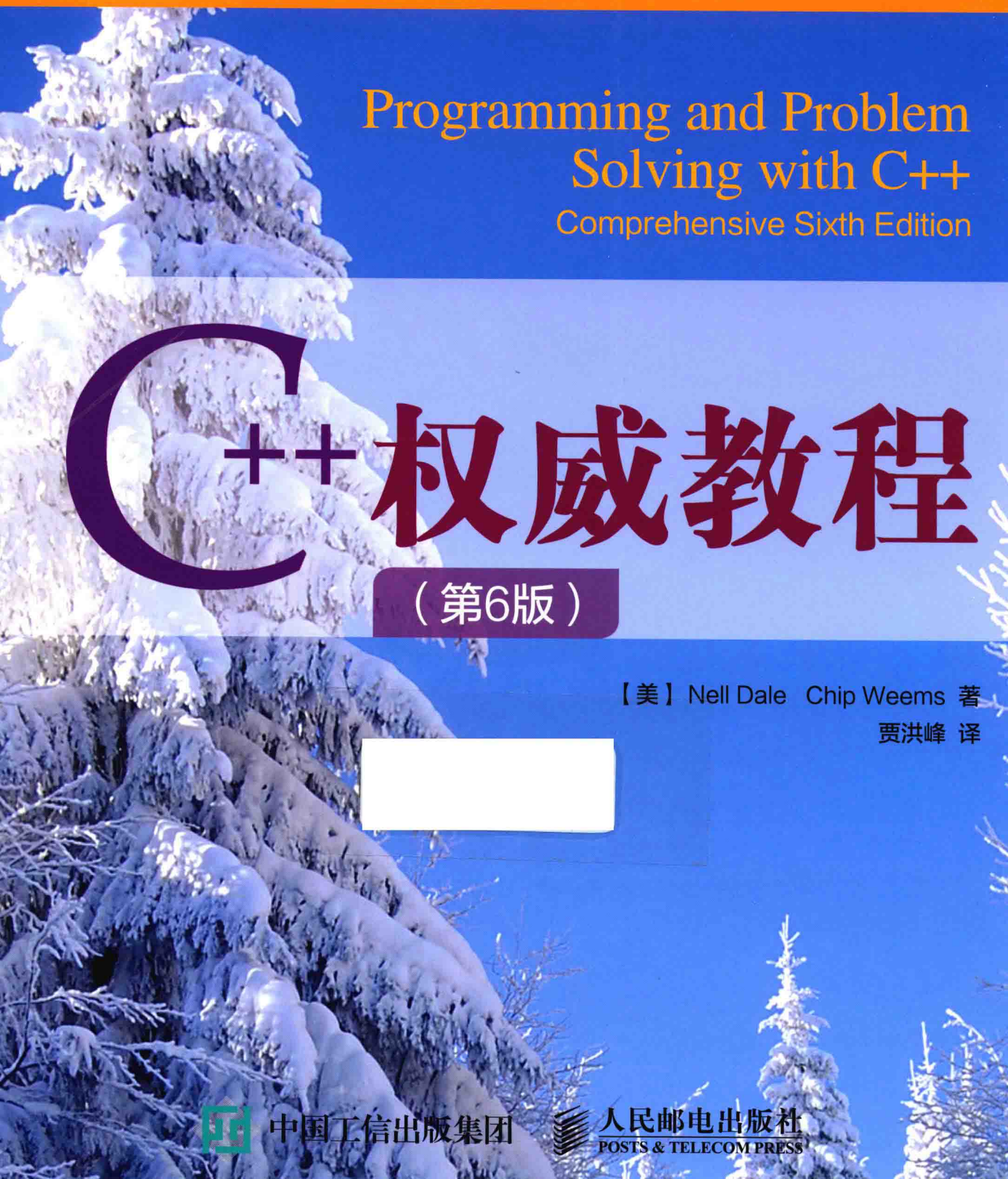
TURING 图灵程序设计丛书



Programming and Problem Solving with C++

Comprehensive Sixth Edition

C++权威教程

（第6版）

【美】Nell Dale　Chip Weems 著

贾洪峰 译

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Programming and Problem Solving with C++

