高职高专

高职高专工作过程·立体化创新规划教材

——计算机系列

数据结构实用教程

张居晓　葛武滇　乔正洪　朱胜强　编著

赠送电子课件

- 以培养技能型创新人才为目标，设置丰富的板块合理安排全文，突出实用性和可操作性。
- 以工作过程为导向，全面展示案例实施的全过程，提炼技术要点，即学即用面向就业。
- 以强化实际操作技能为主线，答疑解惑，解决工作实践中的常见问题。

清华大学出版社

高职高专工作过程·立体化创新规划教材——计算机系列

数据结构实用教程

张居晓　葛武滇
乔正洪　朱胜强　编　著

清华大学出版社
北京

内 容 简 介

本书依据高职高专计算机基础教育的特点，结合作者多年从事计算机教育的经验编写而成。全书共10章，主要内容包括绪论、线性表、栈和队列、串、数组及广义表、树、图、查找、排序以及综合实训。

本书以“工作场景导入”→“知识讲解”→“回到工作场景”→“工作实训营”为主线编写，以例题配合深入学习，知识讲解细致。同时，每章都有配套的实训练习，突出了实用性和操作性，另外还提供了实践中常见问题解析，能够进一步拓展学生的知识，灵活应对实际操作时会遇到的困难，使学生提高操作能力。本书结构清晰、易教易学、实例丰富、可操作性强、学以致用、注重能力的培养。

本书注重实际应用，既可作为高职高专院校计算机及相关专业的教材，也可作为各类培训班的培训教程。此外，本书也适合于有关工程技术人员、技师参考阅读。

图书在版编目(CIP)数据

数据结构实用教程/张居晓，葛武滇，乔正洪，朱胜强编著. --北京：清华大学出版社，2012
(高职高专工作过程·立体化创新规划教材——计算机系列)
ISBN 978-7-302-30215-5

Ⅰ. ①数… Ⅱ. ①张… ②葛… ③乔… ④朱… Ⅲ. ①数据结构—高等职业教育—教材 Ⅳ. ①TP311.12

中国版本图书馆 CIP 数据核字(2012)第 228434 号

责任编辑：章忆文　桑任松
封面设计：刘孝琼
版式设计：北京东方人华科技有限公司
责任校对：周剑云
责任印制：张雪娇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn,010-62791865

印 装 者：北京密云胶印厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：22　　**字　　数**：533 千字
版　　次：2012 年 11 月第 1 版　　**印　　次**：2012 年 11 月第1 次印刷
印　　数：1～3000
定　　价：37.00 元

产品编号：035922-01

丛 书 序

高等职业教育强调“以服务为宗旨，以就业为导向，走产学结合发展的道路”。能否服务于社会、促进就业和提高社会对毕业生的满意度，是衡量高等职业教育是否成功的重要指标。坚持“以服务为宗旨，以就业为导向，走产学结合发展的道路”体现了高等职业教育的本质，是其适应社会发展的必然选择。为了提高高职院校的教学质量，培养符合社会需求的高素质人才，我们计划打破传统的高职教材以学科体系为中心、讲述大量理论知识、再配以实例的编写模式，设计一套突出应用性、实践性的丛书。一方面，强调课程内容的应用性。以解决实际问题为中心，而不是以学科体系为中心；基础理论知识以应用为目的，以“必需”、“够用”为度。另一方面，强调课程的实践性。在教学过程中增加实践性环节的比重。

2009 年 5 月，我们组织全国高等职业院校的专家、教授组成了“高职高专工作过程·立体化创新规划教材”编审委员会，全面研讨人才培养方案，并结合当前高职教育的实际情况，历时近两年精心打造了这套“高职高专工作过程·立体化创新规划教材”丛书。我们希望通过对这一套全新的、突出职业素质需求的高质量教材的出版和使用，能促进技能型人才培养的发展。

本套丛书以“工作过程为导向”，强调以培养学生的职业行为能力为宗旨，以现实的职业要求为主线，选择与职业相关的教学内容组织开展教学活动和过程，使学生在学习和实践中掌握职业技能、专业知识及工作方法，从而构建属于自己的经验和知识体系，以解决工作中的实际问题。

