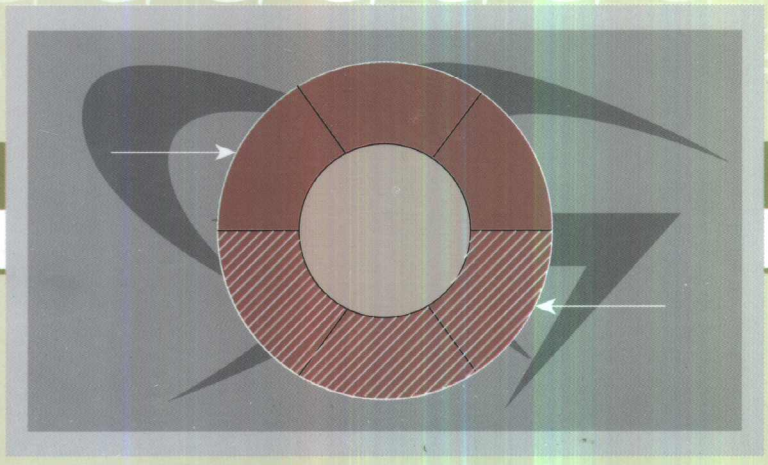
高职高专计算机系列教材

双高系列教材

实用数据结构

佟 维 谢爽爽 编著



科学出版社
www.sciencep.com

高职高专计算机系列教材

实用数据结构

佟　维　谢爽爽　编著

科学出版社
北　京

内 容 简 介

本书是为高职、高专计算机相关专业编写的教材，选材基本上覆盖了数据结构的主要内容。考虑到高职、高专的特点，本教材对各种数据结构和有关的算法多以实例来讲解，叙述上较为通俗、详尽。

本书特别注重实际应用，在每一章介绍一种数据结构后都给出相关的应用实例，并配有大量的例题、习题。书中对线性表、链表、树和图等典型数据结构，以及排序和查找两项技术都给出了相关的实验，每个实验中除了给出有关的实验目的、实验内容和实验要求外，还给出了大部分参考程序，并对每章后的习题和算法设计题全部给出答案，以供学生学习和参考。

本教材可供高职、高专计算机相关专业使用，也可供从事计算机应用工作的技术人员参考或用作培训教材。

图书在版编目（CIP）数据

实用数据结构/佟维，谢爽爽编著. —北京：科学出版社，2003

（高职高专计算机系列教材）

ISBN 7-03-010930-9

Ⅰ.实… Ⅱ.①佟…②谢… Ⅲ.数据结构—高等学校：技术学校—教材 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字（2002）第 084905 号

责任编辑：鞠丽娜 丁 波/责任校对：赵惠玲

责任印制：吕春珉 /封面设计：王 浩

科学出版社 出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

新蕾印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2003 年 1 月第 一 版 开本：B5(720×1000)

2003 年 1 月第一次印刷 印张：15

印数：1—5 000 字数：286 000

定价：21.00 元

（如有印装质量问题，我社负责调换〈环伟〉）

前　言

“数据结构”是计算机应用及相关专业的重要专业基础课，它不仅是“操作系统”、“数据库”等课程的基础，也是我们编写计算机程序的重要基础，是学习计算机软件设计必不可少的基本知识。

本书是为高职、高专计算机应用及相关专业编写的教材，选材基本上覆盖了数据结构的主要内容。全书分两篇，第一篇主要讲述数据结构的基础知识，共分为8章。其中，第1章介绍数据结构与算法的基本概念及算法分析的基本方法；第2～4章分别讲述数组与线性表、链表以及串等基本数据结构和相关算法，并给出应用实例；第5、6章介绍两种非线性数据结构——树和图；第7、8章介绍数据处理中广泛应用的各种排序和查找方法。第二篇为上机实验指导与习题解答，在上机实验指导中，对线性表、链表、树和图等典型数据结构，以及排序和查找两项技术都给出了相关的实验。每个实验中除了给出有关的实验目的、实验内容和实验要求外，还给出大部分参考程序，以供学生学习和参考。在习题解答中，对全书中各章的基础知识题和算法设计题均给出了答案。

由于数据结构及相关的算法较抽象，对于高职、高专学生及一般初学者理解和掌握其内容比较困难，因此，本书在编写上，对各种数据结构和有关的算法多以实例来讲解，叙述上也较为通俗、详尽。每一章之后，都给出了相关的应用实例，并配有大量的例题，可使学生加深对基本概念的理解，提高分析问题和解决问题的能力。本书适合高职、高专计算机相关专业学生使用，也可用作各种计算机培训班的教材。

本书由佟维主编，书中第1、2、3、5、7、9章由佟维执笔，第4、6、8、10章由谢爽爽执笔。本书由东北大学信息学院王宝库教授主审，他在百忙之中对编写大纲及书稿做了全面、仔细的审定，并提出了宝贵的修改意见，在此表示衷心的感谢。