Comprehensive Sixth Edition

C++权威教程

（第6版）

【美】Nell Dale　Chip Weems 著

贾洪峰 译

人民邮电出版社

北京

图书在版编目（CIP）数据

C++权威教程 : 第6版 / (美) 戴尔 (Dale,N.) , (美) 威姆斯 (Weems,C.) 著 ; 贾洪峰译. -- 北京 : 人民邮电出版社, 2016.1
(图灵程序设计丛书)
ISBN 978-7-115-40792-4

Ⅰ. ①C… Ⅱ. ①戴… ②威… ③贾… Ⅲ. ①C语言－程序设计－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第253710号

版 权 声 明

内 容 提 要

本书是一本全面的 C++ 教程，介绍 C++ 编程知识及问题解决方法，重在让读者更轻松地理解计算机科学基本概念，养成良好编程习惯。主要内容涵盖 C++ 程序设计的方方面面，包括计算机软硬件基础知识、数值类型、输入与程序设计方法、控制结构、数据结构等。书中给出了大量案例，并用相关度极高的练习强化读者对关键概念的理解。

本书可作为大中专院校计算机及相关专业高级程序设计语言的教材和辅导用书，也可供相关技术人员参考。

◆ 著　[美] Nell Dale Chip Weems
译　贾洪峰
责任编辑　岳新欣
执行编辑　张　曼
责任印制　杨林杰
◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路11号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷
◆ 开本：880×1230　1/16
印张：45.75
字数：1546千字　2016年1月第1版
印数：1－3 000册　2016年1月河北第1次印刷
著作权合同登记号　图字：01-2014-4183号

定价：149.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

前　言

第 6 版简介

本书英文版前 5 版一直位于美国最畅销的计算机科学教科书之列。这 5 个版本和相应的 Java、Ada 和 Pascal 版本被广泛采用，作为 ACM/IEEE CS1/C101 推荐课程、计算机科学先修课程（Advanced Placement）考试的模范教科书。

本书一版再版，始终未变的是对学生的承诺。一如既往，我们努力让所有学生都能更轻松地理解计算机科学中有些令人费解的概念。这一版同样反映了我们的观点：一本教科书就应当像领路人一样，一路披荆斩棘，引导它的读者穿越初看起来似乎难以“通行”的领域。

第 6 版中的改动

本书这一版继续扩展知识面，希望可供两学期的课程使用，以便为学生提供更大的价值。

在这一版中，我们用更为现代的 `static_cast<type>`代替了原来用函数符号表示的强制类型变换。每一节的最后都提供了“小测”练习，并在网上提供了对应的答案。我们还对指针内容进行了调整：第 10 章介绍指针和引用类型，第 11 章增加了结合使用指针与数组的内容，还有内存的动态分配与解除分配。第 11 章还新增加了一节，讨论 C 字符串以及它们与 C++ `string` 类的组合应用。由于许多用户还在使用 C++03，所以这一版没有采用 C++11 标准的特性。

教学方法

概念介绍仍然采用循序渐进的方法，没有大量堆叠学生们可能无法消化吸收的细节和各种相关候选方式。例如，许多教科书都是在一开篇就介绍数值类型。而本书首先在第 2 章介绍 `char` 和 `string` 类型，使学生能够专心学习如何构建执行基本输出功能的 C++程序，迅速写出能正常运行的示例程序。在确保读者熟悉了这些基础知识之后，我们才在第 3 章增加了数值类型。这样就将语句、程序语法方面的概念性难题与类型的复杂性隔离开来（类型的复杂性涉及数值精度、有无符号、类型变换和优先级等），让更多的学生在课程伊始就能品尝成功的喜悦。

Elliot Soloway 曾对初学程序设计的人员进行过研究，根据其研究结果，我们在最初介绍选择结构时仍然仅使用 If 语句，而对于循环结构则仅使用 While 语句。但是，因为许多老师喜欢将所有选择控制结构放在一起、将所有循环控制结构放在一起讲授，所以在讲完选择和循环之后，我们马上用一章来介绍其他控制结构。

同样，学生们发现 `struct` 类型要比数组类型更容易理解（`struct` 的命名字段可以混合不同类型，而数组的索引经常需要通过算术表达式进行访问）。因此，第 10 章介绍了 `struct` 和 `union` 类型，数组则延后到第 11 章介绍。第 10 章还给出了指针和引用类型的一些简单应用，将指针与数组的组合应用、内存的动态分配延后到第 11 章，使学生能够在处理堆中数据之前熟悉间接访问。