1. 首推书目

本套丛书的首推书目如下：

- 计算机应用基础
- 办公自动化技术应用教程
- 计算机组装与维修技术
- C++语言程序设计与应用教程
- C 语言程序设计
- Java 2 程序设计与应用教程
- Visual Basic 程序设计与应用开发
- Visual C# 2008 程序设计与应用教程
- 网页设计与制作
- 计算机网络安全技术
- 计算机网络规划与设计
- 局域网组建、管理与维护实用教程
- 基于.NET 3.5 的网站项目开发实践
- Windows Server 2008 网络操作系统
- 基于项目教学的 ASP.NET(C#)程序开发设计

- SQL Server 2008 数据库技术实用教程
- 数据库应用技术实训指导教程(SQL Server 版)
- 单片机原理及应用技术
- 基于 ARM 的嵌入式系统接口技术
- 数据结构实用教程
- AutoCAD 2010 实用教程
- C# Web 数据库编程

2. 丛书特点

(1) 以项目为依托，注重能力训练。以“工作场景导入”→“知识讲解”→“回到工作场景”→“工作实训营”为主线编写，体现了以能力为本的教育模式。

(2) 内容具有较强的针对性和实用性。丛书以贴近职业岗位要求、注重职业素质培养为基础，以解决工作场景问题为中心展开内容。书中每一章都涵盖了完成工作所需的知识和具体操作过程。基础理论知识以应用为目的，以“必需”、“够用”为度，因而具有很强的针对性与实用性，可提高学生的实际操作能力。

(3) 易于学习、提高能力。通过具体案例引出问题，在掌握知识后立刻回到工作场景中解决实际问题，使学生能很快上手，提高实际操作能力；每章末的“工作实训营”板块都安排了有代表意义的实训练习，针对问题给出明确的解决步骤，阐明了解决问题的技术要点，并对工作实践中常见问题进行分析，使学生进一步提高操作能力。

(4) 示例丰富、由浅入深。书中配备了大量经过精心挑选的例题，既能帮助读者理解知识，又具有启发性。针对较难理解的问题，例子都是从简单到复杂，内容逐步深入。

3. 读者定位

本系列教材主要面向高等职业技术院校和应用型本科院校，同时也非常适合计算机培训班和编程开发人员培训、自学使用。

4. 关于作者

丛书编委会特聘执教多年且有较高学术造诣和实践经验的名师参与各册之编写。他们长期从事有关的教学和开发研究工作，积累了丰富的经验，对相应课程有较深的体会与独特的见解，本丛书凝聚了他们多年的教学经验和心血。

5. 互动交流

本丛书保持了清华大学出版社一贯严谨、科学的图书风格，但由于我国计算机应用技术教育正在蓬勃发展，要编写出满足新形势下教学需求的教材，还需要不断地努力实践。因此，我们非常欢迎全国更多的高校老师积极加入到“高职高专工作过程·立体化创新规划教材——计算机系列”编审委员会中来，推荐并参与编写有特色、有创新的教材。同时，我们真诚希望使用本丛书的教师、学生和读者朋友提出宝贵的意见和建议，使之更臻成熟。联系信箱：Book21Press@126.com。

丛书编委会

前　言

数据结构课程是我国计算机教学中较早形成和完善的一门专业基础课程，也是计算机课程体系的核心课程，是从事计算机软件开发、应用人员必备的专业基础知识。如今数据结构的教材也有很多，但经常被采用的只有几种，这些被采用的教材比较注重理论，适合计算机本科专业或者是更深层次的院校使用，却忽略了可操作性。我们苦于寻找不到一本适合高职高专院校使用的数据结构教材。于是，我们决定自己编写一本主要适合高职高专学生使用的数据结构教材，把数据结构的经典算法与实际的工作场景相结合，让同学们轻松、快速地掌握这些算法，为将来走上工作岗位打下坚实的基础。

本书每章通过导入工作场景引出问题，然后详细讲解用来解决问题的知识点，同学们开始就带着问题来学习本章内容，学完本章内容就会发现工作场景的问题变得相当简单，从而掌握本章的重点内容。本书主要讲述了线性表、串、树、图等几种数据结构，以及查找、排序中的经典算法。最后一章则是对前面各章的知识点的应用，在学习完前面 9 章内容后，读者可以尝试自己编写程序。

