国外经典教材·计算机科学与技术

PEARSON
Addison Wesley

Data Abstraction and Problem Solving with C++:Walls and Mirrors, 3rd Edition

数据结构与C++高级教程（第3版）

(美) Frank M. Carrano
Janet J. Prichard 著
田玉敏 译

清华大学出版社

国外经典教材·计算机科学与技术

数据结构与C++高级教程

（第3版）

（美）Frank M. Carrano, Janet J. Prichard 著

田玉敏 译

清华大学出版社

北京

内 容 简 介

本书详细介绍了数据间的逻辑关系、存储方式和相关运算。针对各种实际问题，作者以C++程序设计语言为工具，说明了在问题求解过程中类和抽象数据类型的作用，并在许多实例和习题中使用了递归方法。同时，作者还提供了一个学习C++程序设计语言的教程，本教程可供初学者使用，对于已有一定基础的读者，也大有裨益。

本书可作为计算机及相关专业的本科生、研究生的教材和教学参考书，也可供程序开发人员自学。

EISBN: 0-201-74119-9

图书在版编目（CIP）数据

数据结构与C++高级教程（第3版）/（美）卡里诺（Carrano, F. M.），（美）普里查德（Prichard, J. J.）著；田玉敏译. —北京：清华大学出版社，2004

（国外经典教材·计算机科学与技术）

书名原文：Data Abstraction and Problem Solving with C++:Walls and Mirrors, 3rd Edition

ISBN 7-302-08326-6

Ⅰ.数… Ⅱ.①卡…②普…③田… Ⅲ.①数据结构－教材②C语言－程序设计－教材

Ⅳ.TP311.12②TP312

中国版本图书馆CIP数据核字（2004）第022118号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969

文稿编辑：徐刚
封面设计：久久度文化
印 刷 者：北京鑫霸印务有限公司
装 订 者：三河市化甲屯小学装订二厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：42.75　**字数**：1040千字
版　　次：2004年6月第3版　2004年6月第1次印刷
书　　号：ISBN 7-302-08326-6/TP·6000
印　　数：1～4500
定　　价：69.00元

出版说明

近年来，我国的高等教育特别是计算机学科教育，进行了一系列大的调整和改革，急需一批门类齐全、具有国际先进水平的计算机经典教材，以适应当前我国计算机科学的教学需要。通过使用国外先进的经典教材，可以了解并吸收国际先进的教学思想和教学方法，使我国的计算机科学教育能够跟上国际计算机教育发展的步伐，从而培育出更多具有国际水准的计算机专业人才，增强我国计算机产业的核心竞争力。为此，我们从国外知名的出版集团Pearson引进这套“国外经典教材·计算机科学与技术”教材。

作为全球最大的图书出版机构，Pearson在高等教育领域有着不凡的表现，其下属的Prentice Hall和Addison Wesley出版社是全球计算机高等教育的龙头出版机构。清华大学出版社与Pearson出版集团长期保持着紧密友好的合作关系，这次引进的“国外经典教材·计算机科学与技术”教材大部分出自Prentice Hall和Addison Wesley两家出版社。为了组织该套教材的出版，我们在国内聘请了一批知名的专家和教授，成立了一个专门的教材编审委员会。

教材编审委员会的运作从教材的选题阶段即开始启动，各位委员根据国内外高等院校计算机科学及相关专业的现有课程体系，并结合各个专业的培养方向，从Pearson出版的计算机系列教材中精心挑选针对性强的题材，以保证该套教材的优秀性和领先性，避免出现“低质重复引进”或“高质消化不良”的现象。

为了保证出版质量，我们为该套教材配备了一批经验丰富的编辑、排版、校对人员，制定了更加严格的出版流程。本套教材的译者，全部来自于对应专业的高校教师或拥有相关经验的IT专家。每本教材的责编在翻译伊始，就定期不间断地与该书的译者进行交流与反馈。为了尽可能地保留与发扬教材原著的精华，在经过翻译、排版和传统的三审三校之后，我们还请编审委员或相关的专家教授对文稿进行审读，以最大程度地弥补和修正在前面一系列加工过程中对教材造成的误差和瑕疵。

由于时间紧迫和受全体制作人员自身能力所限，该套教材在出版过程中很可能还存在一些遗憾，欢迎广大师生来电来信批评指正。同时，也欢迎读者朋友积极向我们推荐各类优秀的国外计算机教材，共同为我国高等院校计算机教育事业贡献力量。