我们意识到，许多学生都是通过模仿已有方案来学习程序设计的，所以本书每一章都提供了大量短小的示例程序。这些程序用于阐明相关概念的上下文要比代码段中更为完整，它们会在介绍新概念之后立即给出。根据我们对各章的结构设定，在讨论一个概念之后，本书会快速转向讨论其实际应用，之后才会讨论其相关概念。各章提供了一系列的具体示例，可以在通向主要案例研究的路途中充当路标。

C++与面向对象的程序设计

一些老师反对 C 和 C++，理由是它们太容易编写出隐晦难懂的程序。但根据我们的经验，如果能够正确规范对各种语言功能的应用，上述观点并不成立。C 家族的确允许采用简洁、紧凑的程序设计风格，这是事实，但我们不能简单地为其打上“好”或“坏”的标签。几乎所有程序设计语言都可以采用一种简洁、精巧到令人难以理解的风格来编写程序。尽管以这种方式使用 C 家族的情况确实多于其他语言，但我们发现，如果能够认真贯彻软件工程学中的相关规则，在设计程序时追求简单明了、遵守规则、不使用晦涩难懂的语言特性，学生们仍然可以学会用 C++编写清晰易懂的代码。

必须强调一点：尽管我们以 C++作为讲授计算机科学概念的工具，但本书并不是语言手册；它无意于涵盖 C++语言的所有特性，而是将各种语言构造与相关理论一起进行介绍，以此来演示其实际应用。

面向对象程序设计（OOP）这一主题应当在什么时候开始介绍呢？关于这一问题人们有不同观点。一些老师主张在一开始就完全沉浸于 OOP，而另外一些老师则赞成一种更加另类的方法，即功能分解和面向对象设计都作为设计工具出现（本书就是为后一类老师准备的）。本书的章节结构反映了一种逐渐过渡到 OOP 的方法。类和面向对象的术语在第 12 章给出，而继承、多态和面向对象设计则推后到第 15 章。中间各章为学生提供了充足的时间和体验来熟悉类的应用。

纲要

第 1 章旨在为让学生更好地理解本书主题打好基础。本章介绍了软硬件基础知识，提出了计算机行为准则问题，在“软件维护案例研究”中第一次提及 C++语法，在“实战案例研究”中引入和补充了解决实际问题的技巧。

第 2 章并没有立时阐述 C++中的大量数值类型以免击垮学生的斗志，而只是集中介绍了两种类型：`char` 和 `string`（对于 `string` 类型，使用了标准库中提供的 ISO/ANSI 字符串类）。由于学生需要记住的数据类型较少，所以可以专注于程序的整体结构，提早开始创建和运行简单程序。随后的第 3 章讨论了 C++数值类型，然后转而介绍算术表达式、函数调用和输出。很多书都是一次性详细介绍所有 C++数据类型和所有 C++运算符，而本书的这两章则与之不同，仅专门介绍了 `int`、`float`、`char` 和 `string` 类型，以及一些基本的算术运算符。其他数据类型的讨论一直推迟到第 10 章。

输入与程序设计方法是第 4 章的主题。这一章解释了面向对象设计（OOD）与功能分解之间的区别，随后更深入地介绍了功能分解法。这样，学生们可以很快认识到：共有两种（而不是一种）被广泛使用的设计方法，每一种设计方法都有自己的专用领域。第 4 章还介绍了文件 I/O。提前介绍了文件之后，我们就可以安排一些需要使用示例数据文件的程序设计任务了。

第 5 章首先介绍控制流与分支的概念，然后介绍关系与布尔运算。采用 If-Then 和 If-Then-Else 结构的选择过程显示了语句的物理顺序与逻辑顺序之间的区别。我们还进一步引出了嵌套控制结构的概念。第 5 章通过较大篇幅的“测试与调试”一节介绍了前置条件和后置条件，从而拓展了对模块化设计的讨论。这一节中的算法演练和代码演练可以帮助避免错误，而执行跟踪用于发现代码中的错误。另外，这一节还全面讨论了数据验证与测试策略。