本书具有以下特点。

(1) 结构清晰、模式合理。以“工作场景导入”→“知识讲解”→“回到工作场景”→“工作实训营”为主线编写，以这种新颖的模式合理安排全文。

(2) 针对性强、实用性强。学生们最需要的是提高实际操作能力，本书正是以解决工作场景为中心展开内容，每一章中都涵盖了完成工作所需的知识和具体操作过程，因而具有很强的针对性与实用性。

(3) 上手快、易教学。通过具体案例引出问题，在掌握知识后立刻回到工作场景解决问题，使学生很快上手；以教与学的实际需要取材谋篇，方便教师教学。

(4) 安排实训，提高能力。除绪论和综合实训外，每一章都安排了“工作实训营”板块，针对问题给出明确的解决步骤，并对工作实践中常见问题进行分析，使学生进一步提高操作能力。

本书组织精练，例题简单，易于理解，并配合了各种类型的练习和实践操作题，便于学生进一步掌握知识和提高操作能力。对于数据结构中重要和较难理解、容易出错的内容，书中均加以特别强调和说明。本书后附有习题的参考答案，可供读者学习时参考。

本书由张居晓、葛武滇、乔正洪、朱胜强编著。参与本书的编写和资料整理的还有姜传金、汪大锋、何光明、王珊珊、吴涛涛、陈海燕、周海霞、毛幸甜、卢振侠、江梅、刘宇松等，在此表示感谢。

本书可作为高职高专、成人高校及应用型本科计算机类专业的教材，也可作为各类工程技术人员的参考书，亦可供计算机爱好者自学使用。

由于编者水平有限，书中难免有不足和疏漏之处，恳请读者批评指正。

编　者

目　录

第1章

绪论

本章要点

- 数据结构的基本概念。
- 算法的基本概念。

技能目标

- 掌握数据结构和算法的基本概念。
- 学会如何进行算法分析。

1.1 什么是数据结构

1.1.1 数据结构的产生与发展

数学模型是程序设计中十分必要的一步，它可以解决由需求产生的实际问题。但是程序设计并不是纯粹的数值计算问题，它还要涉及诸如数据的存储、检索、遍历等问题。

表 1-1 所示为一份学生成绩单。我们可以把这张表看成由记录组成，表中的每个记录由 5 个数据项组成。记录之间是一种前后有序的线性关系。账目管理、图书馆书籍管理等都与成绩表具有类似的数学模型。这类数学模型可以称为线性的数据结构，对这种数据结构可以进行的操作有插入、删除、查询数据等。

表 1-1 学生成绩单

姓 名	网络安全	Visual Basic	计算机理论基础	数据库
张三	80	90	70	78
李四	85	75	80	60
王五	62	86	95	98
赵六	70	74	85	80

在体育比赛中，一般都是通过逐层选拔，产生最后的冠军。因此可以把最初的参赛者看成树叶，冠军看成一个树根，其他各级看成是树的枝干，这样就构成一个树形结构，如图 1-1 所示。

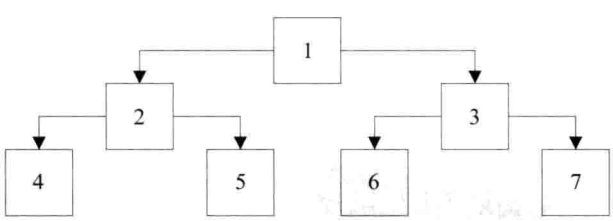


图 1-1 树状图

如图 1-1 所示，树中的结点之间不再是顺序的关系，而是分层、分叉的非线性结构。对于树的主要操作有插入、删除、遍历和查找数据等。

还有一类问题可以用通信网络来模拟。在通信网络中，如果把路由器看成若干顶点，把路由器之间的链路看成边，则它们可以构成一个网状图，如图 1-2 所示。

图 1-2 中顶点之间的关系纵横交错、错综复杂，这种关系称为图形结构。在实际应用中，如果一个数据包要在图中传递，则需要仔细考虑它的传送路径，以使传送时间最短。网络图的存储、管理，结点的插入、删除等已不再是一个单纯的数值计算问题，而是一个非数值的信息处理问题。