清华大学出版社
2004.03.20

译者序

本书是作者多年来讲授数据抽象和问题求解方法的经验总结。概括起来，本书有以下几个特点：

- 形象地将常用的两种基本问题求解方法——数据抽象和递归比喻成墙和镜子。
- 强调数据抽象的作用。问题求解方法始终贯穿于抽象数据类型的设计、实现及其描述，同时举例说明了在问题求解过程中类和抽象数据类型(ADT)的作用，论述了 ADT 的主要用途。
- 分别用英语、伪代码和 UML 表示法给出了所有重要的 ADT 的规范说明。
- 深入地介绍了递归的概念，讨论了简单的递归定义和语言识别、检索、排序等递归算法的例子。
- 重点集中在数据结构而不是语言的语法上，所有的 C++代码都用 ANSI C++进行了验证。
- 包括了标准模板库的内容。
- 介绍了标准建模语言。

本书的读者对象很广泛，可以作为计算机及其相关专业的本科生、研究生的教材和教学参考书，也可供程序开发人员自学使用。

本书由西安电子科技大学田玉敏译。贾元元、周正林、牛海敏、王君、孟丽娜、吕爱琴、康效龙、邹娟，潘洪涛等作为本书的第一批读者，他们花了很多时间，指出了本书的许多不足之处。其中王君、孟丽娜、康效龙、潘洪涛分别是附录、第 5 章、第 6 章、第 7 章、第 8 章、第 11 章、第 12 章的初稿译者，正是由于这些朋友的帮助，才使得本书能够顺利完成。由于本书涉及的内容较多、翻译难度较大；加之译者水平有限，书中难免有疏漏和错误之处，欢迎读者批评指正。

田玉敏
2004 年 2 月 1 日

前　言

欢迎阅读本书。自本书第2版面世以来，通过使用C++和面向对象的方法讲授数据抽象，我们获取了更多经验。这一版本集中反映了C++语言的发展和我们的实践经验。

本书源于Paul Helman和Robert Veroff编著的*Intermediate problem Solving and Data Structures: Walls and Mirrors*一书(该书1986年由Benjamin/Cummings有限出版公司出版)。本书基于原书的组织框架和整体观点，其中包括相关的技术以及文字，例如，从原著中选取的图(表)以及习题。Helman和Veroff教授借用了两个非常强大的比喻，墙(wall)和镜子(mirror)，这简化了我们讲解和学习计算机科学的难度。

本书的重点是数据抽象和其他问题求解工具，是作为计算机科学的第二教程来设计的。考虑到大学本科计算机课程的极大灵活性和要求的多样性，本书的主题覆盖了相当广泛的内容以使其适应其他课程。例如，可以在介绍数据结构或高级语言程序设计及其问题求解课程中使用本书。本书的目标仍然是在数据抽象、面向对象程序设计和其他现代问题求解方法方面给学生打下坚实的基础。

致学生

很多大学生阅读并学习过原书第2版。标题中的墙和镜子表示贯穿于整本书的两种基本的问题求解方法。数据抽象将模块的实现细节同程序的其他部分相隔离，很像把你同邻居隔开并隐藏自己的墙。递归是通过重复解决类型绝对相同的小型子问题来解决大问题的一种方法 。

编写本书时我们一直将读者放在第一位。我们也曾经是大学生，我们也是坚持学习的教育工作者，我们更能理解清晰表述的重要性，我们的目的是使本书尽可能便于理解。

本书对读者的C++知识有一些基本假设。一些读者可能需要学习C++语言或参考本书附录A。读者需要掌握if和switch语句；for，while和do语句；函数以及参数传递、数组、串、结构和文件。本书在第1章、第3章和第8章中讨论了C++类，我们假设读者没有这方面的基础。我们还假设读者也不具备第2章、第5章中论述的递归函数的使用经验。

