高等学校应用型“十二五”规划教材 • 计算机类

数据结构与算法分析

荣 政 主编
樊 凯 刘彦明 邵晓鹏 车书玲 参编

西安电子科技大学出版社
http://www.xduph.com

高等学校应用型“十二五”规划教材·计算机类

数据结构与算法分析

荣　政　　主　编

樊　凯　刘彦明　　参　编

邵晓鹏　车书玲

西安电子科技大学出版社

内 容 简 介

本书以程序设计能力的培养为目标，系统地介绍了数据结构和算法设计的相关知识，其主要内容包括：线性表、栈、队列、串、数组、树、图、索引和散列等基本数据结构及其应用；分治法、动态规划、贪心算法、回溯法、分支界限法等常用的算法设计方法。书中还通过具体实例的分析和设计，介绍了软件设计规范及程序设计的关键技术，具有较高的使用价值。

本书可作为高等学校电子信息类非计算机专业“数据结构”课程的本科(或大专)教材，也可供自学计算机软件基础知识的读者参考。

图书在版编目(CIP)数据

数据结构与算法分析 / 荣政主编.—西安：西安电子科技大学出版社，2012.2

高等学校应用型“十二五”规划教材

ISBN 978-7-5606-2718-2

Ⅰ. ① 数… Ⅱ. ① 荣… Ⅲ. ① 数据结构—高等学校—教材 ② 算法分析—高等学校—教材

Ⅳ. ① TP311.12

中国版本图书馆 CIP 数据核字(2011)第 278528 号

策　　划　李惠萍

责任编辑　李惠萍　曹　锦

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)88242885　88201467　　邮　　编　710071

网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com

经　　销　新华书店

印刷单位　陕西天意印务有限责任公司

版　　次　2012 年 2 月第 1 版　2012 年 2 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　　印　张　19

字　　数　450 千字

印　　数　1～3000 册

定　　价　33.00 元

ISBN 978-7-5606-2718-2/TP • 1317

XDUP 3010001-1

前　言

计算机的使用极大地改变了人们的生活。目前，各行业、各领域都广泛采用了计算机信息技术，并由此产生了开发各种应用软件的需求。为了以最少的成本、最快的速度、最好的质量开发出适合各种应用需求的软件，必须遵循软件开发的原则，掌握软件开发的基础知识和基本技能。一个熟练的程序设计人员，至少应具备以下三个条件：一是熟练地掌握一门程序设计语言；二是能够熟练地选择和设计各种数据结构和算法；三是要熟知应用领域的相关知识。数据结构和算法的设计是区分一个程序员水平高低的重要标志，缺乏数据结构和算法的深厚功底，是很难设计出高水平应用程序的。

本书是在多年“数据结构”课程教学实践的基础上，根据新的教学计划和读者对原有教材的修改建议编写而成的。全书共 10 章，分为两个部分。第一部分包含第 1 章到第 7 章，第 1 章综述数据、数据结构、抽象数据类型、算法等基本概念，介绍了程序设计中的关键技术；第 2 章到第 7 章介绍了线性表、栈、队列、串、数组、树、图、索引和散列等基本数据结构及其应用。第二部分包含第 8 章到第 10 章，介绍了分治法、动态规划、贪心算法、回溯法、分支界限法以及一些“难”问题的求解算法。

本书以培养学生的程序设计能力为主要目标，通过对数据结构与算法分析的介绍，使读者学会分析研究计算机加工的数据特性，培养其对数据的抽象能力，为选择(或设计)合适的数据结构及其相应的算法奠定基础。另外，学习的过程也是复杂程序设计的训练过程，因此本书在第 1 章通过具体实例介绍了软件开发应遵循的原则及程序设计的关键技术，使读者在掌握一门程序设计语言的基础上，能够在实际应用中，编写出结构清晰、正确易读、符合软件工程规范的程序。虽然全书采用类 C 语言作为数据结构和算法的描述语言，但是为了方便读者学习及编程，在书中第一部分介绍的大多数算法都是以规范的 C 函数形式给出的，有的应用实例甚至给出了经过上机调试的完整 C 程序，这为指导学生上机实践提供了便利。在书中的第二部分，通过介绍经典算法，对数据结构及其应用进行了深入的研究和探讨，将应用实例与数据结构和算法设计方法紧密地结合起来，不但能更好地激发学生的学习兴趣，而且使他们对数据结构和算法在软件设计中的作用有了更深入的理解，有助于其程序设计水平更上一层楼。