第 6 章专门采用 While 语句介绍循环控制策略和循环操作。这一章并没有介绍多种句法结构，而是仅使用

While 语句来讲授循环的概念。接着，第 7 章介绍了 C++中的其他控制结构（Switch、Do-While 和 For），还有 Break 与 Continue 语句。这些结构就和“餐后甜点”一样，有了当然是锦上添花，但算不上不可或缺。其他 C++运算符的讨论也移到了这一章，因为这些运算符也是虽然有用，但并非必不可少。

到了第 8 章，学生们应该已经非常熟悉问题的模块化分解和库函数的使用了，也更容易接受自己编写函数的想法了。第 8 章专门讨论参数的按值传递，介绍函数调用中的控制流，还有实参与形参、局部变量和接口设计。关于接口设计的介绍中包括接口文档中的前置条件和后置条件、控制抽象、封装、物理上与概念上的实现隐藏。第 9 章拓展了这一讨论，增加了具有返回值的函数、引用参数、作用域与生存期、存根（stub）与驱动测试程序，还有包括副作用在内的更多接口设计内容。

第 10 章首先由本书第一部分的控制结构导向过渡到第二部分的数据结构导向。我们回顾了内置的简单数据类型，包括每种类型表示的取值范围、允许对这些值进行的运算等。我们还介绍了枚举类型、struct 和 union。此外，第 10 章介绍了简单数据类型与结构化数据类型。这一版中还增加了指针与引用类型。

在第 11 章，将数组作为同构的数据结构进行了介绍，其各个组成项按位置（而非名称）进行访问。这一章深入探讨了一维数组，包括 struct 的数组。C 字符串、二维数组、三维数组和多维数组的相关材料使数组类型的讨论变得圆满。这一章最后还讨论了指针与数组的组合使用以及数据的动态分配。

第 12 章以抽象数据类型的概念引入对类结构的探讨，介绍了面向对象的术语，着重介绍了类与对象之间的区别，强调了优秀的类设计原则。这一章还给出了规范说明文件与实现文件的使用，作为隐藏信息的一种形式。尽管在一个简单示例中讨论和演示了可变性，但本章中的所有类都有意设定为不可变类：这里需要学生们消化吸收的新软件设计概念已经够多了，所以就不再添乱了。

但接下来的一章反复演示了可变类。第 13 章将第 11 章和第 12 章的内容整合在一起：将列表定义为 ADT，然后用一个包含数组的类来实现它。由于我们已经介绍了类和数组，所以在刚开始讨论时，读者应该就能清楚地区分数组和列表。数组是一种大小固定的内置数据结构。而列表是一种由用户定义的、大小可变的结构，在这一章里表示为一个长度变量和一个项目数组，一起捆绑在一个类对象中。在第 13 章，我们为无序以及有序列表 ADT 设计了 C++类，并编写列表算法的代码，将其作为类的成员函数。第 14 章研究数据的动态分配和解除分配。然后，我们使用指针给出了第 13 章中无序、有序列表 ADT 的动态链接实现。

第 15 章扩展数据抽象和 C++类的概念，开始探讨面向对象的软件开发。这里介绍了继承的概念，学生们将会学习如何区分继承与组合。这一章还介绍了 C++虚函数；C++虚函数在运行时绑定运算，从而支持多态。这一章还更深入地回顾了在第 4 章简要介绍、第 12 章进一步扩展的面向对象设计。我们还给出一个纲要，说明如何由问题的描述转入软件解决方案的设计阶段。随后，我们在案例研究中遵循这一纲要，实现设计，并讨论测试策略。

第 16 章介绍 C++模板、运算符重载和异常处理。第 17 章讨论经典数据结构，并介绍其相应的 STL 结构。这一章还利用几种 STL 容器类重新实现了前几章的程序。

第 18 章以递归概念结束了全书。这一章共分三部分：使用简单变量的递归、使用数组的递归、使用指针的递归。这种组织方式使各位老师可以根据自己的喜好，将所有递归作为单独的主题进行讲解，也可以结合具有返回值的函数、数组和指针来分别介绍。

更多特色

特写小节

书中还给出了五种特色内容：

- **理论基础**　给出各计算机科学分支背后的基础理论材料。
- **软件工程提示**　讨论使程序更可靠、更健壮、更高效的方法。
- **风格问题**　强调了程序编码中的风格问题。

