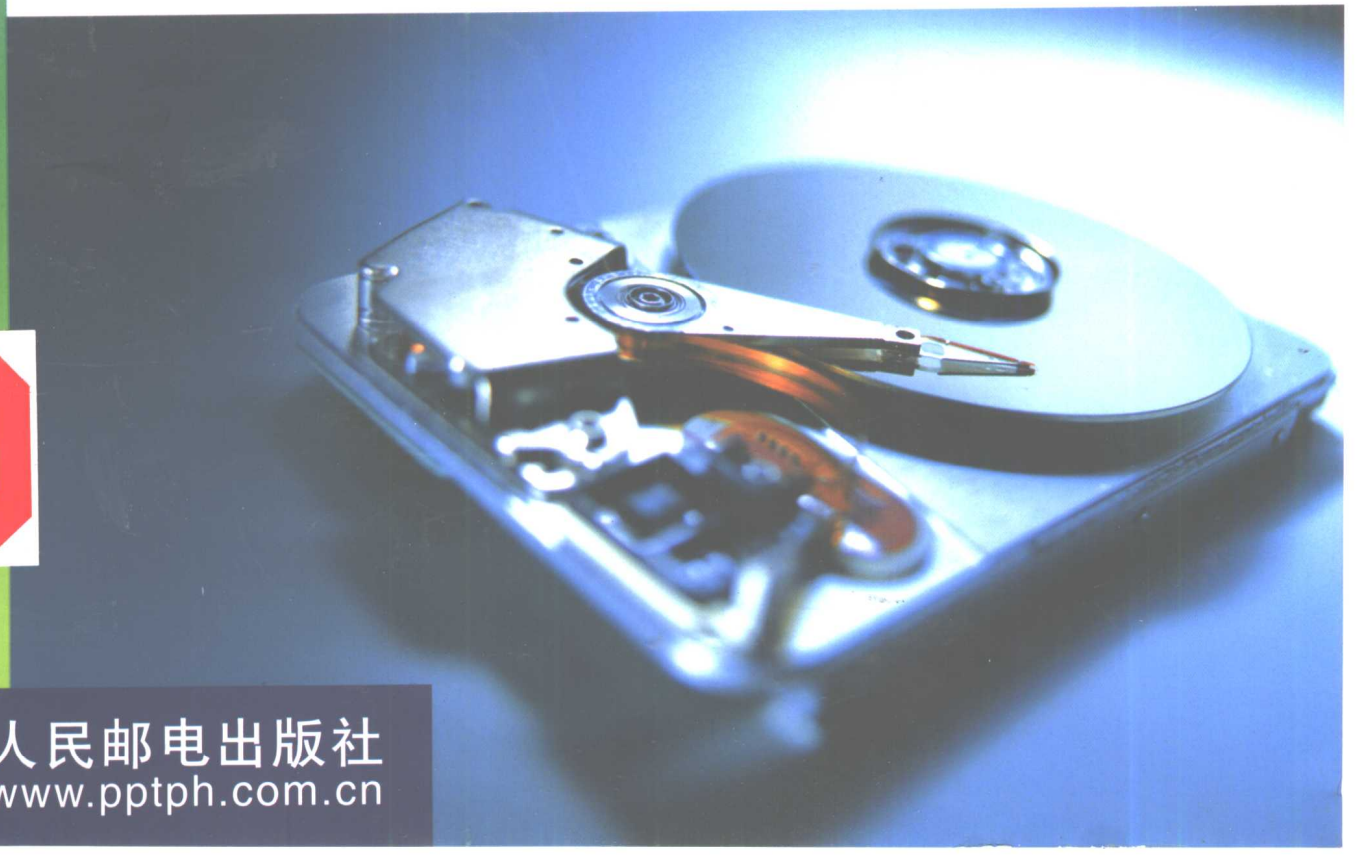
计.算.机.系.列.教.材

COMPUTER

数据结构

刘遵仁 编著



人民邮电出版社
www.pptph.com.cn

计算机系列教材

数据结构

刘遵仁　编著

人民邮电出版社

内容提要

本书共分 10 章，详细地介绍了各种数据的逻辑结构和存储结构，内容包括线性表、堆栈、队列、二叉树、图、查找和排序等。算法用 C 语言给出，简明易懂，具有较好的可读性。