本书在编写过程中得到了西安电子科技大学通信工程学院、技术物理学院各位领导的大力支持，西安电子科技大学出版社的领导和编辑对本书的出版进行了精心的组织和安排，在此衷心地表示感谢。

限于作者水平，书中难免存在不当之处，恳请同行专家和读者批评指正。

作　者

2011 年 9 月

目　录

第1章 绪 论

随着计算机科学技术的发展和日趋成熟，软件已经成为社会发展的重要驱动力之一：软件是商业决策的基本引擎，是解决现代科学研究和工程问题的基础，也是提升现代产品和服务品质的关键因素。软件在现代社会中是不可缺少的，它可以应用于各种类型的应用系统中，如办公、娱乐、交通、教育、医药、通信……数不胜数，并成为从基础教育到基因工程的所有领域的推进器。

所有这一切改变了软件原有的概念，软件无所不在，已经成为人们现实生活中必不可少的实用技术。

1.1 软件的基本概念

从一般意义上讲，能够在计算机上运行的程序都可以称为软件。软件更准确的定义是利用计算机本身提供的逻辑功能，合理地组织计算机的工作流程，简化或替代人们使用计算机过程中的各个环节，提供给使用者一个便于操作的工作环境的“程序集”。它包括计算机程序和与之相关的文档资料的总和(文档是指编制程序所使用的技术资料和使用该程序的说明性资料，如使用说明书等，即开发、使用和维护程序所需的一切资料)。

软件是逻辑的而不是物理的产品，与硬件相比具有以下完全不同的特征：

(1) 软件是由开发或工程化而形成的，而不是由传统意义上的制造产生的。硬件的再制造过程中可能引入质量问题，软件则几乎不可能发生类似问题。软件的成本集中于开发，而硬件的成本除了开发成本外还有生产成本。

(2) 软件不会磨损。软件的故障率随时间的推移而降低，而硬件的故障率随时间的推移而增加。硬件模块磨损后可以用新的备用模块替换，而软件故障则是其中的错误，没有可替换的备件。

(3) 大多数软件是自定义的，而不是通过已有的组件组装而来的。当然，目前基于组件的软件开发已经成为软件未来的主要发展趋势。

1.1.1 软件应用

从某种程度上讲，对软件应用给出一个确切的分类是比较困难的。下面给出的一些软件的应用领域，可以看做是一种潜在的应用分类：

(1) 系统软件。系统软件是指一组为其他程序服务的程序，如操作系统、编译器等。

(2) 实时软件。管理、分析和控制现实世界中发生的事件的程序称为实时软件。实时软件包括：① 数据接收部件——负责从外部环境获取和格式化信息；② 分析部件——负责将信息转换为具体应用所需要的形式；③ 控制/输出部件——负责对外部事件做出响应；④ 管理部件——负责协调其他各部件，使得系统能够保持一个可接受的实时响应时间。

(3) 商业软件。商业信息处理是目前最大的软件应用领域。具体的“信息系统”均可归入管理信息系统(MIS)软件。使用这类软件可以访问一个或多个包含商业信息的大型数据库。该领域的应用使已有的数据重新构造，变换成一种能够辅助商业操作或管理决策的形式。除了传统的数据处理应用外，商业软件还包括交互式计算和客户/服务器模式计算(如POS 事务处理)。