- **背景知识** 探讨了一些附带话题，为学生补充计算机科学的一般知识。
- **人物介绍** 给出了一些计算机先驱的传记，比如布莱斯·帕斯卡、查尔斯·巴贝奇、埃达·洛夫莱斯和格雷丝·默里·霍珀。

目标

每一章的开头都为学生列出一张目标清单，分为两类：知识目标和技能目标。章末习题用来对这两类目标进行巩固和测试。

演示程序

演示程序比案例研究示例短得多，也简单得多，它们在句法概念与解决实际问题中的应用之间架起了一道桥梁。现在的每一章都包含多个完整的演示程序，并穿插对新程序设计主题与语言主题的介绍。所有这些程序都可以在 go.jblearning.com/PPS6e 或 ituring.com.cn/book/1441/中获得，因此学生们可以很轻松地试验它们，并在自己的项目中重用这些代码。

软件维护案例研究

现代软件工程的主要工作就包括对原有代码的维护。鉴于这一事实，学生们很有必要学习一些阅读、理解、扩充和改正原有程序的技能。这些技巧很少在导论性课程中讲授，那些课程注重于根据问题描述编写新程序。而实际上，这些维护技能也是成功编写新程序的重要组成方面：编写了中等规模的代码，使其正常运行的过程实质上就是在进行维护。这些案例研究的目的就是培养阅读、剖析、修改和测试现有代码的技能。

实战案例研究

在这种案例研究中，本书很好地演示了解决实际问题的过程。每个“实战案例研究”都给出一个问题，并使用解决问题的方法逐步给出示范解决方案。接下来，我们使用功能分解方法或面向对象设计方法将解决方案扩展为一种算法；然后用 C++编写算法代码。我们还将给出示例测试数据与输出结果，接着探讨一些与程序全面测试相关的内容。

测试与调试

“测试与调试”部分跟在每一章的案例研究之后，深入探讨对该章内容进行全面程序测试的可能影响。它们都在最后给出一个测试与调试提示清单。

小测

每节最后给出了一些问题，用来检验学生对该章目标要点的记忆情况。在阅读每道题时，学生们应当立即给出答案，随后可以利用章末给出的答案（网上资源[①]）进行核对。每道题的最后都给出了讨论该概念的本书页码，使学生能够在回答有误时快速回顾相关内容。

备考练习（网上资源）

备考练习旨在帮助学生为参加考试做准备。这些问题通常都有客观答案，你应当在几分钟内完成解答。

程序设计热身练习（网上资源）

程序设计热身练习为学生提供了编写 C++代码段的体验。学生无须编写完整程序就可以练习每一章的语法构造。

程序设计问题（网上资源）

程序设计问题是从各种不同学科中提取出来的，要求学生设计解决方案和编写完整的程序。一些问题可能持续多章，后续章的“程序设计问题”要求学生使用新的构造或技术重新实现之前的解决方案，以此说明同一个问题可以有许多种不同的解决方案。

案例研究跟进（网上资源）

案例研究跟进练习要求学生回答一些需要理解或修改案例研究代码的问题，使学生们有机会强化自己的软

① 读者可到以下网址免费注册并下载：ituring.com.cn/book/1441/。——编者注

件维护技能。

补充资源

教师资源

线上资源可以提供有力的教学支持，这些资源中包括完整的练习答案（exercise answers）、测试题库（Test Bank）、PowerPoint 授课概要（PowerPoint lecture outlines），以及正文中的完整程序。读者可访问 go.jblearning.com/PPS6e 申请这些资源。

程序

书中所有完整程序的源代码都可以供老师和学生从出版社网站免费下载。此代码中包括所有案例研究程序，还有散布在各章中的演示程序。这些程序文件可以用任何标准文本编辑器查看和编辑，但要编译和运行这些程序，必须使用 C++编译器。

A Laboratory Course in C++, Sixth Edition

本书配套的数字实验手册采用与这一版教科书中相同的组织结构。其设计使教师在安排作业时拥有最大的灵活性，既可安排开放性实验，也可安排封闭式实验。每一章都包括三种活动：实验前、实验中和实验后。每一章都分为若干课，全面演示了各章介绍的概念。在本书网站上你可以找到与实验手册配套的程序、程序 shell（部分程序）和数据文件。