本书中所有C++源代码都可以供你使用。前言中稍后部分列出的补充材料告诉读者如何获得这些文件。

方法

本版进一步强调数据抽象和数据结构。本书详细说明了C++语言的优势和不足，同时继续采用容易理解的教学方法和材料。

背景知识

本书假设读者已具备C++或其他语言的基础知识，并有教师帮助他们过渡到C++。本书正式介绍了C++类，并没有采用以前有关C++类的知识。本书的内容包括用C++描述的面向对象程序设计的基本概念、继承、虚函数以及类模板。尽管本书介绍这些内容以及抽象数据类型的实现，但重点仍然是ADT，而不是C++。这些内容在面向对象程序设计部分中有介绍，但我们假设未来的课程将详细讨论面向对象设计和软件工程，因此重点仍是数据抽象。不过，我们确实引入了统一建模语言(Unified Modeling Language，简称UML)作为设计工具。

适应性

本书所涵盖的广泛内容为我们的课程提供了必要的资料。读者可以选择其需要的内容，按照个性化的学习顺序进行。各章的从属关系图说明了在某个章节前应先学习的章节。

第Ⅰ部分中的各章可以根据学生的背景知识选择。其中有3章大篇幅地介绍了数据抽象和递归。数据抽象和递归都很重要，但到底应先讲哪部分却意见不一。尽管本书递归的章节在前，数据抽象的章节在后，但这一顺序可以重新调整。

第Ⅱ部分的内容也可用灵活的顺序论述。例如，可在介绍栈(第6章)之前或之后介绍高级C++(第8章)；也可以在学完第5章后的任何时候引入算法效率与排序(第9章)；可以在队列之前介绍树，或在查找表一章之前介绍图，或在学完查找表后学习散列法、平衡二叉树和优先级队列(其他顺序亦可)，也可在课程中稍早的时候学外部方法(第14章)，例如，在学完第9章中的归并排序后介绍外部排序。

数据抽象

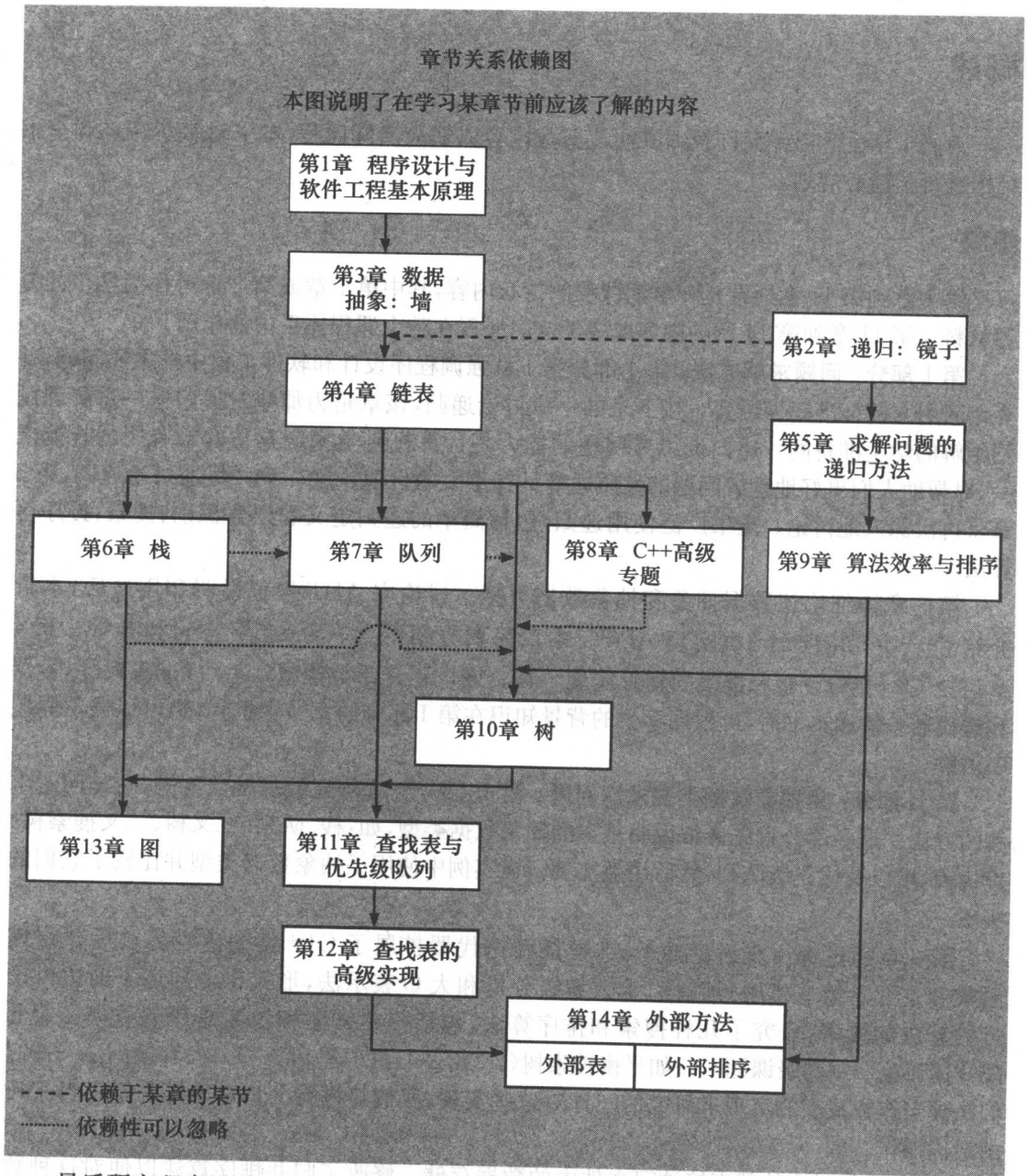
抽象数据类型的设计及其应用是始终贯穿本书的问题求解方法。我们用几个实例说明了ADT如何设计。我们首先用英语和伪代码定义了所有ADT，然后在论述这些ADT的实现之前，在简单的应用中使用了它们。ADT与实现它的数据结构之间的差异仍然在最前面论述。本书一开始就解释了封装和C++类，并使学生明白C++类如何在ADT的客户程序中将已实现的数据结构隐藏起来。抽象数据类型包括：表、栈、队列、树、查找表、堆和优先级队列，它们构成了我们讨论的基础。