(4) 工程和科学计算软件。工程和科学计算软件的特征是“数值分析”算法。此类软件应用的涵盖面很广，从天文学到火山学，从汽车压力分析到航天飞机的轨道动力学，从分子生物学到自动化制造等。

(5) 嵌入式软件。智能产品在工业和消费电子市场上都是必不可少的，而嵌入式软件正是驻留在这些智能产品的只读内存中，实现产品的智能应用。嵌入式软件能够执行有限但专职的功能。

(6) 个人软件。仅供个人使用的程序称为个人软件，如文字处理、电子表格、多媒体娱乐、数据库管理、个人金融应用、访问外部网络应用等。

(7) 人工智能软件。利用非数值算法解决复杂的问题，这些问题不能通过计算或直接分析得到答案，这就要使用人工智能软件，如专家系统(基于知识的软件)、模式识别、定理证明、游戏等。

1.1.2 软件生存期

软件生存期是指从软件项目的需求定义到完成使命后而废弃的全过程。把整个过程中的活动进一步展开，可以得到软件生存期的六个工作阶段，即制定计划、需求分析和定义、软件设计、程序编制、软件测试及运行/维护。

1. 制定计划

首先要确定开发软件系统的总目标，给出对其在功能、性能、可靠性以及接口等方面的要求；然后由系统分析员和用户合作，论证该项软件任务的可行性，探讨解决问题的方案，并对可利用的资源(计算机硬件、软件、人力等)、研发的成本、可取得的效益、开发的进度做出评估，制定出完成开发任务的实施计划，连同可行性研究报告，提交管理部门审查。

2. 需求分析和定义

对开发软件提出的需求进行分析并给出详细的定义。首先由软件人员和用户共同讨论决定哪些需求是可以满足的，并对其加以确切的描述；然后编写出软件需求说明书或系统功能说明书，以及初步的系统用户手册，提交管理机构评审。

3. 软件设计

在软件设计阶段中，设计人员把已确定的各项需求转换成一个相应的体系结构。结构中的每一个组成部分都是意义明确的模块，每个模块都和某些需求相对应，即概要设计。然后对每个模块要完成的工作进行具体的描述，为源程序的编写打下基础，即详细设计。在设计中涉及

的所有内容都应以设计说明书的形式加以描述，以供在后继工作中使用并提交评审。

4. 程序编制

程序编制是指把软件设计转换成计算机可以接受的程序代码，即写成以某一种特定程序设计语言表示的“源程序清单”，也称为编码。当然，写出的程序应当是结构良好、清晰易读，且与设计要求相一致的。

5. 软件测试

软件测试是保证软件质量的重要手段，其主要方式是在设计测试用例的基础上检验软件的各个组成部分。首先是进行单元测试，查找各模块在功能和结构上存在的问题并加以纠正；其次是进行组装测试，将已测试过的模块按一定顺序组装起来；最后按规定的各项需求，逐项进行有效性测试，决定已开发的软件是否合格，能否交付用户使用。

6. 运行/维护

已交付的软件投入正式使用后，便进入运行阶段。这一阶段可能持续若干年甚至几十年。软件在运行中可能由于多方面的原因，需要对它进行修改，其原因可能有：运行中发现了软件中的错误需要修正；为了适应变化了的软件工作环境，需做适当的变更；为了增强软件的功能需做变更。

将软件生存期分成上述六个阶段，为研制软件提供了一个框架。然而实际软件系统的研制工作不可能是直线进行的，肯定存在着反复，软件设计人员往往会在后阶段发现前阶段工作结果中的错误或疏漏，这就需要回到前阶段进行纠正或补充。另外，软件生存期各阶段之间没有明确的界限，如分析阶段可能需要考虑一些设计上的问题，编制阶段也可能与模块的测试并行进行等。软件生存期模型如图 1-1 所示。