致谢

我们要感谢许多在这一版准备期间提供帮助的人们。尤其感谢得克萨斯大学奥斯汀分校、马萨诸塞大学阿默斯特分校计算机科学系的教职员工。

我们要特别感谢 Jeff Brumfield 开发了语法模板元语言，并允许我们在本书中使用。还要特别感谢 Tim Richards 帮助完善这一版及其补充材料。

感谢从事本书目标课程的讲师、助教、顾问、学监，还有学生们，他们给出了许多富有帮助的建议。

感谢以下人员花费时间，对这一版及先前各版可能做出的修改给出评论意见：纽约州立大学弗雷多尼尔分校的 Ziya Arnavut；伊利诺伊中央学院的 Trudee Bremer；东北伊利诺伊大学的 Mira Carlson；底特律大学的 Kevin Daimi；密歇根大学迪尔本分校的 Bruce Elenbogen；阿拉巴马大学汉茨维尔分校的 Letha Etzkorn；黑鹰学院的 Ilga Higbee；达拉斯浸会大学的 Sue Kavli；温斯顿–塞勒姆州立大学的 Sandria Kerr；费尔蒙特州立大学的 Alicia Kime；得克萨斯大学泛美分校的 Shahadat Kowuser；密歇根大学迪尔本分校的 Bruce Maxim；弗吉尼亚理工大学的 William McQuain；巴克内尔大学的孟宪农；布罗沃德大学的 William Minervini；沃什特瑙社区学院的 Janet Remen；奥克兰大学的 Viviana Sandor；弗吉尼亚理工大学的 Mehdi Setareh；底特律大学的 Katherine Snyder；密歇根大学迪尔本分校的 Tom Steiner；西切斯特大学的 John Weaver；南缅因大学的 Charles Welty；西切斯特大学的 Cheer-sun Yang。

我们还要感谢 Jones & Bartlett Learning 的许多人，他们为本书做出了巨大贡献，特别是高级组稿编辑 Tim Anderson、高级策划编辑 Amy Bloom、制作总监 Amy Rose。

任何一个曾经写过书的人（或者身边有人曾经写过书的人）都了解，这样一个工程需要花费大量时间。因此，感谢我们的家人——Dale clan 以及 Dale 大家庭的所有人（人数太多，无法一一列出姓名），感谢 Lisa、Charlie 和 Abby，感谢他们的大力支持和无限宽容。

Nell Dale

Chip Weems

下面的文字摘录自靡菲斯特的话，他是诱惑浮士德的主要魔鬼之一：

……我的朋友，我要劝你，
应当先从逻辑学起……
……假以时日让你知道
原本一蹴而就的事情
譬如平常的饮食
必要分出一、二、三
但思想之织品并无此等褶皱
更像是织工的佳作
一脚踏下，千条丝绪升起
织梭来回穿行
经线流淌，纤细难见
一打连出千万结
哲学家前来分析
证明它必为如此
第一如是，第二如是
则第三、第四亦如是
如无第一、第二
则第三、第四永不出现
所有学生都深信于此
却无一人学会纺织

——J. W. 冯·歌德，《浮士德》

你在学习本书时，不要让算法的逻辑束缚你的想象力，而要让它成为你织就思想艺术佳品的工具。

目　录

第1章 程序设计与问题解决概述

知识目标

- ❑ 理解什么是计算机程序。
- ❑ 理解什么是算法。
- ❑ 了解什么是高级程序设计语言。
- ❑ 理解编译和执行过程。
- ❑ 了解 C++语言的历史。
- ❑ 了解计算机的主要组件以及它们如何协同工作。
- ❑ 了解计算专业人员面对的一些基本行为准则问题。

技能目标

- ❑ 能列出编写计算机程序时所涉及的基本步骤。
- ❑ 能描述什么是编译器及其任务。
- ❑ 能区分软件和硬件。
- ❑ 能选择适当的问题求解方法，给出问题的算法方案。

1.1 程序设计概述

Com•put•er \kəm-py $\overline{\text{oo}}$ 'tər\ n. often attrib (1646): one that computes; *specif*: a programmable electronic device that can store, retrieve, and process data. ①