综上所述，描述非数值问题的数学模型是诸如树、线性表、图之类的结构。而数据结

构则是研究这些结构的一门学科。因此，它的重要性是不言而喻的。

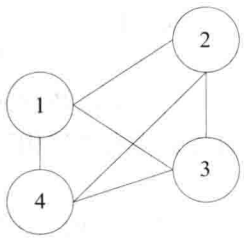


图 1-2　网状图

1968 年，Knuth 教授开创了数据结构的最初体系。数据结构主要研究一类普通数据的表示及其相关的运算操作，它是一门研究怎样合理地组织数据、建立合适的数据结构、提高计算机执行程序所用的时间和空间效率的学科。在计算机科学中，“数据结构”是一门综合性的专业基础课，不仅涉及计算机硬件，而且和计算机软件的研究也有着密切的关系。无论是编译程序还是操作系统，都涉及数据元素在存储器中的分配问题。在研究信息检索时必须考虑如何组织数据，使得查找和存取数据元素更为方便。因此，可以认为“数据结构”是介于数学、计算机硬件和软件三者之间的一门核心课程。

1.1.2　数据和数据结构

数据(data)是信息的载体，是描述客观事物的数、字符以及所有能输入到计算机中并被计算机程序识别和处理的符号的集合。它是计算机程序加工的“原科”。数据分为两类：数值型数据(主要用于数学计算等)和非数值型数据(文字、图形、图像、音频和视频等)。

数据元素(data element)是数据的基本单位。在不同的条件下，数据元素又可称为元素、结点、顶点、记录等，如八皇后问题中状态树的一个状态。一个数据元素可由不可分割的若干个数据项(data item)组成。例如，在图书馆管理系统中，通常将每本书的信息存储在一个表中，这样一本书的书目信息就为一个数据元素。而书目信息中的每一项，如书名、作者等就称为数据项。数据项是数据不可分割的最小单位。

数据对象(data object)是性质相同的数据元素的集合，是数据的一个子集。例如，整数的数据对象是集合 $\mathbf{Z}=\{0, \pm1, \pm2, \cdots\}$，字母字符的数据对象是集合 $C=\{\text{'A'}, \text{'B'}, \cdots, \text{'Z'}\}$。

数据结构(data structure)是指相互之间存在着一种或多种关系的数据元素的集合。在任何问题中，数据元素之间总是存在联系的。把某一数据对象及该数据对象中所有数据成员之间的关系组成的实体叫做数据结构。

数据结构的形式定义如下：

$$\text{Data\_Structure} = (D, R)$$

其中，D 是数据元素的有限集，R 是 D 上关系的有限集。

1.1.3　数据的逻辑结构和存储结构

研究数据结构，是指研究数据的逻辑结构和存储结构。

1. 数据的逻辑结构

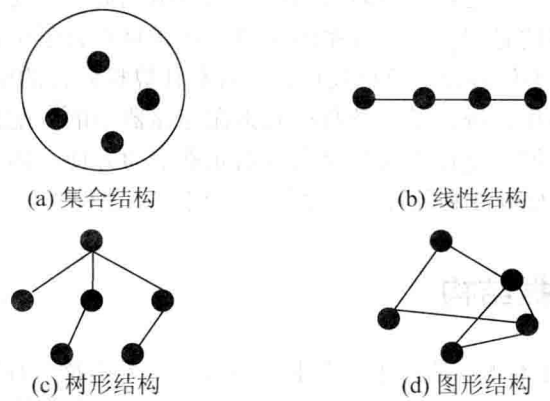
数据逻辑结构是相互之间存在一种或多种特定关系的数据元素的集合。它是指数据元素之间的相互关系，即数据的组织形式。它并不涉及数据元素在计算机存储设备中的具体存储方式，是独立于计算机的。在任何问题中，数据元素都不是孤立存在的，而是它们之间存在着某种联系，这种元素之间的关系称为“结构”。根据关系的不同特性，通常有 4 种结构。