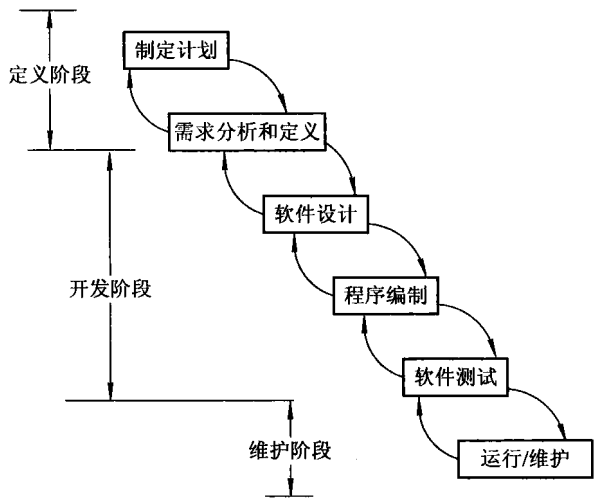

图 1-1 软件生存期模型

1.1.3 软件技术

软件是由程序、数据和文档组成的。为了开发高质量的软件，软件工程师必须遵循一

个开发策略，即软件过程模型。这种策略为软件工程的各个阶段提供了一套范形，使得工程的进展达到预期的目的。一个软件的开发，无论其规模大小，我们都需要选择一个合适的软件过程模型，这种选择基于项目和应用的性质、采用的方法、需要的控制，以及要交付的产品的特点等。

所有软件开发都可以看成是一个问题循环解决的过程，其中包括四个截然不同的阶段：状态描述、问题定义、技术开发和方案综述。其中，状态描述表示“事物当前的状态”；问题定义标识当前要解决的问题；技术开发是通过采用某种技术解决问题；方案综述是将结果提交给需要方案的人。

目前软件的过程模型有很多，如线性顺序过程模型(比较传统的软件过程模型，包括分析、设计、测试和维护)、原型实现过程模型(包括听取用户意见、建造/修改原型、用户测试运行原型、听取用户意见等)、快速应用过程原型(包括业务建模、数据建模、处理建模、应用生成和测试及反复)、演化软件过程模型(包括增量模型、螺旋模型、构建组装模型、并发开发模型)和形式化方法模型等。

软件的过程模型各具特点，但无论用何种模型开发软件，都必须采用合适的软件过程管理技术，即软件项目管理技术。

软件的最终目的是形成产品，但形成产品需要开发过程，将产品和过程完美地结合是当前软件产品化的一个重大课题。而软件技术正是为了解决该课题而形成的一门学科。

软件技术是指开发软件所需的所有技术的总称。按照软件分支学科的内容划分，软件技术包括以下几个领域：

(1) 软件工程技术。软件工程技术是软件技术的一个分支学科，主要研究软件开发全过程中的各种技术。它包括：软件开发的原则与策略，软件开发方法与软件过程模型，软件标准与软件质量的衡量，软件开发的组织与项目管理，软件版权等。

(2) 程序设计技术。程序设计技术主要包括：程序的结构与算法设计，程序设计的风格，程序设计语言，程序设计方法，程序设计自动化，程序的正确性证明，程序的变换(将低效的程序变为高效的程序称为程序的变换)等。

(3) 软件工具环境技术。软件工具环境技术主要包括：人机接口技术、软件自动生成、软件工具的集成、软件开发环境、软件的复用、逆向工程等。

(4) 系统软件技术。系统软件技术主要包括：操作系统、编译方法、分布式系统的分布处理与并行计算、并行处理技术、多媒体软件处理技术等。

(5) 数据库技术。数据库技术主要包括：数据模型、数据库与数据库管理系统、分布式数据库、面向对象的数据库技术、工程数据库、多媒体数据库等。

(6) 实时软件技术。实时软件技术主要包括：实时监控软件技术和嵌入式实时软件技术。

(7) 网络软件技术。网络软件技术主要包括：协议工程、网络管理、局域网技术、网络互连技术、智能网络等。

(8) 与实际工作相关的软件技术。与实际工作相关的软件技术主要包括：如何延长软件的使用时间和不断增强软件的性能，如何控制软件的质量，如何改变管理和配置记录，如何设计用户的在线帮助文档和图标，以及软件规模的控制、软件评估和软件开发计划的制订、软件需求的表示、软件规格说明书的确定等。