（计算机，名词，常作定语（1646）：计算者；特指一种能够存储、检索和处理数据的可编程电子设备。）

这是多么简单的一个定义呀，但它在仅仅几十年的时间里，已经改变了人们在工业化社会的生活方式！计算机涉及人们生活中的所有领域：支付账单、驾驶汽车、使用电话、购物。事实上，列出一份不受计算机影响的领域清单反而可以更快一些。

令人遗憾的是，如此出色的一种设备却常常受到人们的诋毁，让人退避三舍。“对不起，我们的计算机把事情弄坏了”“我就是搞不懂计算机；对我来说，它们太复杂了”，这样的话，你听到多少次了？不过，既然你正在阅读本书，那就意味着你准备把这些偏见扔到一边，着手学习计算机知识了。但我得提前提醒一下：本书可不只是在理论上讨论计算机，它教你学习如何为计算机设计程序。

1.1.1 什么是程序设计

人类的许多行为和思想都可以用逻辑顺序来描述。从幼年开始，我们就在学习如何行动、如何做事。我们还知道了如何预测其他人的特定行为。

我们日常生活中所做的很多事情，都是在下意识中完成的。很幸运，我们并不需要有意识地去分析一些简单过程中的每个步骤，比如用手翻书页：

---

① 获准使用。摘自 *Merriam-Webster's Collegiate Dictionary, Tenth Edition*. © 1994 Merriam-Webster Inc.

(1) 抬起手；
(2) 将手移到书的右侧；
(3) 捏住书页的右上角；
(4) 从右向左移动手，直到将书页放下并可以阅读书页另一面的位置；
(5) 松开书页。

想一想，为了移动胳膊和手，有多少个神经元必须被激发，又有多少块肌肉必须作出反应，而这一切都得遵循特定的次序。不过，我们在做这件事情时完全没有意识到这些。

人们在下意识中所做的很多事情都是需要学习的：观察一个婴孩在学习走路时是如何聚精会神地将一只脚放在另一只脚前面的，然后再观察一群三岁大的小孩子是怎样玩追人游戏的。

从更广泛的范围上来说，没有解决问题、证明定理的逻辑步骤，就不可能发展出数学，而没有按照特定顺序执行的作业，就不可能促生大规模生产。我们的整个文明就是以事件和行动的顺序为基础的。

程序设计　规划或安排一项任务或事件的执行顺序。
计算机　一种可以存储、检索和处理数据的可编程设备。
计算机程序　由计算机执行的指令序列。
计算机程序设计　为计算机规划执行步骤的过程。

我们有意、无意地通过一种称为**程序设计**的过程来创建步骤。本书研究的就是如何为我们的工具之一——**计算机**——来设计程序。

就像音乐会节目单列出演员演奏文艺作品的顺序一样，**计算机程序**列出了计算机执行的步骤。从现在开始，当我们再提到程序设计和程序时，就是指**计算机程序设计**和计算机程序。

用计算机完成任务，要比人工方式更高效、更快速、更准确，而有些任务根本就无法以人工方式完成。要使用这一功能强大的工具，我们必须详细说明希望它完成什么、按什么顺序来完成。程序设计就是做这个的。

1.1.2　怎样编写程序

计算机本身并不智能，它不能分析问题，给出解决方案。因此，必须由人（程序员）来分析问题，确定解决问题的指令序列，然后再将其告知计算机。如果计算机不能解决问题，那使用计算机有什么好处呢？答案是，一旦我们将解决方案编写为供计算机执行的指令序列，计算机就能一遍又一遍快速地、始终如一地重复这个解决方案。计算机将人类从重复性的单调任务中解放了出来。