(1) 集合结构：数据元素之间存在着“属于同一个集合”的关系，如图 1-3(a)所示。

(2) 线性结构：数据元素之间存在着“一对一”的关系，如图 1-3(b)所示。

(3) 树形结构：数据元素之间存在着“一对多”的关系，如图 1-3(c)所示。

(4) 图形结构：数据元素之间存在着“多对多”的关系，如图 1-3(d)所示。

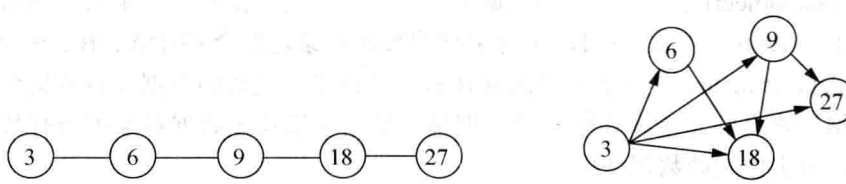


(a) 集合结构　(b) 线性结构　(c) 树形结构　(d) 图形结构

图 1-3　4 类基本数据结构示意图

【实例 1-1】定义集合 $D = \{3, 6, 9, 18, 27\}$的数据结构。

$DS_1=(D, R_1)$，其中 R_1 定义为 D 上的“>”(大于)关系，则数据结构 DS_1 可以表示为如图 1-4(a)所示的形式。$DS_2=(D, R_2)$，其中 R_2 定义为 D 上的“整除”关系，则 $R_2=\{(3, 3), (3, 6), (3, 9), (3, 18), (3, 27), (6, 18), (9, 18), (9, 27)\}$，数据结构 DS_2 可以表示为如图 1-4(b)所示的形式。



(a) 数据结构DS_1　(b) 数据结构DS_2

图 1-4　集合 D 上定义的两个数据结构

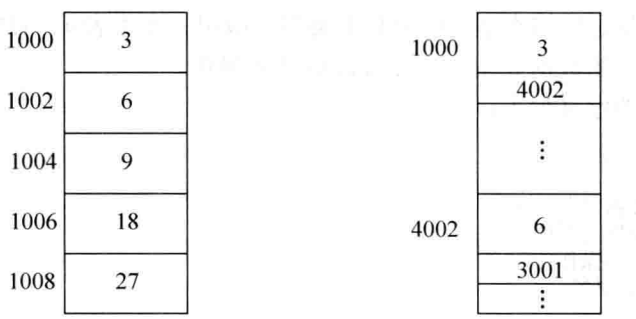
从上面这个例子可以看出，即使是由相同元素构成的集合，只要定义的关系不同，也不是同一数据结构。数据结构不仅描述了结构中的元素，还描述了这些元素之间的关系。数据结构的定义是对操作对象的一种数学描述，结构中定义的关系是数据元素之间的逻辑关系。

2. 数据的存储结构

讨论数据结构的最终目的是在计算机中实现它，因此，必须要知道数据在计算机内的存储表示。所以有必要了解计算机的实际存储结构，也称作物理结构。数据的存储结构是逻辑结构在计算机存储设备中的映像。

常见的存储结构有顺序存储结构和链式存储结构两种。顺序存储结构的特点是用物理地址相邻来表示数据元素在逻辑上的相邻关系；链式存储结构的特点是逻辑上相邻的数据元素在存储地址上不一定相邻，元素之间逻辑上的相邻关系通过指针来描述，常用它来描述树状结构和图状结构在计算机内的存储。除此之外，还有两种存储方式：索引方式和散列方式。索引存储方式：除数据元素存储在一地址连续的内存空间外，尚需建立一个索引表，索引项指示存储结点的存储位置或存储区间端点；散列存储方式：利用哈希函数和解决冲突的方法，将关键字散列在连续的有限的地址空间内，并将哈希函数的值解释成关键字所在元素的存储地址。

逻辑结构和存储结构是描述数据结构的两个方面。任何一个算法的设计取决于选定的逻辑结构，而算法的实现取决于其所依托的存储结构。图 1-5 给出了图 1-4 中数据结构 DS_1 的不同存储方式。



(a) 顺序存储结构　　(b) 链式存储结构

图 1-5　数据结构 DS_1 存储结构示意图

1.1.4　数据类型

1. 数据类型

数据类型(data type)是和数据结构密切相关的一个概念，用来刻画操作对象的属性。从软件的角度讲，在高级语言(C、C++和 Java)中，每一个变量或常量都拥有确定的数据类型，如 int、float 和 char 型等，熟悉 C、C++的人应该不会对这些类型感到陌生。不同的数据类型拥有不同的取值范围和允许的操作。从硬件的角度讲，数据类型涉及具体存储单位，比如 int 型占用两个字节，float 型占用 4 个字节等。由此可以帮助程序开发人员了解内存的使用情况。