1.1.4 程序设计技术

程序设计技术的目标是提高程序的质量与生产率，最终实现程序的工业化生产。影响程序质量的因素有很多，如正确性、性能、可靠性、容错性、易用性、灵活性、可扩充性、可理解性、可维护性和复用性等。下面所介绍的程序设计技术主要是针对规模小且一个人能够独立完成的程序设计技术。程序设计的主要环节有：明确需求、设计、实现(即编码)、测试等，如图 1-2 所示。

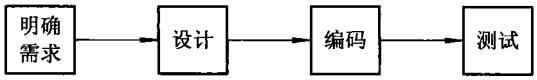

图 1-2 程序设计的主要环节

1. 明确需求

程序设计的第一步工作是明确需求，即明确待解决的问题是什么，也称为问题分析阶段。该阶段是面向“问题”的，它主要是对问题的业务活动进行分析，明确在问题的业务环境中，程序应该“做什么”，而设计、编程阶段则是面向“解答”的。明确需求就是考虑如何构造一个满足问题要求的程序。所以，在该阶段中，我们应集中考虑程序“做什么”，而尽可能少地考虑系统将怎样具体实现的问题，这些问题都应尽量推迟到设计、编码、测试阶段去解决。

为了更好地明确要求，必须回答下列问题：

(1) 谁使用程序？

(2) 使用程序的具体目的是什么？

(3) 程序需要做哪些工作，即具有的功能是什么？

2. 设计

设计是把需求转化为程序的最重要的环节。设计的优劣在根本上决定了程序的质量，它包括程序结构设计、模块设计、数据结构与算法设计和用户界面设计。

1) 结构设计

结构是程序中最本质的内容。

(1) 结构是对复杂事物的一种抽象。良好的结构是普遍适用的，它可以高效地处理多种多样的个体需求。例如，一提起“房子”，我们的大脑中马上就会出现房子的印象(而不是空洞的印象)，“房子”是人们对住宿或办公环境的一种抽象，不论是办公楼还是民房，同一类建筑物(甚至不同类的建筑物)之间都具有非常相似的结构和构造方式。

(2) 结构在一定的时间内保持稳定。程序设计最怕的就是需求变化，而在程序实际开发的过程中，“需求会发生变化”是个无法逃避的现实。人们希望在需求发生变化时，最好只对程序做些皮毛的修改，而不改动程序的结构。就如同人们对住宿的需求也会有变化一样，你可以经常改变房间的装潢和摆设，但不会在每次变动时都要去拆墙、拆柱、挖地基。因此，当需求发生变化时程序员不得不去修改程序结构，那么这个程序设计就是失败的。

良好的结构意味着普适、高效和稳定。通用的程序结构就是层次结构。层次结构表达了这么一种常识：有些事情比较复杂，我们没法一口气把事情做完，就把事情分为好几层，

一层一层地去做，高层的工作总是建立在低层的工作之上。通常层次关系主要有两种：上下级关系和顺序相邻关系。

上下级关系的层次结构，如同在军队中，上级可以命令下级，而下级不能命令上级一样，程序的上下层关系也符合上述常识，即上层子程序(模块)可以使用下层子程序(模块)的功能，而下层子程序(模块)不能够使用上层子程序(模块)的功能。