本书可作为大专类和非计算机本科类数据结构课程的教材，也可供软件开发与应用人员参考。

计算机系列教材

数据结构

◆ 编　　著　刘遵仁

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号

邮编　100061　　电子函件　315@ pptph.com.cn

网址　http://www.pptph.com.cn

北京汉魂图文设计有限公司制作

北京朝阳隆昌印刷厂印刷

新华书店总店北京发行所经销

◆ 开本:787 x 1092　1/16

印张:14

字数:341 千字　　2000 年 7 月第 1 版

印数:1 - 6 000 册　　2000 年 7 月北京第 1 次印刷

ISBN 7-115-08214-6/TP·1388

定价:19.80 元

编委会名单

序　言

为了适应“逐步实现教材多样化，增加不同品种、不同档次、不同风格、不同改革实验的教材”要求，我们组织编写了这套《计算机系列教材》，以适应大、中专计算机教学的需要。

本套教材的基本任务是系统地阐述计算机的基本概念和基本操作，这些基本概念和基本操作将是未来掌握计算机知识的基础。因此，本套教材的构架并不打算定位在不断变化的、十分活跃的研究和应用领域，而是立足于对基础知识的介绍上。本套教材共包括以下10本，计划1999年出版前5本，2000年出齐。

1. 《微型计算机应用基础》
2. 《中文Windows 98实用教程》
3. 《操作系统概论》
4. 《C语言程序设计》
5. 《PASCAL语言程序设计》
6. 《数据结构》
7. 《计算机实用软件》
8. 《计算机网络基础》
9. 《微机系统原理与维修》
10. 《汇编语言程序设计》

本套教材根据计算机技术的最新发展，在取材的深度和广度方面作了精心的优化选择，从而使其在新颖性、系统性和准确性方面达到了新的高度。在写作方面，力求深入浅出、通俗易懂，并特别注重实例的选择和说明。为了加深对基本概念的掌握，各章的末尾均给出大量习题供学员课外练习。同时，每本教材的后面都附有实验题配合学员上机实习使用。

由于编写时间紧促，而且限于水平和经验的不足，书中肯定存在不少错误和遗漏，我们诚心希望使用本套教材的广大教师和同学们提出宝贵的批评建议。

教材编委会

1999年6月

前　言

随着计算机科学的迅速发展，数据结构课程越来越受到各地理工科类院校相关专业的重视，被认为是计算机领域一门十分重要的基础学科。通过对该课程的学习，能使学生学会如何分析和研究计算机加工的数据对象的特性，学会数据的组织方法，以便选择数据合适的逻辑结构和存储结构，从而能够利用计算机解决实际问题。

本书较详细地介绍了数据的逻辑结构和存储结构。从逻辑结构上看，数据有三种基本结构：线性结构、树结构和图结构；从存储结构上看，数据有两种基本结构：顺序结构和链式结构。一种逻辑结构可以采用不同的存储结构。程序设计语言与数据结构之间存在着密切的联系：程序设计语言为数据结构的描述提供了很好的手段；数据结构为程序设计语言类型系统的发展与完善奠定了基础。因此，本书着重强调了数据结构的概念及其在程序设计中的应用，以便提高学生的编程能力。

本书是为大专类学生学习数据结构课程而编写的一本教材。在编写过程中，编者参阅了国内诸如徐孝凯、严蔚敏、唐发根、李大友等专家所编写的有关教材，在此向他们表示衷心的感谢。本书内容取材适中，注重概念表述，由浅入深，前后衔接；在算法分析上，力求描述简洁。全书共分 10 章。第 1 章阐述了数据结构的一些基本概念；第 2 章到第 6 章主要讨论了线性结构，其中包括线性表、堆栈、队列、广义表以及字符串；第 7 章和第 8 章讨论了非线性结构，重点讨论了树和二叉树、图的基本概念及其应用；第 9 章和第 10 章分别介绍了几种查找和排序方法。本书讲授时数为 50～70 学时。对于大专类学生，目录中标记“ * ”号的内容可以不作为基本教学要求。

