重任交付译者，感谢朱劼等老师为本书所付出的心血，没有她们辛苦的编辑和审校，本书不可能完成。

译者

2016 年 11 月于南开大学

前 言

Programming: Principles and Practice Using C++, Second Edition

该死的鱼雷！全速前进。

——Admiral Farragut

程序设计是这样一门艺术，它将问题求解方案描述成计算机可以执行的形式。程序设计中很多工作都花费在寻找求解方案以及对其求精上。通常，只有在真正编写程序求解一个问题的过程中才会对问题本身理解透彻。

本书适合于那些从未有过编程经验但愿意努力学习程序设计技术的初学者，它能帮助读者理解使用 C++ 语言进行程序设计的基本原理并获得实践技巧。本书的目标是使你获得足够多的知识和经验，以便能使用最新、最好的技术进行简单有用的编程工作。达到这一目标需要多长时间呢？作为大学一年级课程的一部分，你可以在一个学期内完成这本书的学习（假定你有另外四门中等难度的课程）。如果你是自学的话，不要期望能花费更少的时间完成学习（一般来说，每周 15 个小时，14 周是合适的学时安排）。

三个月可能看起来是一段很长的时间，但要学习的内容很多。写第一个简单程序之前，就要花费大约一个小时。而且，所有学习过程都是渐进的：每一章都会介绍一些新的有用的概念，并通过真实应用中的例子来阐述这些概念。随着学习进程的推进，你通过程序代码表达思想的能力——让计算机按你的期望工作的能力，会逐渐稳步地提高。我绝不会说："先学习一个月的理论知识，然后看看你是否能使用这些理论吧。"

为什么要学习程序设计呢？因为我们的文明是建立在软件之上的。如果不理解软件，那么你将退化到只能相信"魔术"的境地，并且将被排除在很多最为有趣、最具经济效益和社会效益的领域之外。当我谈论程序设计时，我所想到的是整个计算机程序家族，从带有 GUI（图形用户界面）的个人计算机程序，到工程计算和嵌入式系统控制程序（如数码相机、汽车和手机中的程序），以及文字处理程序等，在很多日常应用和商业应用中都能看到这些程序。程序设计与数学有些相似，认真去做的话，会是一种非常有用的智力训练，可以提高我们的思考能力。然而，由于计算机能做出反馈，程序设计不像大多数数学形式那么抽象，因而对多数人来说更易接受。可以说，程序设计是一条能够打开你的眼界，将世界变得更美好的途径。最后，程序设计可以是非常有趣的。

为什么学习 C++ 这门程序设计语言呢？学习程序设计是不可能不借助一门程序设计语言的，而 C++ 直接支持现实世界中的软件所使用的那些关键概念和技术。C++ 是使用最为广泛的程序设计语言之一，其应用领域几乎没有局限。从大洋深处到火星表面，到处都能发现 C++ 程序的身影。C++ 是由一个开放的国际标准组织全面考量、精心设计的。在任何一种计算机平台上都能找到高质量的、免费的 C++ 实现。而且，用 C++ 所学到的程序设计思想，大多数可直接用于其他程序设计语言，如 C、C#、Fortran 以及 Java。最后一个原因，我喜欢 C++ 适合编写优美、高效的代码这一特点。

本书不是初学程序设计的最简单入门教材，我写此书的用意也不在此。我为本书设定的目标是——这是一本能让你学到基本的实用编程技术的最简单书籍。这是一个非常雄心勃勃的目标，因为很多现代软件所依赖的技术，不过才出现短短几年时间而已。

我的基本假设是：你希望编写供他人使用的程序，并愿意认真负责地以较高质量完成这个工作，也就是说，假定你希望达到专业水准。因此，我为本书选择的主题覆盖了开始学习实用编程技术所需要的内容，而不只是那些容易讲授和容易学习的内容。如果某种技术是你做好基本编程工作所需要的，那么本书就会介绍它，同时展示用以支持这种技术的编程思想和语言工具，并提供相应的练习，期望你通过做这些练习来熟悉这种技术。但如果你只想了解“玩具程序”，那么你能学到的将远比我所提供的少得多。另一方面，我不会用一些实用性很低的内容来浪费你的时间，本书介绍的内容都是你在实践中几乎肯定会用到的。