顺序相邻关系的层次结构表明：通信只能在相邻两层之间发生，信息只能被一层一层地顺序传递。这种层次结构的经典之作是计算机网络的 OSI(开放式系统互连)参考模型。为了减少结构设计的复杂性，大多数网络都按层(Layer)或级(Level)的方式组织，每一层的目的都是向它的上一层提供一定的服务，而把实现这个服务的细节对其上一层加以屏蔽。一台机器上的第 n 层与另一台机器上的第 n 层进行对话，通话的规则就是第 n 层的协议。数据不是从一台机器的第 n 层直接传送到另一台机器的第 n 层，而是发送方把数据和控制信息逐层地向下传递，最低层是物理介质，由它进行实际的通信，接收方则将收到的数据和控制信息再逐层地向上传递。

每一对相邻层之间都有接口，接口定义了下层提供的原语操作和服务。当网络设计者在决定一个网络应包含多少层，每一层应当做什么的时候，其中很重要的工作是在相邻层之间定义清晰的接口。接口使得同一层能轻易地用某一种实现(Implementation)来替换另一种完全不同的实现(如用卫星信道来代替所有的电话线)，只要新的实现能向上层提供同一组服务就可以了。

2) 模块设计

在程序结构设计完成后，就已经在宏观上明确了各个模块具有的功能以及在结构中的位置。我们习惯地从功能上划分模块，并保持“功能独立”是模块化设计的基本原则，这是因为功能独立的模块可以降低开发、测试、维护等阶段的代价。但是功能独立并不意味着模块之间保持绝对的孤立，一个系统要完成某项任务，需要各个模块相互配合才能实现，此时模块之间就要进行信息交流。

在设计一个模块时，不仅要考虑“这个模块应该提供什么样的功能”，还要考虑“这个模块应该怎样与其他模块交流信息”。

评价模块设计优劣的三个特征因素是信息隐藏、内聚与耦合和封闭—开放性。

(1) 信息隐藏。为了尽量避免某个模块的行为去干扰同一个系统中的其他模块，在设计模块时就要注意信息隐藏，让模块仅仅公开必须要让外界知道的内容，而隐藏其他内容。

模块的信息隐藏可以通过接口设计来实现。一个模块仅提供有限个接口(Interface)，执行模块的功能或与模块交流信息必须且只须通过调用公有接口来实现。

(2) 内聚与耦合。内聚(Cohesion)是一个模块内部各成分之间相关联程度的度量；耦合(Coupling)是模块之间依赖程度的度量。内聚和耦合是密切相关的，与其他模块之间存在强耦合的模块通常意味着弱内聚，而强内聚的模块通常意味着与其他模块之间存在弱耦合。模块设计追求强内聚和弱耦合。

耦合的强度依赖于以下几个因素：

① 一个模块对另一个模块的调用；

② 一个模块向另一个模块传递的数据量；

③ 一个模块施加到另一个模块的控制的多少；

④ 模块之间接口的复杂程度。

(3) 封闭—开放性。如果一个模块可以作为一个独立体被其他程序引用，则称该模块具有封闭性，如果一个模块可以被扩充，则称该模块具有开放性。

3) 数据结构与算法设计

设计高效率的程序是基于良好的数据结构与算法，而不是基于编程小技巧。一般来说，数据结构与算法就是一类数据的表示及其相关的操作(这里算法不是指数值计算的算法)。从数据表示的观点来看，存储在数组中的一个有序整数表也是一种数据结构。算法是指对数据结构施加的一些操作，例如对一个线性表进行检索、插入、删除等操作。一个算法如果能在所要求的资源限制范围内将问题解决好，则称这个算法是有效率的。算法的代价是指消耗的资源量。一般来说，代价是由关键资源(例如时间或空间)来评估的。

人们对常用的数据结构与算法的研究已经相当透彻了，可以归纳出下列设计原则：

(1) 每一种数据结构与算法都有其时间、空间的开销和收益。当面临一个新的设计问题时，设计者要彻底地掌握怎样权衡时空开销和算法有效性的方法。这就需要懂得算法分析的原理，而且还需要了解所使用的物理介质的特性(例如，数据存储在磁盘上与存储在内存中，对其就有不同的考虑)。