要编写一个供计算机遵照执行的指令序列，我们必须经历两个阶段：问题求解和实现（见图 1-1）。

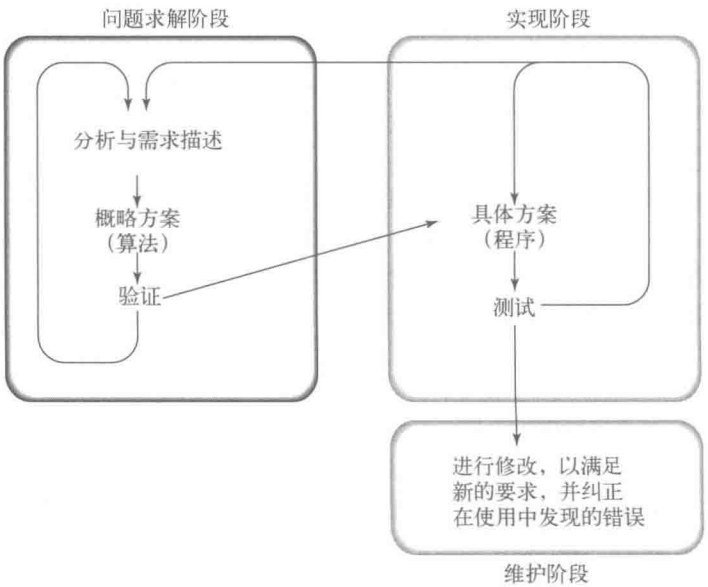


图 1-1　程序设计过程

1. 问题求解阶段

(1) 分析与需求描述。理解（定义）问题以及解决方案必须达到的目标。

(2) 概略方案（算法）。确定解决问题的逻辑步骤。

(3) 验证。严格执行方案中规定的步骤，看其能否真正解决问题。

2. 实现阶段

(1) 具体方案（程序）。将算法翻译为程序设计语言。

(2) 测试。让计算机执行指令，然后人工核对结果。如果发现错误，对程序和算法进行分析，确定错误来源并改正。

在编写了程序之后，我们就进入第三阶段：维护阶段。

3. 维护阶段

(1) 使用。使用程序。

(2) 维护。修改程序，使其满足新的需求，或者纠正在使用中发现的错误。

这一系列步骤称为软件开发的瀑布模型。还有其他一些开发模型，可以在不同情景下使用。比如，螺旋模型需要确定问题的初始需求，对一部分方案进行程序设计，请客户评估结果，然后再修改技术需求，并重复该过程，直到客户满意为止。如果最初的问题定义不是非常明确，或者问题的某些方面在开发过程中总在变化，那采用螺旋模型就比较合适。科学或工程研究问题经常采用螺旋模型来确定解决方案。本书始终采用瀑布模型，因为它非常适合解决那些定义非常明确的问题，而你将在程序设计入门课程中遇到的大多都是此类问题。

1.1.3 什么是算法

程序员在开始程序设计时，首先分析问题，确定一个称为**算法**的概略解决方案。问题的理解与分析要花费大量时间，远远超过图 1-1 所隐含的时间。这些步骤是程序设计过程的核心。

> **算法** 在有限时间内解答一个问题的逐步过程。

如果我们给出的计算机程序和算法的定义看起来有些相似，那是因为所有程序都是算法。程序就是为计算机编写出来的算法。

算法是对一序列逻辑操作的口头或书面描述。我们每天都在使用算法，处方、说明书等都是非程序算法的实例。

人们在发动汽车时，就是在执行一个逐步过程。其算法可能如下所示：

(1) 插入钥匙；

(2) 踩下制动踏板；

(3) 确认变速杆处于 P 档（或 N 档）；

(4) 将钥匙转动至发动位置；

(5) 如果引擎在 6 秒内成功发动，将钥匙释放至点火位置；

(6) 如果引擎没有在 6 秒内成功发动，松开钥匙和加速踏板，等待 10 秒，重复步骤(3)至步骤(6)，但不要超过 5 次；

(7) 如果汽车未能发动，请联系修理厂。

如果步骤(6)中没有提到“但不要超过 5 次”，你可能会一直尝试启动下去。为什么？因为如果汽车发生了什么故障，一遍遍地重复步骤(3)至步骤(6)并不能发动汽车。这种永不终止的情景称为无限循环。如果从步骤(6)中删除了“但不要超过 5 次”，这一过程就不符合我们给出的算法定义。对于任何可能发生的情况，算法都必须在有限时间内结束。

假定一位程序员需要一个算法来计算员工的周薪。下面的算法反映了人工完成工作的方式：

(1) 查看员工的工资标准；

(2) 确定该周的工作时数；

(3) 如果工作时数小于或等于 40，则将该时数乘以工资标准，计算出常规工资；