如果你只是希望直接使用别人编写的程序，而不想了解其内部原理，也不想亲自向代码中加入重要的内容，那么本书不适合你，采用另一本书或另一种程序设计语言会更好些。如果这大概就是你对程序设计的看法，那么请同时考虑一下你从何得来的这种观点，它真的满足你的需求吗？人们常常低估程序设计的复杂程度和它的重要性。我不愿看到，你不喜欢程序设计是因为你的需求与我所描述的软件世界之间不匹配而造成的。信息技术世界中有很多地方是不要求程序设计知识的。本书面向的是那些确实希望编写和理解复杂计算机程序的人。

考虑到本书的结构和注重实践的特点，它也可以作为学习程序设计的第二本书，适合那些已经了解一点 C++ 的人，以及那些会用其他语言编程而现在想学习 C++ 的人。如果你属于其中一类，我不好估计你学习这本书要花费多长时间。但我可以给你的建议是，多做练习。因为你在学习中常见的一个问题是习惯用熟悉的、旧的方式编写程序，而不是在适当的地方采用新技术，多做练习会帮助你克服这个问题。如果你曾经按某种更为传统的方式学习过 C++，那么在进行到第 7 章之前，你会发现一些令你惊奇的、有用的内容。除非你的名字是 Stroustrup，否则你会发现我在本书中所讨论的内容不是“你父辈的 C++”。

学习程序设计要靠编程实践。在这一点上，程序设计与其他需要实践学习的技艺是相似的。你不可能仅仅通过读书就学会游泳、演奏乐器或者开车，必须进行实践。同样，你也不可能不读写大量代码就学会程序设计。本书给出了大量代码实例，都配有说明文字和图表。你需要通过读这些代码来理解程序设计的思想、概念和原理，并掌握用来表达这些思想、概念和原理的程序设计语言的特性。但有一点很重要，仅仅读代码是学不会编程实践技巧的。为此，你必须进行编程练习，通过编程工具熟悉编写、编译和运行程序。你需要亲身体验编程中会出现的错误，学习如何修改它们。总之，在学习程序设计的过程中，编写代码的练习是不可替代的。而且，这也是乐趣所在！

另一方面，程序设计远非遵循一些语法规则和阅读手册那么简单。本书的重点不在于 C++ 的语法，而在于理解基础思想、原理和技术，这是一名好程序员所必备的。只有设计良好的代码才有机会成为一个正确、可靠和易维护的系统的一部分。而且，“基础”意味着延续性：当现在的程序设计语言和工具演变甚至被取代后，这些基础知识仍会保持其重要性。

那么计算机科学、软件工程、信息技术等又如何呢？它们都属于程序设计范畴吗？当然不是！但程序设计是一门基础性的学科，是所有计算机相关领域的基础，在计算机科学领域占有重要的地位。本书对算法、数据结构、用户接口、数据处理和软件工程等领域的重要概念和技术进行了简要介绍，但本书不能替代对这些领域的全面、均衡的学习。

代码可以很有用，同样可以很优美。本书会帮你了解这一点，同时理解优美的代码意味

着什么，并帮你掌握构造优美代码的原理和实践技巧。祝你学习程序设计顺利！

致学生

到目前为止，我在德州农工大学已经用本书教过几千名大一新生，其中 60% 曾经有过编程经历，而剩余 40% 从未见过哪怕一行代码。大多数学生的学习是成功的，所以你也可以成功。

你不一定是在某门课程中学习本书，本书也广泛用于自学。然而，不管你学习本书是作为课程的一部分还是自学，都要尽量与他人协作。程序设计有一个不好的名声——它是一种个人活动，这是不公正的。大多数人在作为一个有共同目标的团体的一份子时，工作效果更好，学习得更快。与朋友一起学习和讨论问题不是“作弊”，而是取得进步最有效同时也是最快乐的途径。如果没有特殊情况的话，与朋友一起工作会促使你表达出自己的思想，这正是测试你对问题理解和确认你的记忆的最有效方法。你没有必要独自解决所有编程语言和编程环境上的难题。但是，请不要自欺欺人——不去完成那些简单练习和大量的习题（即使没有老师督促你，你也不应这样做）。记住：程序设计（尤其）是一种实践技能，需要通过实践来掌握。如果你不编写代码（完成每章的若干习题），那么阅读本书就纯粹是一种无意义的理论学习。

大多数学生，特别是那些爱思考的好学生，有时会对自己努力工作是否值得产生疑问。当你产生这样的疑问时，休息一会儿，重新读一下前言，读一下第 1 章和第 22 章。在那里，我试图阐述我在程序设计中发现了哪些令人兴奋的东西，以及为什么我认为程序设计是能为世界带来积极贡献的重要工具。如果你对我的教学哲学和一般方法有疑问，请阅读引言。