问题求解

本书通过研究计算机科学家所用的方法和思维过程，帮助学生掌握问题求解和程序设计的能力。学习开发、分析和实现解决方案的方法同学习算法一样重要。

本书包括针对示例问题的分析方法，在贯穿整本书的问题解决方案设计过程中，涉及了算法、数据结构以及递归逐步求精。

本书很早就介绍了C++指针和链表的处理，并在构建数据结构时使用了它们。本书还初步介绍了算法的数量级分析方法。这种方法首先非正式地考虑，然后进一步详细地考虑基于数组和基于指针的数据结构的优缺点。各种可能的解决方案的折衷选择及其实现是问题求解的一个核心课题。



最后研究程序设计风格，包括前提和结果的文档和辅助调试手段。循环不变量是实现、验证解决方案的问题求解方法学的重要组成部分。整本书都论述这些课题。

应用

本书的重点章节给出了典型应用，如：折半检索、快速排序和归并排序算法，这些是递归的重要应用，而且还介绍了数量级分析。在诸如平衡二叉树搜索、散列法和文件索引这样一些课题中，又继续讨论搜索问题。在外部文件一节中还将再次讨论搜索和排序问题。

概述

为便于学生学习并允许教师方便地剪裁内容以适应实际课程，本书对教学特点和全书结构进行了精心规划。

组织

第 1 章到第 11 章通常是一学期课程的核心内容，其中第 1 章或第 2 章可作为学生的复习材料。第 11 章到第 14 章的内容取决于这门课程在整个课程体系中的作用。

第 I 部分：问题求解方法。第 I 部分第 1 章强调程序设计和软件工程中的主要问题，并介绍新引入的标准建模语言。接下来的一章讨论递归，该章是为那些对递归这一重要课题不了解的学生准备的。递归地思考问题的能力是计算机科学家应具备的最有用的技能之一，对帮助人们更好地理解问题的性质通常具有重大的价值。这一章详尽地讨论递归，第 5 章将再次深入地讨论。全书广泛使用递归，包括简单的递归定义和语言识别、搜索、排序等算法。

第 3 章详细论述数据抽象和抽象数据类型。讨论完 ADT 规范说明和用途后，本章介绍 C++类，并用 C++类实现 ADT。本章简要介绍继承、C++名字空间和异常。第 4 章讨论 C++指针变量和链表的辅助实现工具。该章还介绍类模板、C++标准模板库，容器和迭代器。使用本书时，可根据学生的背景知识在第 I 部分的章节中选择，也可以按不同顺序讲解。