在本书编写过程中，得到高职高专计算机系列教材编委会和科学出版社有关同志的关心和大力支持，谨此一并表示谢意。

由于编者水平有限，加之时间仓促，书中难免有错误和不妥之处，希望读者不吝指正。

作　者

目　录

第一篇　数据结构基础

第二篇 上机指导与习题解答

第一篇

数据结构基础

本篇基本上覆盖了数据结构的主要内容，介绍了各种数据结构知识和有关的算法。由于数据结构及相关的算法较抽象，对于高职、高专学生及一般初学者理解和掌握其内容比较困难。因此，本篇在编写上注重高职、高专的特点，对各种数据结构和有关的算法多以实例来讲解，叙述上也注意较为通俗、详尽。该篇在编写上也特别注重实际应用，在每一章介绍了一种数据结构后，配有一定的例题，并给出了相关的应用实例，每章后面还给出了大量的基础知识题和算法设计题，供学生复习和练习，以使学生加深对基本概念的理解，提高分析问题和解决问题的能力。

第1章 绪论

知识点

- 数据结构中常用的基本概念和术语
- 算法描述和分析方法

难点

- 算法复杂性的分析方法

要求

- 了解数据的逻辑结构和物理结构，算法的基本概念，以及它们对于程序设计的重要性
- 掌握算法复杂性的概念及分析方法

随着计算机技术的飞速发展，计算机应用领域迅速扩展，计算机应用已不再局限于科学计算，而广泛应用于数据处理、信息管理等非数值计算的各个方面。数据结构主要研究非数值应用问题中数据之间的逻辑关系和对数据的运算，同时还研究如何将具有逻辑关系的数据按一定的存储方式存放在计算机内。本章介绍贯穿全书的基本概念和术语，以及算法描述和分析的方法。

1.1 基本概念

数据(data)就是一切能够由计算机接收和处理的对象。随着计算机技术的发展，数据这一概念的含义越来越广泛。不仅整数、实数、复数等是数据，字符、表格、声音、图形、图像等也都能够由计算机接收和处理，它们也是数据。

数据元素(data element)是数据的基本单位，在程序中作为一个整体加以考虑和处理。换句话说，数据元素被当作运算的基本单位，并且通常具有完整确定的实际意义，在数据结构中，根据需要，数据元素又被称为元素、顶点或记录。

数据项(data item)是数据的不可分割的最小单位，在有些场合下，数据项又称为字段或域。例如，将一个学生的自然情况信息作为一个数据元素，而学生信息中的每一项如学号、姓名、出生年月等为一个数据项。

数据结构(data structure)指的是数据之间的相互关系，即数据的组织形式。研究数据结构是指研究数据的逻辑结构和物理结构。数据的逻辑结构是指数据元素之间的逻辑关系，如数组中各元素的先后次序关系等。数据的物理结构是研究数据元素在计算机存储器中如何存储的，如一个二维数组在内存中是如何安排的等。所以，数据的物理结构又称为存储结构。

研究数据结构，除了要定义数据元素之间的相互关系，更重要的是要定义一组有关数据元素的运算。例如，对于数组这种数据结构，除了要定义每个数组元素的类型和元素个数以外，还要定义有关数组的一些运算，如向数组中插入新元素或删除某个数组元素等。

算法(algorithm)是对特定问题求解步骤的一种描述。它是一个有穷的规则序列，这些规则决定了解决某一特定问题的一系列运算。输入特定问题，计算机就会依照这些规则进行计算和处理，经过有限的计算步骤后能得到一定的输出。

进行计算机程序设计，掌握数据结构知识和算法设计的有关知识是十分重要的。因此，有人称数据结构和算法是计算机程序设计的两个支柱，瑞士的著名计算机科学家 N·沃斯教授更提出了“算法+数据结构=程序”的著名公式，并以此公式作为他所著的一本书的书名，由此可以看出算法和数据结构二者之间的密切关系。本书以讲述数据结构为主，也结合介绍与各种数据结构相关的一些算法。

1.2 算法的描述

算法需要用一种语言来描述，通常可有各种描述方法来满足不同的需要。例如，可以直接用某种高级程序设计语言(如 Pascal，C 语言等)来描述算法，也可以用框图、类高级语言或自然语言来描述算法。类高级语言与程序设计语言相仿，它包括高级语言中的基本语言成分，它比程序设计语言简单，但增加了语言的描述功能的语言。用类高级语言描述的算法，虽不能在计算机上直接运行，但简明清晰，不拘泥于高级程序设计语言的细节，容易编写和阅读。因此，用这种描述语言来描述算法是比较合适的，在数据结构的算法描述中被广泛应用。