你可能会对本书的厚度感到担心。本书如此之厚的一部分原因是，我宁愿反复重复一些解释说明或增加一些实例，而不是让你自己到处找这些内容，这应该令你安心。另外一个主要原因是，本书的后半部分是一些参考资料和补充资料，供你想要深入了解程序设计的某个特定领域（如嵌入式系统程序设计、文本分析或数值计算）时查阅。

还有，学习中请耐心些。学习任何一种重要的、有价值的新技能都要花费一些时间，而这是值得的。

致教师

本书不是传统的计算机科学导论书籍，而是一本关于如何构造能实际工作的软件的书。因此本书省略了很多计算机科学系学生按惯例要学习的内容（图灵完全、状态机、离散数学、乔姆斯基文法等）。硬件相关的内容也省略了，因为我假定学生从幼儿园开始就已经通过不同途径使用过计算机了。本书也不准备涉及一些计算机科学领域最重要的主题。本书是关于程序设计的（或者更一般地说，是关于如何开发软件的），因此关注的是少量主题的更深入的细节，而不是像传统计算机课程那样讨论很多主题。本书只试图做好一件事，而且计算机科学也不是一门课程可以囊括的。如果本书被计算机科学、计算机工程、电子工程（我们最早的很多学生都是电子工程专业的）、信息科学或者其他相关专业所采用，我希望这门课程能和其他一些课程一起进行，共同形成对计算机科学的完整介绍。

请阅读引言，那里有对我的教学哲学、一般方法等的介绍。请在教学过程中尝试将这些观点传达给你的学生。

ISO 标准 C++

C++ 由一个 ISO 标准定义。第一个 ISO C++ 标准于 1998 年获得批准，所以那个版本的 C++ 被称为 C++98。写本书第 1 版时，我正从事 C++11 的设计工作。最令人沮丧的是，当时我还不能使用一些新语言特性（如统一初始化、范围 for 循环、move 语义、lambda 表达式、concept 等）来简化原理和技术的展示。不过，由于设计该书时考虑到了 C++11，所以很容易在合适的地方添加这些特性。在写作本版时，C++ 标准是 2011 年批准的 C++11，2014 ISO 标准 C++14 中的一些特性正在进入主流的 C++ 实现中。本书中使用的语言标准是 C++11，并涉及少量的 C++14 特性。例如，如果你的编译器不能编译下面的代码：

```
vector<int> v1;
vector<int> v2 {v1};    // C++14 风格的拷贝构造
```

可用如下代码替代：

```
vector<int> v1;
vector<int> v2 = v1;    // C++98 风格的拷贝构造
```

若你的编译器不支持 C++11，请换一个新的编译器。好的、现代的 C++ 编译器可从多处下载，见 www.stroustrup.com/compilers.html。使用较早且缺少支持的语言版本会引入不必要的困难。

资源

本书支持网站的网址为 www.stroustrup.com/Programming，其中包含了各种使用本书讲授和学习程序设计所需的辅助资料。这些资料可能会随着时间的推移不断改进，但对于初学者，现在可以找到这些资料：

- 基于本书的讲义幻灯片；
- 一本教师指南；
- 本书中使用的库的头文件和实现；
- 本书中实例的代码；
- 某些习题的解答；
- 可能会有用处的一些链接；
- 勘误表。

欢迎随时提出对这些资料的改进意见。

致谢

我要特别感谢已故的同事和联合导师 Lawrence “Pete” Petersen，很久以前，在我还未感受到教授初学者的惬意时，是他鼓励我承担这项工作，并向我传授了很多能令课程成功的教学经验。没有他，这门课程的首次尝试就会失败。他参与了这门课程最初的建设，本书就是为这门课程所著。他还和我一起反复讲授这门课程，汲取经验，不断改进课程和本书。在本书中我使用的“我们”这个字眼，最初的意思就是指“我和 Pete”。

我要感谢那些直接或间接帮助过我撰写本书的学生、助教和德州农工大学讲授 ENGR 112、113 及 CSCE 121 课程的教师，以及 Walter Daugherity、Hyunyoung Lee、Teresa Leyk、Ronnie Ward、Jennifer Welch，他们也讲授过这门课程。还要感谢 Damian Dechev、Tracy