以“值”不同的特性来讲，高级程序语言将数据类型分为两类：原子类型和结构类型。原子类型是指不可再分的类型，比如 C 语言中的 int、float、char 型等。而结构类型的值可由若干原子类型的值按照某种结构组成，因此是可分的，最典型的例子是 C 语言中的结构体。

2. 抽象数据类型

抽象数据类型指一个数学模型以及定义在该模型上的一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关。对一个抽象数据类型进行定义时，必须给出它的名字及各运算的运算符名，即函数名，并且规定这些函数的参数性质。一旦定义了一个抽象数据类型及具体实现，程序设计中就可以像使用基本数据类型那样，十分方便地使用抽象数据类型。由此可见，抽象数据类型和数据类型实质上是一个概念。但是它的范畴更加广泛，它不再局限于已有的数据类型，如 int、float 等，还包括用户在设计软件时自己定义的数据类型。

抽象数据类型的描述包括抽象数据类型的名称、数据的集合、数据之间的关系和操作的集合等方面的描述。抽象数据类型的设计者根据这些描述给出操作的具体实现，抽象数据类型的使用者依据这些描述使用抽象数据类型。

抽象数据类型可以分为如下 3 种类型。

(1) 原子类型(atomic data type)：其值是不可分的。

(2) 固定聚合类型(fixed-aggregate data type)：其值由确定数目的成分按某种结构组成。

(3) 可变聚合类型(variable-aggregate data type)：其值由不确定数目的成分构成。

其中，后两种都是结构类型。抽象数据类型一般用三元组表示：(D, S, P)，其中 D 表示数据对象、S 是 D 上的数据关系，P 表示 D 的基本操作。

抽象数据类型的定义如下：

```
ADT 数据抽象类型名
{
   数据对象：数据对象的定义
   数据关系：数据关系的定义
   基本操作：基本操作的定义
}ADT 数据抽象类型名
```

抽象数据类型的三元组定义如下：

```
ADT Structure
{
   数据对象：D={e1,e2,e3}
   数据关系：R={<e1,e2>,<e2,e3>}
   基本操作：
        Init Structure (&S, v1, v2, v3)
           操作结果：构造三元组 S，元素 e1,e2,e3 分别被赋予参数 v1,v2,v3 的值。
        DestroyList(&S)
           操作结果：销毁三元组 S。
        Get(S, i, &e)
           初始条件：三元组 S 已存在。
           操作结果：用 e 返回 S 的第 i 元的值。
        Put(&S, i, e)
           初始条件：三元组 S 已存在。
           操作结果：改变 S 的第 i 元的值为 e。
        IsAscending(S)
           初始条件：三元组 S 已存在。
           操作结果：如果 S 的元素按升序排列，返回 1，否则返回 0。
        IsDescending(S)
           初始条件：线性表 S 存在。
           操作结果：如果 S 的元素按降序排列，返回 1，否则返回 0。
        Max(S, &e)
           初始条件：三元组 S 已存在。
```

```
        操作结果：用 e 返回 S 中最大的数值。
    Min(S, &e)
        初始条件：三元组 S 已存在。
        操作结果：用 e 返回 S 中最小的数值。
} ADT Structure
```

多型数据类型(polymorphic data type)：指其值的成分不确定的数据类型。上述中的 e1、e2 和 e3 可以是 int 型、float 型、char 型甚至由多种成分构成。但是，不论其元素具有何种特性，元素之间的关系和基本操作都相同。

1.2　算法与算法分析

1.2.1　算法

算法(algorithm)是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每条指令表示一个或多个操作。它有以下 5 个特性。

(1) 有穷性：一个算法必须总是在执行有限步骤之后结束，其每一步都在可接受的时间内完成。

(2) 确定性：算法中的每一条指令都必须有明确的含义，并且在任何条件下，算法都只有唯一的一条执行路径，即对于相同的输入只能得到相同的输出。

(3) 可行性：一个算法是可行的，即算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现。

(4) 输入：根据实际问题的需要，一个算法有一个或者多个外部输入，也可以没有，这些输入取自于某个特定的对象集合。

(5) 输出：算法执行完成之后，必须至少有一个结果输出，这些输出和输入之间有着某种联系。

1.2.2　算法设计的目标

一个好的算法应该满足以下目标。

(1) 正确性：这是算法最基本的要求。但是所谓“正确”在算法设计的过程中可以分为 4 个层次：依据算法所编制的程序中不含语法错误；程序对于几组输入数据能够得到满足规格说明要求的结果；程序对于经过精心挑选、较为苛刻的几组输入数据能够得到满足规格说明要求的结果；程序对于所有符合要求的输入数据都能得到满足规格说明要求的数据。在大型软件的开发过程中，通常以第三个层次作为衡量正确性的标准。