第 II 部分：用抽象数据类型求解问题。第 II 部分继续研究数据抽象问题，它是问题求解的方法。这一部分首先详细说明基本的抽象数据类型，如：栈、队列、二叉树、二叉搜索树、查找表、堆以及优先级队列，然后用类实现。在实例中使用了抽象数据类型并比较了它们的实现。

第 8 章通过进一步讨论继承、类模板和迭代器扩展了 C++类的内容。之后介绍虚函数和友元。第 9 章中，通过引入数量级分析和大 O 表示法，形式化地描述了前边讨论过的算法的效率，研究了几种搜索和排序算法，包括归并排序和快速排序的效率。第 II 部分还讨论一些高级课题——如平衡搜索树(2-3，2-3-4，红一黑和 AVL 树)和散列法——它们是被当作查找表的实现来研究的。针对这些实现，我们将进行分析以确定它们各自支持的最优操作。

最后，我们讨论外部直接访问文件中的数据存储。修改了归并排序算法以便对这种数据进行排序，使用外部散列法和 B-树索引对其进行搜索。这些搜索算法是已经讨论过的内部散列法方案和 2-3 树的推广。

补充材料

教师和学生可以在线获得下列补充材料：

- 源代码：读者可以使用本书中的所有 C++类、函数和程序。
- 勘误表：我们尽力不犯错误，但错误是难免的。勘误表列出了目前已发现的错误和必

要的修正(指英文原版书中的勘误表——译者注)。请将您发现的错误反馈给我们(包括英文原版和中文简体翻译版中的错误,我们将分别转达给原著作者和中文译者——本书责任编辑注)。这里预致谢意。

- 通过 *http://www.aw.com/cssupport*,你可以得到源代码和勘误表。

目　录

第Ⅰ部分　问题求解方法

第Ⅱ部分　用抽象数据类型求解问题

第 I 部分　问题求解方法

本书第 I 部分中的 5 章主要讨论问题求解方法的技巧，这构成了本书其他章节的基础。第 1 章首先叙述了一个好的解决方案的特点以及获取好解决方案的途径。这些方法强调了抽象、模块化和信息隐藏。第 I 部分的其他章节讨论方案设计中的数据抽象，在实现过程中使用的 C++指针和类以及递归。

第 1 章　程序设计与软件工程基本原理

本章概述了几个基本原理，这些是处理大型程序的基础。论述了程序设计的基本原理，同时也说明了编写精心设计、资料完备的程序可以避免许多消耗。本章还简要地讨论算法和数据抽象问题，指出这些问题与本书主题——提高问题求解与程序设计技能的相关性。在后续章节，重点从程序设计的基本原理转到数据的组织与使用方法上。即使讨论重点落到这些新方法时，仍应注意解决方案如何遵循本章讨论的基本原理。

1.1　问题求解与软件工程

你最近一次写程序是从哪里开始的？读完问题描述并等待片刻后，大多数缺乏经验的程序员自然开始了编码。显然，他们的目标是使程序运行，最好能得到正确结果。因此，他们运行程序，检查错误信息，插入分号，修改逻辑，删去分号，祈祷，还要用其他那些折磨人的方法，直到程序工作正常。他们的大部分时间也许是花在检查语法和程序逻辑上了。当然，程序员的程序设计技能比他们第一次写程序时强多了，但使用刚才的方法，程序员真能编写出大型程序吗？也许能，但有更好的方法。

要点提示：没有方案设计就进行编码，将增加调试时间。

应该明确的是，特大型软件开发项目通常需要一组程序员而不仅仅是一个程序员。而团队工作需要整体规划、组织和交流。没有计划的程序设计方法不适合小组中的成员，而且存在严重的浪费。幸运的是，计算机科学的一个分支，即软件工程，促进了计算机程序开发方法的发展。

要点提示：软件工程促进了程序开发方法的发展。

尽管计算机科学的第一门课程一般强调程序设计问题，但本书重点是在问题求解问题，该问题覆盖面更加宽广。本章首先概述问题求解过程和解决问题的各种方法。

1.1.1 什么是问题求解

此处的“问题求解”指的是理解问题的陈述并开发解题程序的整个过程。这一过程需要经过许多阶段，从理解问题开始，通过设计概念性解决方案，一直到用计算机程序实现这个解决方案。