Hammond、Arne Tolstrup Madsen、Gabriel Dos Reis、Nicholas Stroustrup、J. C. van Winkel、Greg Versoonder、Ronnie Ward 和 Leor Zolman 对本书初稿提出的建设性意见。感谢 Mogens Hansen 为我解释引擎控制软件。感谢 Al Aho、Stephen Edwards、Brian Kernighan 和 Daisy Nguyen 帮助我在夏天躲开那些分心的事，完成本书。

感谢 Art Werschulz，他在纽约福特汉姆大学的课程中使用了本书第 1 版，并据此提出了很多建设性的意见。还要感谢 Nick Maclaren，他在剑桥大学使用了本书的第 1 版，并对本书的习题提出了非常详尽的建议。他的学生在知识背景和专业需求上与德州农工大学大一学生有巨大的差异。

感谢 Addison-Wesley 公司为我安排的审阅者 Richard Enbody、David Gustafson、Ron McCarty 和 K. Narayanaswamy，他们基于自身讲授 C++ 课程或大学计算机科学导论课程的经验，对本书提出了宝贵的意见。还要感谢我的编辑 Peter Gordon 为本书提出的很多有价值的意见以及极大的耐心。我非常感谢 Addison-Wesley 公司的制作团队，他们为本书的高质量出版做出了很多贡献，他们是：Linda Begley（校对），Kim Arney（排版），Rob Mauhar（插图），Julie Nahil（制作编辑），Barbara Wood（文字编辑）。

感谢本书第 1 版的译者，他们发现并帮助澄清了很多问题。特别是，Loïc Joly 和 Michel Michaud 在法语翻译版中做了全面的技术检查，修改了很多问题。

我还要感谢 Brian Kernighan 和 Doug McIlroy 为撰写程序设计类书籍定下了一个非常高的标杆，以及 Dennis Ritchie 和 Kristen Nygaard 为实用编程语言设计提供的非常有价值的经验。

当实际地形与地图不符时，相信实际地形。

——瑞士军队谚语

讲授和学习本书的方法

我们是如何帮助你学习的？又是如何安排学习进程的？我们的做法是，尽力为你提供编写高效的实用程序所需的最基本的概念、技术和工具，包括程序组织、调试和测试、类设计、计算、函数和算法设计、绘图方法（仅介绍二维图形）、图形用户界面（GUI）、文本处理、正则表达式匹配、文件和流输入输出（I/O）、内存管理、科学 / 数值 / 工程计算、设计和编程思想、C++ 标准库、软件开发策略、C 语言程序设计技术。认真完成这些内容的学习，我们会学到如下程序设计技术：过程式程序设计（C 语言程序设计风格）、数据抽象、面向对象程序设计和泛型程序设计。本书的主题是程序设计，也就是表达代码意图所需的思想、技术和工具。C++ 语言是我们的主要工具，因此我们比较详细地描述了很多 C++ 语言的特性。但请记住，C++ 只是一种工具，而不是本书的主题。本书是“用 C++ 语言进行程序设计”，而不是“C++ 和一点程序设计理论”。

我们介绍的每个主题都至少出于两个目的：提出一种技术、概念或原理，介绍一个实用的语言特性或库特性。例如，我们用一个二维图形绘制系统的接口展示如何使用类和继承。这使我们节省了篇幅（也节省了你的时间），并且还强调了程序设计不只是简单地将代码拼装起来以尽快地得到一个结果。C++ 标准库是这种“双重作用”例子的主要来源，其中很多主题甚至具有三重作用。例如，我们会介绍标准库中的向量类 vector，用它来展示一些广泛使用的设计技术，并展示很多用来实现 vector 的程序设计技术。我们的一个目标是向你展示一些主要的标准库功能是如何实现的，以及它们如何与硬件相配合。我们坚持认为一个工匠必须了解他的工具，而不是仅仅把工具当作“有魔力的东西”。

对于一个程序员来说，总是会对某些主题比对其他主题更感兴趣。但是，我们建议你不要预先判断你需要什么（你怎么知道你将来会需要什么呢？），至少每一章都要浏览一下。如果你学习本书是作为一门课程的一部分，你的老师会指导你如何选择学习内容。