(2) 与开销和收益有关的是时间—空间的权衡。通常可以用更大的时间开销来换取空间的收益，反之亦然。时间—空间的权衡普遍地存在于程序设计的各个阶段之中。

(3) 程序员应该充分地了解一些常用的数据结构与算法，避免不必要的重复设计工作。

(4) 数据结构与算法为应用服务。我们在作设计的时候，必须先了解应用的需求，再寻找或设计与实际应用相匹配的数据结构和算法。

4) 用户界面设计

(1) 界面的合适性。

界面的合适性是指界面是否与程序功能相融洽。如果界面不适合于程序的功能，那么界面将毫无用处，界面美的内涵就无从谈起。界面的合适性是界面美的首要因素，它提醒设计者不要片面地追求外观漂亮而导致失真或华而不实。

(2) 界面的风格。

界面的风格有两类：一是“一致性”，二是“个性化”。

商业应用程序的界面设计注重一致性。设计者必须密切注意在相同应用领域中最流行的程序界面，必须尊重用户使用这些程序的习惯。例如，商业程序习惯于设置 F1 键为帮助热键，如果某个设计者别出心裁地让 F1 键成为程序终止的热键，那么在用户渴望得到帮助而按下 F1 键的一刹那，他的工作就此终止。

在娱乐领域运用的程序中，有个性化的界面自然比泯然于众的界面更具有吸引力。

(3) 界面的广义美。

尽管界面的美并没有增加程序的功能与性能，但是它又是必不可少的。用户使用界面时，除了直接感官获得的美感外，还有很大一部分美感是间接的，它们来源于用户的体验中，例如方便、实用等。与图形用户界面相比，命令行是最原始的界面，它难记又难看。但对于熟练的用户而言，他们乐于使用命令行以获得高效率。命令行因其具有的高效率而赢得了专业人士的喜爱，早期的 UNIX 系统就是彻头彻尾的命令系统。可以说，一切有利于人机交互的界面设计因素都具有广义美。

3. 编码

编码(Coding)的任务是为每个模块编写程序，也就是说，将设计的结果转换为用某种程序设计语言编写的程序，这个程序必须是无错的，并且要有必要的内部文档和外部文档。

程序经常需要被人阅读，而且阅读程序是程序开发工作中的一个重要环节，往往读程序的时间比编写程序的时间还要多。在这种背景下，人们开始认识到：程序实际上也是一种供人阅读的“文章”，只不过它不是用自然语言而是用程序设计语言编写的。一个逻辑上杂乱无章的程序是没有价值的，原因是它无法供人阅读，无法再利用。

为此，人们提出了程序的“可读性”，主张程序应易于阅读。所以编程的目标是编写出逻辑上正确又易于阅读的程序，这个观念现已得到所有程序设计人员的认可。

程序编码完成后，必须对程序进行调试。程序调试是指在编程环境中排除错误的过程。错误主要包括语法错误和逻辑错误。语法错误是指程序中的语句不符合程序语言规范所产生的错误，程序设计语言编程环境中的编译器负责检查程序的语法错误。逻辑错误是指程序的执行结果不正确，程序设计人员负责排除逻辑错误。

对于语法错误，编程人员可根据编译器给出的语法错误提示(错误类型和错误出现的位置)对错误进行更正。注意，语法错误的提示位置与真正出现错误的位置有可能不一致，排除该类语法错误需要编程人员经过大量的编程实践才能掌握。

需要注意的是，在程序调试时，程序员应将调试过程做比较详细的记录，以便更好地积累经验，提高以后的程序调试能力。

4. 测试

程序测试是指在受控制的条件下对程序进行操作并评价操作结果的过程。所谓控制条件，应包括正常条件与非正常条件。测试是程序的执行过程，测试的目的是发现尽可能多的缺陷。这里的缺陷是一种泛称，它可以指功能的错误，也可以指性能低下，易用性差等。测试总是先假设程序中存在缺陷，再通过执行程序来发现并最终弥补和修正缺陷。