(2) 可读性：算法主要是为了方便人的阅读与理解，其次才是计算机的执行。由于在解决实际问题时，通常需要若干人共同完成，因此，可读性就显得尤为重要。在设计算法时，一般使用有意义的标识符给变量和函数起名。其次，可以在程序中加入注释。

(3) 健壮性：当输入不合法的数据时，程序也能做出适当的处理，而不至于导致莫名其妙的结果出现。

(4) 高效率：算法执行时间短。

(5) 低存储：依据算法编制的程序运行时所需的空间较少。

对于一个实际系统来说，前三项易于实现；而后两点在评价算法的优劣的问题上更加重要。主要通过算法设计的时间复杂度和空间复杂度这两个方面对算法进行度量。

1.2.3 算法设计的时间复杂度

算法的执行时间需要用依据算法所编制的程序在计算机上运行时所消耗的时间来衡量。衡量一个程序执行的时间通常有两种方法：事后统计法和事先分析法。这两种方法都是以绝对的时间单位来衡量的。由于计算机硬件条件、编译程序、编程语言的不同，应该抛弃特定的软硬件配置，而使用指令的执行次数作为算法的时间度量单位。在这种情况下，算法的时间和问题的规模 n 有关系，当 n 趋向于无穷大时的时间量级就称为时间复杂度，记作 $T(n)=O(f(n))$，即 $T(n)$是 $f(n)$的同阶无穷大。

在这种描述中使用的基本参数是 n，即问题实例的规模，把复杂性或运行时间表达为 n 的函数。这里的“O”表示量级，比如说“折半查找算法是 $O(\log_2 n)$的”，也就是说它需要通过 $\log_2 n$ 量级的步骤去检索一个规模为 n 的数组。记法 $f(n)$表示当 n 增大时，运行时间最多将以正比于 $f(n)$的速度增长。

这种渐近估计对算法的理论分析和大致比较是非常有价值的，但在实践中细节也可能造成差异。例如，一个低附加代价 $O(n^3)$算法在 n 较小的情况下可能比一个高附加代价的 $O(n\log_2 n)$算法运行得更快。当然，随着 n 足够大以后，具有较慢上升函数的算法必然工作得更快。

一个算法由控制结构(顺序、分支和循环)和基本操作构成。所谓基本操作是指数据类型固有的操作，比如加、减、乘、除等。一般的做法是以基本操作重复的次数来作为时间复杂度的度量。

【实例 1-2】分析以下算法的时间复杂度。

```
(1) t=1;                          /*  1次  */
for(int i=0; i<n; i++)            /*  i=0: 1次; i<n: n次; i++: n次  */
  for(int j=0; j<n; j++ )         /*  j=0: 1次; j<n: n次; j++: n次  */
    t=t*t;                        /*  n*n次;  */
(2) t=1;                          /*  1次  */
for(int i=0; i<n; i++)            /*  i=0: 1次; i<n: n次; i++: n次  */
      t=t*t;                      /*  n次  */
(3) t=1;                          /*  1次  */
t=t*t;                            /*  t*t:  1次  */
```

分析：

算法(1)共执行了 n^2+4n+3 次，其中，最高阶为 n^2 次，时间复杂度为 $O(n^2)$。

算法(2)共执行了 $3n+2$ 次，其中，最高阶为 n 次，时间复杂度为 $O(n)$。

算法(3)共执行了 2 次，最高阶为 1 次，时间复杂度为 $O(1)$。

由此可知，时间复杂度是由算法的最高阶数决定。时间复杂度只是描述了一个数量级的概念，并不是精确的结果。这三个时间复杂度分别被称为平方阶、线性阶和常量阶。

实际上，算法的时间量级有多种形式，见表 1-2。