我们的教学方法可以描述为“深度优先”，同时也是“具体优先”和“基于概念”。首先，我们快速地（好吧，是相对快速地，从第 1 章到第 11 章）将一些编写小的实用程序所需的技巧提供给你。在这期间，我们还简明扼要地提出很多工具和技术。我们着重于简单具体的代码实例，因为相对于抽象概念，人们能更快领会具体实例，这就是多数人的学习方法。在最初阶段，你不应期望理解每个小的细节。特别是，你会发现对刚刚还工作得好好的程序稍加改动，便会呈现出“神秘”的效果。尽管如此，你还是要尝试一下！还有，请完成我们提供的简单练习和习题。请记住，在学习初期你只是没有掌握足够的概念和技巧来准确判断什么是简单的，什么是复杂的。请等待一些惊奇的事情发生，并从中学习吧。

我们会快速通过这样一个初始阶段——我们想尽可能快地带你进入编写有趣程序的阶段。有些人可能会质疑，“我们的进展应该慢些、谨慎些，我们应该先学会走，再学跑！”但

是你见过小孩学习走路吗？实际上小孩在学会平稳地慢慢走路之前就开始尝试跑了。与之相似，你可以先勇猛向前，偶尔摔一跤，从中获得编程的感觉，然后再慢下来，获得必要的精确控制能力和准确的理解。你必须在学会走之前就开始跑！

⚠ 你不要投入大量精力试图学习一些语言或技术细节的所有相关内容。例如，你可以熟记所有 C++ 的内置类型及其使用规则。你当然可以这么做，而且这么做会使你觉得自己很博学。但是，这不会使你成为一名程序员。如果你学习中略过一些细节，将来可能偶尔会因为缺少相关知识而被“灼伤”，但这是获取编写好程序所需的完整知识结构的最快途径。注意，我们的这种方法本质上就是小孩学习其母语的方法，也是教授外语的最有效方法。有时你不可避免地被难题困住，我们鼓励你向授课老师、朋友、同事、指导教师等寻求帮助。请放心，在前面这些章节中，所有内容本质上都不困难。但是，很多内容是你所不熟悉的，因此最初可能会感觉有点难。

随后，我们介绍一些入门技巧来拓宽你的知识。我们通过实例和习题来强化你的理解，为你提供一个程序设计的概念基础。

我们非常强调思想和原理。思想能指导你求解实际问题——可以帮助你知道在什么情况下问题求解方案是好的、合理的。你还应该理解这些思想背后的原理，从而理解为什么要接受这些思想，为什么遵循这些思想会对你和使用你的代码的用户有帮助。没有人会满意“因为事情就是如此”这样的解释。更为重要的是，如果真正理解了思想和原理，你就能将自己已知的知识推广到新的情况；就能用新的方法将思想和工具结合来解决新的问题。知其所以然是学会程序设计技巧所必需的。相反，仅仅不求甚解地记住大量规则和语言特性有很大局限，是错误之源，是在浪费时间。我们认为你的时间很珍贵，尽量不要浪费它。

我们把很多 C++ 语言层面的技术细节放在了附录和手册中，你可以随时按需查找。我们假定你有能力查找到需要的信息，你可以借助目录来查找信息。不要忘了编译器和互联网的在线功能。但要记住，要对所有互联网资源保持足够的怀疑，直至你有足够的理由相信它们。因为很多看起来很权威的网站实际上是由程序设计新手或者想要出售什么东西的人建立的。而另外一些网站，其内容都是过时的。我们在支持网站 www.stroustrup.com/Programming 上列出了一些有用的网站链接和信息。

请不要过于急切地期盼“实际的”例子。我们理想的实例都是能直接说明一种语言特性、一个概念或者一种技术的简短代码。很多现实世界中的实例比我们给出的实例要凌乱很多，而且所能展示的知识也不比我们的实例更多。包含数十万行代码的成功商业程序中所采用的技术，我们用几个 50 行规模的程序就能展示出来。理解现实世界程序的最快途径是好好研究一些基础的小程序。

另一方面，我们不会用“聪明可爱的风格”来阐述我们的观点。我们假定你的目标是编写供他人使用的实用程序，因此书中给出的实例要么是用来说明语言特性，要么是从实际应用中提取出来的。我们的叙述风格都是用专业人员对（将来的）专业人员的那种口气。

一般方法

☛ 本书的内容组织适合从头到尾一章一章地阅读，当然，你也常常要回过头来对某些内容读上第二遍、第三遍。实际上，这是一种明智的方法，因为当遇到还看不出什么门道的地方时，你通常会快速掠过。对于这种情况，你最终还是会再次回到这个地方。然而，这么做要适度，因为除了交叉引用之外，对本书其他部分，你随便翻开一页，就从那里开始学习，