本书将采用类 C 语言描述算法，类 C 语言是标准 C 语言的简化。它比较简洁，便于突出算法设计的主要方面，而不是细节方面，如果需要也不难改写为 C 语言或其他高级语言程序。

下面仅对类 C 语言与标准 C 语言的主要区别作简单说明。

(1) 所有算法都以如下所示的函数形式表示：

```
函数类型　函数名(参数表)
{
```

语句序列

}

类 C 语言的形参书写比标准 C 语言简单，如 int xyz(int a,int b,int c)可以简单写成 int xyz (int a,b,c)。

(2) 局部量的说明可以省略，必要时对其作用给予注释。

(3) 不含 go to 语句，增加一个出错处理语句 error(字符串)，其功能是终止算法的执行并给出表示出错信息的字符串。

(4) 输入/输出语句有：

输入语句　scanf([格式串]，变量 1，…，变量 N)；

输出语句　printf([格式串]，变量 1，…，变量 N)；

通常省略格式串。

1.3　算法的评价

一般地，对于同一种问题可往往有求解它的多种不同的算法，这就产生了如何评价这些算法的问题。通过对算法的评价，一方面可以从解决同一问题的不同算法中区分相对优劣，选出较为适用的一种，另一方面也有助于设计人员考虑对现有算法进行改进或设计出新的算法。

1.3.1　评价算法的一般原则

一般来说，设计一个“好”的算法主要应考虑以下几个方面。

- 正确性：算法应能正确地实现处理要求。即该算法在合理的输入数据下，能在有限的时间内得出正确的结果。分析算法的正确性，对于较复杂的问题，可以将其算法分成一些局部的过程来分析，只有每个局部过程都是正确的，才能保证整个算法的正确性。
- 易读性：算法易读性有助于对算法的理解，便于纠正和扩充。
- 简单性：算法的简单性使得证明其正确性比较容易，对算法进行修改也比较方便。
- 高效率：即达到所需的时、空性能。一个算法的时、空性能是指该算法的时间性能和空间性能。解决同一问题如果有多个算法，执行时间短的算法时间效率高，而占用存储容量少的算法空间效率高。

1.3.2　算法复杂性的分析

算法的复杂性一般包括时间复杂性(所需运算时间)和空间复杂性(所占存储空

间)。随着计算机硬件技术的飞速发展，扩展内存与外存的容量已并不困难，所以我们现在研究算法的复杂性，重点是指时间复杂性。

一个算法所需的运算时间通常与所解决问题的规模大小有关。我们用 n 作为表示问题规模的量。例如，树的问题中 n 是树的顶点数；排序问题中 n 为需排序元素的个数等。我们经常把算法运行所需的时间 T 表示为 n 的函数，记为 T(n)。评价一个算法时，增长率的概念是非常重要的，不同的 T(n)算法，当 n 增长时，运算时间增长的快慢很不相同。凡是 T(n)为 n 的对数函数、线形函数或多项式函数(包括幂函数，幂函数是多项式的特例)，我们称这种算法为“好”的算法，而 T(n)为指数函数或阶乘函数的算法，因 T(n)随 n 的增长而增长得太快，当 n 较大时是不适用的，故称为“坏”的算法。

一个算法所消耗的时间，应该是该算法中每条语句的执行时间之和，而每条语句的执行时间是该语句的执行次数(也称为频度，frequency count)与该语句执行一次所需时间的乘积。但是，当算法转换为程序之后，每条语句执行一次所需的时间取决于机器的指令性能、速度以及编译所产生代码的质量，这是很难确定的。因此，我们在研究时间复杂性时，假设每条语句执行一次所需的时间均是单位时间。一个算法所需的执行时间就是该算法中所有语句执行次数之和。

讨论时间复杂性时，我们往往研究所谓的“渐进时间复杂性”，即当 n 逐渐增大时 T(n)的极限情况。一般把算法的渐进复杂性简称为时间复杂性。为了分析方便，时间复杂性常用数量级的形式来表示，记作：T(n)=O(f(n))。其中，大写字母 O 为 Order(数量级)的字头，f(n)为函数形式，如 $T(n)=O(n^2)$。用数量级的形式表示 T(n)，当 T(n)为多项式时，可只取其最高次幂项，且它的系数也可略去不写，因为当 n 大到一定程度时，总是最高次幂占 T(n)的主要部分。例如：