要点提示：解决方案规定了算法和存储数据的方法。

到底什么是解决方案呢？通常，解决方案由两部分构成：算法以及存储数据的方法。算法是在有限时间内，对解题方法的一步步的说明。算法经常对大量数据进行操作，例如：一个算法可能将一个新的数据加入某个集合中，或从某个集合中删除数据，或查询某个数据集合。

也许对解决方案的描述会给人留下一种虚假的印象，问题求解过程中人的所有聪明才智都用来研究算法了，而如何存储数据仅仅起支撑作用。这种印象远远不是实际情况，实际上，要做的事情比单纯地存储数据多得多。在构造解决方案时，必须对数据集精心组织以便按照算法要求的方式很方便地操作数据。事实上，本书大部分内容论述组织数据的方法。

在设计已知问题的解决方案时，可以利用简化任务的几种方法。本章简单介绍这些方法，后续章节将更详细地讨论。

1.1.2 软件的生命周期

优秀软件的开发是一个漫长的、持续的过程，这个过程称为软件的生命周期。这个过程从最初的一种思想开始，包括编写和调试程序，以及持续多年地对最初软件的修改与完善。图1.1把软件生命周期的9个阶段描绘成水车（这里要感谢Raymond L. Paden，他建议我把“轮子”改成“水车”）上的扇形。这种安排暗示了软件生命周期各阶段是一个周期的一部分，而不仅仅是个线性表。尽管是从定义问题开始的，但通常可以从任意一个阶段转到其

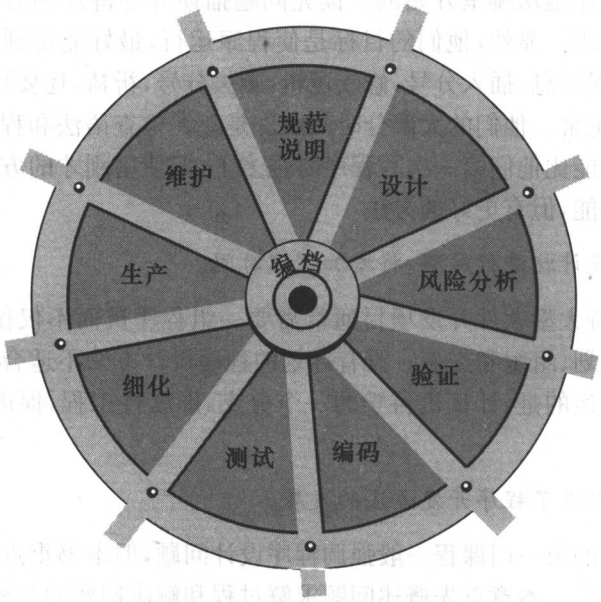


图1.1 软件生命周期比喻成能从一个阶段转到其他阶段的水车

他阶段，例如，测试程序时可能需要修改问题的定义或设计方案。请注意，图中环绕在文档核心的 9 个阶段。文档并没有像人们期望的那样分段，而是与软件生命周期的各阶段结成一体。

因而，这是典型的软件生命周期的各阶段。尽管各阶段都重要，但在此我们只详细讨论与本书最相关的那些阶段。

第 1 阶段：规范说明。已知软件用途的最初描述后，必须清楚地说明问题的各方面。通常描述问题的人并不是程序员，因此最初对问题的描述可能不确切。因而，规范说明阶段要求精确化、细节化原始问题的描述，还要与程序员和非程序员沟通。

这里是书写软件详细规范时必须回答的几个问题。输入数据是什么？什么数据是合法数据？什么数据是非法数据？谁将使用软件？应该使用什么样的用户接口？需要检查什么错误并给出什么样的错误信息？什么假设是可能的？有特殊情况吗？输出格式是什么？什么文档是必需的？程序将来很可能要增加什么功能？

要点提示：要使程序描述精确化、细节化。

改进人们之间的交流、理解软件规范的方法之一是编写原型程序(prototype program)，用它来模拟软件产品的部分行为，例如：一个简单的，甚至是低效的程序，可以演示用户接口以便我们分析，此时发现问题或改变想法要比在程序设计中，甚至是程序设计完成后处理要好得多。