并希望成功，是不可能的。本书每一节、每一章的内容安排，都假定你已经理解了之前的内容。

本书的每一章都是一个合理的自包含单元，这意味着应一口气读完（当然这只是理论上，实际上由于学生紧密的学习计划，不总是可行）。这是将内容划分为章的主要标准。其他标准包括：从简单练习和习题的角度，每章是一个合适的单元；每一章提出一些特定的概念、思想或技术。这种标准的多样性使得少数章过长，所以不要教条地遵循“一口气读完”的准则。特别是当你已经考虑了思考题，做了简单练习，并做了一些习题时，你通常会发现你需要回过头去重读一些小节和几天前读过的内容。

“它回答了我想到的所有问题”是对一本教材常见的称赞，这对细节技术问题是很理想的，而早期的读者也发现本书有这样的特性。但是，这不是全部的理想，我们希望提出初学者可能想不到的问题。我们的目标是，回答那些你在编写供他人使用的高质量软件时需要考虑的问题。学习回答好的（通常也是困难的）问题是学习如何像一个程序员那样思考所必需的。只回答那些简单的、浅显的问题会使你感觉良好，但无助于你成长为一名程序员。

我们努力尊重你的智力，珍惜你的时间。在本书中，我们以专业性而不是精明伶俐为目标，宁可有节制地表达一个观点而不大肆渲染它。我们尽力不夸大一种程序设计技术或一个语言特性的重要性，但请不要因此低估“这通常是有用的”这种简单陈述的重要程度。如果我们平静地强调某些内容是重要的，意思是你如果不掌握它，或早或晚都会因此而浪费时间。

我们不会伪称本书中的思想和工具是完美的。实际上没有任何一种工具、库、语言或者技术能够解决程序员所面临的所有难题，至多能帮助你开发、表达你的问题求解方案而已。我们尽量避免“无害的谎言”，也就是说，我们会尽力避免过于简单的解释，虽然这些解释清晰且易理解，但在实际编程和问题求解时却容易弄错。另一方面，本书不是一本参考手册，如果需要 C++ 详细完整的描述，请参考 Bjarne Stroustrup 的《The C++ Programming Language》第 4 版（Addison-Wesley 出版社，2013 年）和 ISO 的 C++ 标准。

简单练习和习题等

程序设计不仅仅是一种脑力活动，实际动手编写程序是掌握程序设计技巧必不可少的一环。本书提供两个层次的程序设计练习：

- *简单练习*：简单练习是一种非常简单的习题，其目的是帮助学生掌握一些相对死板的实际编程技巧。一个简单练习通常由一系列的单个程序修改练习组成。你应该完成所有简单练习。完成简单练习不需要很强的理解能力、很聪明或者很有创造性。简单练习是本书的基本组成部分，如果你没有完成简单练习，就不能说完成了本书的学习。
- *习题*：有些习题比较简单，有些则很难，但多数习题都是想给学生留下一定的创造和想象空间。如果时间紧张，你可以做少量习题，但题量至少应该能使你弄清楚哪些内容对你来说比较困难，在此基础上应该再多做一些，这是你的成功之道。我们希望本书的习题都是学生能够做出来的，而不是需要超乎常人的智力才能解答的复杂难题。但是，我们还是期望本书习题能给你足够多的挑战，能用光甚至是最好的学生的所有时间。我们不期待你能完成所有习题，但请尽情尝试。

另外，我们建议每个学生都能参与到一个小的项目中去（如果时间允许，能参与更多项

目当然就更好了）。一个项目的目的就是要编写一个完整的有用程序。理想情况下，一个项目由一个多人小组（比如三个人）共同完成。大多数人会发现做项目非常有趣，并在这个过程中学会如何把很多事情组织在一起。

一些人喜欢在读完一章之前就把书扔到一边，开始尝试做一些实例程序；另一些人则喜欢把一章读完后，再开始编码。为了帮助前一种读者，我们用“试一试”板块给出了对于编程实践的一些简单建议。一个“试一试”通常来说就是一个简单练习，而且只着眼于前面刚刚介绍的主题。如果你略过了一个“试一试”而没有去尝试它，那么最好在做这一章的简单练习时做一下这个题目。“试一试”要么是该章简单练习的补充，要么干脆就是其中的一部分。

在每章末尾你都会看到一些思考题，我们设置这些思考题是想为你指出这一章中的重点内容。一种学习思考题的方法是把它们作为习题的补充：习题关注程序设计的实践层面，而思考题则试图帮你强化思想和概念。因此，思考题有点像面试题。