$$T(n)=8n^3+150n+32$$

则

$$T(n)=(n^3)$$

一般地，对于足够大的 n，常用的时间复杂性存在以下顺序

$O(1)<O(\log n)<O(n)<O(n*\log n)<O(n^2)<O(n^3)\cdots<O(2^n)<O(3^n)<\cdots<O(n!)$

其中，O(1)为常数数量级，即算法的时间复杂性与输入规模 n 无关。

一个算法的运行时间除与问题的规模 n 有关外，往往还与具体输入的数据有关。例如，在一个一维数组中欲查找某一数据，若采用从头到尾顺序查找的算法，当碰巧待查找的数据在数组的第一个单元时，只需比较一次即可查找到，而最坏的情况，当待查找的数据在数组的最后一个单元时，则比较 n 次才可查找到。因此，在分析算法的时间复杂性时，通常用以下两种方法来确定一个算法的运算时间：一个是平均时间复杂性，即研究同样的 n 值时各种可能的输入，取它们运算时间的平均值；另一个是最坏时间复杂性，即研究各种输入中运算最慢的一种情况下的运算时间。采用取平均的办法似乎较好，但在很多情况下，各种输入数据

出现的概率很难确定，分析起来比较困难，有时甚至是不可能做到的，所以一般情况下，我们是研究最坏情况下的时间复杂性。本书讨论的时间复杂性，除特别指明外，均指最坏情况下的时间复杂性。

1.4 应用举例及分析

例 1.1 计算下面交换 i 和 j 内容程序段的时间复杂性。

```
temp=i;
i=j;
j=temp;
```

以上 3 条单个语句均执行 1 次，该程序段的执行时间是一个与问题 n 无关的常数，因此，算法的时间复杂度为常数阶，记作 T(n)=O(1)。

例 1.2 计算下面求累加和程序段的时间复杂性。

```
① sum=0;
② for(i=1;i<=n;i++)
③ for(j=1;j<=n;j++)
④ sum++;
```

其中，语句①执行 1 次。语句②的循环变量 i 要增加到 n，故它执行 n 次。语句③作为语句②的循环体内的语句应该执行 n 次，所以语句③执行是 n^2 次。而语句④作为语句③的循环体也要执行 n^2 次。所以，该程序段所有语句执行的次数为：

$T(n)=2n^2+n+1$

故其时间复杂性为：

$T(n)=O(n^2)$

小 结

本章介绍了贯穿全书的基本概念和基本思想。数据是一切能够由计算机接收和处理的对象。“数据结构”主要是描述数据中各种元素间的相互关系，往往是给出有关这些数据元素的一组运算。数据结构包括数据的逻辑结构和物理结构。数据的逻辑结构指数据元素之间的逻辑关系，数据的物理结构指数据元素在计算机存储器中的表示和安排。一般前者称为数据结构，而后者称为存储结构。

算法是个有穷的规则序列，这些规则决定了解决某一特定问题的一系列运算。本书采用一种类 C 语言来描述算法，必须熟悉这种语言，并能将本书中讲述的算法用 C 语言编出程序来。

评价一个算法主要是研究算法的时间复杂性。为了分析方便，算法复杂性通常用数量级的形式来表示，当 T(n)为多项式时，只需取其最高幂项，且此项的系数也可略去不写。

习　题

一、解释下列名词

数据、数据项、数据元素、数据结构、数据逻辑结构、数据物理结构、算法、算法的时间复杂性

二、回答下列问题

(1) 算法分析的目的是什么？

(2) 什么是算法的最坏和平均时间复杂性？

三、分析下列算法的时间复杂性

(1)
```
sum=0;
   for (I=i;I<=n;I++)
{
  sum=sum+i;
}
```

(2)
```
I=1;
While(I<=n)
    I=I*10;
```

(3)
```
sum=0;
for(I=0;I<n;I++)
   for (j=0;j<n;j++)
sum=sum+Array[i][j];
```

第2章　数组与线性表

知识点

- 数组的基本特点及寻址方式
- 线性数据结构的基本特征、基本运算
- 堆栈的定义和基本运算
- 队列的定义和基本运算
- 循环队列的特征、运算以及判断溢出的条件与普通队列的差别
- 堆栈、队列的简单应用

难点

- 循环队列的特点及判断溢出的条件
- 利用本章的基本知识设计有效的算法解决与线性有相关的应用问题

要求

熟练掌握以下内容：

- 线性表的基本运算
- 堆栈的特征基本运算并能设计简单算法
- 队列、循环队列的特征基本运算并能设计简单算法

了解以下内容：

- 线性表运算时间复杂性分析
- 堆栈、队列实际应用

线性表是一种最基本、最常用的数据结构。栈和队列则是线性表的特例，是加有某种限制条件的线性表。将一个线性表放在计算机中，可以采用不同的方法，其中最简单的是顺序表。顺序表是一种紧凑结构，常用一维数组来表示。本章主要介绍顺序表的运算及其应用。

2.1　数组及其顺序存储结构

数组是最常见的一种数据结构，在高级程序设计语言中，数组是应用最为广