程序测试的最终目的是保证程序的最终质量。一般来说，软件测试应由独立的产品评测中心负责，严格按照软件测试的流程，制定测试计划、测试方案、测试规范和实施测试，对测试记录进行分析，并根据回归测试情况撰写测试报告。测试是为了证明程序中有错误，而不能保证程序中没有错误。

合理地设计测试用例是保证测试过程能够最大限度地发现错误的关键。设计测试用例时，应该考虑到合法的输入和不合法的输入以及各种边界条件，特殊情况下还要制造极端状态和意外状态，比如网络异常中断、电源断电等情况。

1.2 数据结构概述

在软件的设计中涉及各种各样的数据，为了存储、组织这些数据，就需要讨论它们之间的关系，从而建立相应的数据结构，并依此实现所要求的软件功能。

1.2.1 数据结构的引入

从提出一个实际问题到计算机解出答案，需要经历下列步骤：分析阶段、设计阶段、

编码阶段和测试维护阶段等。其中，分析阶段就是要从实际问题中提取操作对象以及操作对象之间的关系。下面举例说明。

例 1-1 计算机管理图书目录问题。

要利用计算机帮助查询书目，首先必须将书目存入计算机，那么这些书目如何存放呢？我们既希望查询时间短，又要求节省空间。一个简单的办法就是建立一张表，每本书的信息只用一张卡片表示，在表中占一行，如表 1-1 所示。此时计算机操作的对象(数据元素)便是卡片，卡片之间的关系是顺序排列的。计算机对数据的操作是按某个特定要求(如给定书名)进行查询，找到表中满足要求的一行信息。由此从计算机管理图书目录问题抽象出来的模型，即是包含图书目录的表和对表进行相关的查找运算。

表 1-1 图书信息表

书名	作者	登录号	分类号	出版日期	定价(元)
Java 语言	李 晓 等	97000018	73.8792–99	1977/3/26	43.5
UNIX 系统	张 昊	96000129	73.874–126	1996/9/19	23.5
…	…	…	…	…	…

例 1-2 工厂的组织管理问题。

某工厂的组织机构如图 1-3 所示。

厂长要通过计算机了解各个科室及车间的工作和生产情况，于是，将该组织机构抽象成某个结构——树，如图 1-4 所示，它可以表示问题中各数据之间的关系。只要将数据按一定的方式存入计算机中，并对这棵树遍历，就能了解厂内的整个情况。

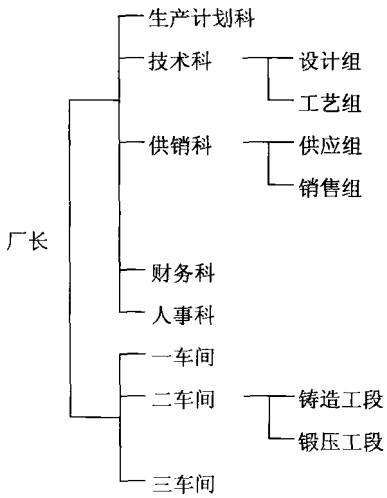

图 1-3 工厂组织机构

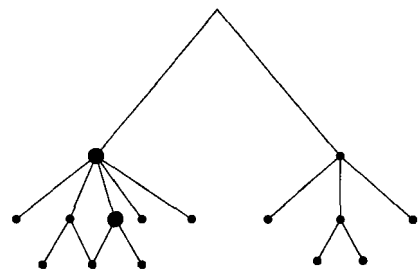

图 1-4 树形结构

例 1-3 多岔路口交通灯管理问题。

通常，在十字交叉路口只要设置红绿两色的交通灯便可保持正常的交通秩序，但是对于多岔路口，需要设置几种颜色的交通灯，既能使车辆相互间不发生碰撞，又能达到交通控制的最大流通量呢？图 1-5(a)所示是一个实际的五岔路口，我们如何设置交通灯，即最少