每章最后都有“术语”一节，给出本章中提出的程序设计或 C++ 方面的基本词汇表。如果你希望理解别人关于程序设计的陈述，或者想明确表达出自己的思想，就应该首先弄清术语表中每个术语的含义。

重复是学习的有效手段，我们希望每个重要的知识点都在书中至少出现两次，并通过习题再次强调。

进阶学习

当你完成本书的学习时，是否能成为一名程序设计和 C++ 方面的专家呢？答案当然是否定的！如果做得好的话，程序设计会是一门建立在多种专业技能上的精妙的、深刻的、需要高度技巧的艺术。你不能期望花四个月时间就成为一名程序设计专家，这与其他学科一样：你不能期望花四个月、半年或一年时间就成为一名生物学专家、一名数学家、一名自然语言（如中文、英文或丹麦文）方面的专家，或是一名小提琴演奏家。但如果你认真地学完了这本书，你可以期待也应该期待的是：你已经在程序设计领域有了一个很好的开始，已经可以写相对简单的、有用的程序，能读更复杂的程序，而且已经为进一步的学习打下了良好的理论和实践基础。

学习完这门入门课程后，进一步学习的最好方法是开发一个真正能被别人使用的程序。在完成这个项目之后或者同时（同时可能更好）学习一本专业水平的教材（如 Stroustrup 的《 The C++ Programming Language 》），学习一本与你做的项目相关的更专门的书（比如，你如果在做 GUI 相关项目的话，可选择关于 Qt 的书，如果在做分布式程序的话，可选择关于 ACE 的书），或者学习一本专注于 C++ 某个特定方面的书（如 Koenig 和 Moo 的《 Accelerated C++ 》、Sutter 的《 Exceptional C++ 》或 Gamma 等人的《 Design Patterns 》）。完整的参考书目参见本引言或本书最后的参考文献。

最后，你应该学习另一门程序设计语言。我们认为，如果只懂一门语言，你是不可能成为软件领域的专家的（即使你并不是想做一名程序员）。

本书内容顺序的安排

讲授程序设计有很多方法。很明显，我们不赞同“我学习程序设计的方法就是最好的学习方法”这种流行的看法。为了方便学习，我们较早地提出一些仅仅几年前还是先进技术的

内容。我们的设想是，本书内容的顺序完全由你学习程序设计过程中遇到的问题来决定，随着你对程序设计的理解和实际动手能力的提高，一个主题一个主题地平滑向前推进。本书的叙述顺序更像一部小说，而不是一部字典或者一种层次化的顺序。

一次性地学习所有程序设计原理、技术和语言功能是不可能的。因此，你需要选择其中一个子集作为起点。更一般地，一本教材或一门课程应该通过一系列的主题子集来引导学生。我们认为，选择适当的主题并给出重点是我们的责任。我们不能简单地罗列出所有内容，必须做出取舍；在每个学习阶段，我们选择省略的内容与选择保留的内容至少同样重要。

作为对照，这里列出我们决定不采用的教学方法（仅仅是一个缩略列表），对你可能有用：

- C 优先：用这种方法学习 C++ 完全是浪费学生的时间，学生能用来求解问题的语言功能、技术和库比所需的要少得多，这样的程序设计实践很糟糕。与 C 相比，C++ 能提供更强的类型检查、对新手来说更好的标准库以及用于错误处理的异常机制。
- 自底向上：学生本该学习好的、有效的程序设计技巧，但这种方法分散了学生的注意力。学生在求解问题过程中所能依靠的编程语言和库方面的支持明显不足，这样的编程实践质量很低、毫无用处。
- 如果你介绍某些内容，就必须介绍它的全部：这实际上意味着自底向上方法（一头扎进涉及的每个主题，越陷越深）。这种方法硬塞给初学者很多他们并不感兴趣而且可能很长时间内都用不上的技术细节，令他们厌烦。这样做毫无必要，因为一旦学会了编程，你完全可以自己到手册中查找技术细节。这是手册擅长的方面，如果用来学习基本概念就太可怕了。
- 自顶向下：这种方法对一个主题从基本原理到细节逐步介绍，倾向于把读者的注意力从程序设计的实践层面上转移开，迫使读者一直专注于上层概念，而没有任何机会实际体会这些概念的重要性。这是错误的，例如，如果你没有实际体会到编写程序是那么容易出错，而修正一个错误是那么困难，你就无法体会到正确的软件开发原理。
- 抽象优先：这种方法专注于一般原理，保护学生不受讨厌的现实问题限制条件的困扰，这会导致学生轻视实际问题、语言、工具和硬件限制。通常，这种方法基于“教学用语言”——一种将来不可能实际应用，有意将学生与实际的硬件和系统问题隔绝开的语言。
- 软件工程理论优先：这种方法和抽象优先的方法具有与自顶向下方法一样的缺点：没有具体实例和实践体验，你无法体会到抽象理论的价值和正确的软件开发实践技巧。
- 面向对象先行：面向对象程序设计是一种组织代码和开发工作的很好方法，但并不是唯一有效的方法。特别是，以我们的体会，在类型系统和算法式编程方面打下良好的基础，是学习类和类层次设计的前提条件。本书确实在一开始就使用了用户自定义类型（一些人称之为“对象”），但我们直到第 6 章才展示如何设计一个类，而直到第 17 章才展示了类层次。
- 相信魔法：这种方法只是向初学者展示强有力的工具和技术，而不介绍其下蕴含的技术和特性。这让学生只能去猜这些工具和技术为什么会有这样的表现，使用它们会付出多大代价，以及它们恰当的应用范围，而通常学生会猜错！这会导致学生过分刻板地遵循相似的工作模式，成为进一步学习的障碍。