要点提示：原型程序可以澄清问题。

你以前的程序设计任务可能已经陈述了程序设计规范，也许这些规范的某些方面不清楚，你不得不进一步澄清，但最可能的是你在编写自己的程序规范方面几乎没有实际经验。

第 2 阶段：设计。一旦规范说明阶段完成，就必须为问题设计一个解决方案。对大多数人来说，一次为整个程序设计规模和复杂程度适中的解决方案很困难。简化问题求解过程的最好方法是将大问题分解成若干小的、容易处理的部分。最终的程序将包含模块(module)。模块是独立的代码单元。一个模块可能是单独一个函数，或是几个函数和其他程序块的代码。模块的设计应尽可能独立，即低耦合，接口部分当然除外。接口负责各模块间的通信。我们可以开发与其他模块相互独立的模块。各模块应该完成一项意义明确的任务，即应该是高内聚的。因此，模块化描述了程序模块的高内聚、低耦合的组织方式。

要点提示：低耦合模块是相互独立的。

要点提示：高内聚模块完成一项意义明确的任务。

在设计阶段，不但明确规定各模块的功能很重要，而且明确规定各模块间的数据流也很重要。例如，针对各模块应明确如下问题，在执行模块前，模块可用的数据是什么？模块做了哪些假设？已经发生了什么动作以及模块执行后数据看起来是什么样子？因此，应详细说明各模块的假设、输入以及输出。

要点提示：应详细说明各模块的用途、假设、输入和输出。

例如，作为整数数组排序程序的设计者，可能为排序函数写出这样的规范说明：此函数将接收包含 *num* 个整数的数组，其中 $num>0$；此函数将返回排序后的数组。可以把这些规

范说明看作函数和调用模块间的合同(contract)条款。

要点提示:如果你独立编写整个程序,这份合同有助于你系统地把问题分解成较小的任务。如果程序是一个程序员组的项目,该合同有助于描述责任,无论谁编写排序函数,都必须履行这份合同。排序函数的编写、测试完成后,合同将描述程序的其他部分如何正确地调用排序函数以及相应结果。

然而应注意,模块的合同并没有限定完成任务的特定方法,这一点很重要。如果程序的另外一部分对该方法做了任何假设,它都要自己承担风险。因此,如果以后的某个时间,有人采用不同的排序算法重新改写了这个函数,应该根本不需要修改程序的其他部分。只要新函数遵守原始的合同,程序的其余部分显然不应改变。

要点提示:模块的规范说明不应该说明解决方案的特定方法。

要点提示:函数的规范说明包括明确的前提和结果。

以上论述对你来讲应该不是什么新内容。尽管从前你可能没有那么明白地使用"合同"这个术语,但有关概念应该熟悉。在书写函数的前提及结果时,实际就是在写合同,前提是函数开始时必备条件的说明,结果是函数结束时的条件的说明。例如,依据上述合同,以伪代码形式书写的排序函数:

```
sort(anArray,num)
// 数组排序
// 前提:数组是有 num 个整数的数组,num>0
// 结果:数组中的整数是有序的
```

要点提示:以上是规范说明的初稿。

这些特定的前提和结果实际上是不完善的,很可能就是合同初稿。例如,"排序"是升序还是降序? *num* 的值可以是多大? 在实现这个函数的时候,可能假定"排序"的意思是升序,*num* 的值不超过 100。设想一下,当另一个人尝试使用 sort 对 500 个整数的数组按降序排列的时候可能遇到的困难,除非在文档说明中修改前提与结果,否则这个用户不知道相关的假设。修改后的规范说明如下:

```
sort(anArray,num)
// 数组以升序排序
// 前提:数组有 num 个整数,1 < num < MAX_ARRAY,MAX_ARRAY 是
// 规定数组最大长度的一个全局常量
// 结果:anArray[0] <= anArray[1] <=…<=anArray[num-1],num 不变
```

要点提示:以上是修改后的规范说明。

写前提时,首先描述函数的输入参数,说明函数使用的所有全局常量,最后列出编写函数时所做的假设。同样,在写结果时,首先描述函数的输出参数。如果是求值函数,就是函数返回的值,然后描述已经发生的其他所有动作。

缺乏经验的程序员容易忽视文档的准确性,特别是当他们同时作为设计师、程序员和一个小程序的用户的时候。如果你设计了 sort 但没有写下合同条款,此后实现该函数时还能记着这些合同条款吗? 编写完 sort 函数数周后还能记住如何使用 sort 吗? 为了想起被遗