自然，我们不会断言这些我们没有采用的方法毫无用处。实际上，在介绍一些特定的内容时，我们使用了其中一些方法，学生能体会到这些方法在这些特殊情况下的优点。但是，当学习程序设计是以实用为目标时，我们不把这些方法作为一般的教学方法，而是采用其他方法：主要是具体优先和深度优先方法，并对重点概念和技术加以强调。

程序设计和程序设计语言

我们首先介绍程序设计，把程序设计语言放在第二位。我们介绍的程序设计方法适用于任何通用的程序设计语言。我们的首要目的是帮助你学习一般概念、理论和技术，但是这些内容不能孤立地学习。例如，不同程序设计语言在语法细节、编程思想的表达以及工具等方面各不相同。但对于编写无错代码的很多基本技术，如编写逻辑简单的代码（第 5 章和第 6 章），构造不变式（9.4.3 节），以及接口和实现细节分离（9.7 节和 19.1 ~ 19.2 节）等，不同程序设计语言则差别很小。

程序设计技术的学习必须借助于一门程序设计语言，代码设计、组织和调试等技巧是不可能从抽象理论中学到的。你必须用某种程序设计语言编写代码，从中获取实践经验。这意味着你必须学习一门程序设计语言的基本知识。这里说“基本知识”，是因为花几个星期就能掌握一门主流实用编程语言全部内容的日子已经一去不复返了。本书中 C++ 语言相关的内容只是我们选出的它的一个子集，是与编写高质量代码关系最紧密的那部分内容。而且，我们所介绍的 C++ 特性都是你肯定会用到的，因为这些特性要么是出于逻辑完整性的要求，要么是 C++ 社区中最常见的。

可移植性

编写运行于多种平台的 C++ 程序是很常见的情况。一些重要的 C++ 应用甚至运行于我们闻所未闻的平台！我们认为可移植性和对多种平台架构 / 操作系统的利用是非常重要的特性。本质上，本书的每个例子都不仅是 ISO 标准 C++ 程序，还是可移植的。除非特别指出，本书的代码都能运行于任何一种 C++ 实现，并且确实已经在多种计算机平台和操作系统上测试通过了。

不同系统编译、链接和运行 C++ 程序的细节各不相同，如果每当提及一个实现问题时就介绍所有系统和所有编译器的细节，是非常单调乏味的。我们在附录 B 中给出了 Windows 平台 Visual Studio 和 Microsoft C++ 入门的大部分基本知识。

如果你在使用任何一种流行的但相对复杂的 IDE（集成开发环境，Integrated Development Environment）时遇到了困难，我们建议你尝试命令行工作方式，它极其简单。例如，下面给出的是在 Unix 或 Linux 平台用 GNU C++ 编译器编译、链接和运行一个包含两个源文件 my_file1.cpp 和 my_file2.cpp 的简单程序所需的全部命令：

```
c++ -o my_program my_file1.cpp my_file2.cpp
./my_program
```

是的，这真的就是全部。

提示标记

为了方便读者回顾本书，以及帮读者发现第一次阅读时遗漏的关键内容，我们在页边空白处放置三种“提示标记”：