本书中的算法都是以“类 C 语言”编写的。学习本教材的学生，最好具有 C 语言的基础。

本书初稿完成后，许曰滨教授及刑玉国、沈精虎副教授对全书进行了审校，提出了许多宝贵意见，在此向他们表示衷心的感谢。

本书若有错误和不当之处，敬请专家和读者批评指正。

作　者

2000 年 5 月

目　录

第1章 绪 论

在电子计算机发展的初期阶段，人们使用计算机的主要目的是处理数据，解决人们用手工或机械计算机难于胜任的数值计算。当时所设计的操作对象都比较简单，不外乎是整型、实型和布尔型数据。以此为对象的程序设计称之为数值型程序设计，对应的软件或程序称之为数学软件。随着计算机应用领域的扩大和深入，解决非数值型问题越来越引起人们的关注。例如，银行业、电信业、工商企业等领域管理信息系统的建立，支持多媒体的资料查询、模式识别，图形化用户界面等，解决诸如此类问题使用的数学工具已经不再是分析数学及其计算方法，而是更多地用到离散数学和计算机的有关知识，所涉及的数据也越来越复杂。

非数值计算问题的数据元素之间所具有的特定联系，已不能用分析数学的方程式来简单地描述，因而产生了“数据结构”这门学科。

本章重点介绍：

- 基本术语。
- 算法。
- 时间复杂度。
- 算法的表述。

1.1 基本术语

数据（Data）：描述客观事物的数字、字符以及一切能够输入到计算机中，并且能够被计算机程序处理的符号的集合，即计算机能够加工处理的对象或信息。数据的含义十分广泛，在不同的场合下具有不同的含义。例如一本书、一篇文章、一张图表、一盘录音或录像带等是数据，一个单词、一个算术表达式、一个数值、一个字符等也都是数据。

数据元素（Data Element）：数据的基本单位，即数据这个集合中的一个个的客体。在程序中数据元素通常是作为一个整体来被处理的。例如对于一个数据库文件来说，每一条记录就是它的数据元素；对于一个字符串来说，每个字符就是它的数据元素；对于一个数组来说，每一个单元就是它的数据元素。

一个数据元素可以由若干个数据项（数据项是数据的最小单位）所组成。例如对于学生简历来说，每一个记录表示一个学生的信息，它是由若干个数据项所构成的，如表1-1所示。

由表中可知，一个数据元素由 6 个数据项所组成，而每个数据项是不可再分解的最小的数据，它分别描述了客体（学生）的某一方面的特征。

表 1-1 学生简历

学号	姓名	性别	爱好	身高	备注
20000301	王 刚	男	足球	1.73	
20000302	张 芳	女	音乐	1.65	
20000303	周小丹	男	小提琴	1.78	
20000304	南晓飞	男	舞蹈	1.80	三好学生
…	…	…	…	…	…

数据对象（Data Object）：具有相同特性的数据元素的集合，是数据这个集合的一个子集。例如，自然数的数据对象是集合｛1，2，3，…｝，而由 26 个英文字母组成的数据对象则是集合｛A，B，C，D，…，Z｝。

数据处理（Data Processing）：是指对数据进行查找、插入、删除、排序、计算、输入、输出等的操作过程，处理的目的是获得有用的信息。

数据结构（Data Structure）：简单而言是指数据元素之间的联系。因为数据是客观世界事物及其活动的描述，而任何事物及其活动都不是孤立存在的，彼此之间必然存在着某种联系。由于这种联系是内在的，或根据人们的需要所定义的，被看作是“逻辑”上的联系，因此，又把数据结构称作数据的逻辑结构或逻辑关系。数据结构在计算机存储器上的存储表示称之为数据的物理结构或存储结构。由于存储表示的方法有顺序、链式两种，所以，一种数据结构可用一种或两种物理结构实现。

为了确切地描述数据结构，通常采用二元组表示：

$$DB=(D,\ R)$$

DB 是一种数据结构，它由数据元素的有限集合 D 和 D 上二元关系的有限集合 R 所组成。其中

$$D=\{d_i \mid 1\leqslant i\leqslant n,\ n\geqslant 0\ \}$$
$$R=\{r_j \mid 1\leqslant j\leqslant m,\ m\geqslant 1\}$$

d_i 表示第 i 个数据元素，n 为 DB 中数据元素的个数，特别地，若 n=0，则 D 是一个空集，故 DB 也就无结构而言，或者说它具有任何结构；r_j 表示第 j 个二元关系（简称关系），m 为 D 上关系的个数。

在本书所讨论的数据结构中，只讨论 m=1 的情况，即 R=｛r｝。

D 上的一个关系 r 是有序偶的集合。对于 r 中的任何一个有序偶对 $<x,\ y>$（$x,\ y\in D$），把 x 叫做有序偶对的第一个元素，把 y 叫做有序偶对的第二个元素。有时，把有序偶对的第一个元素称为第二个元素的直接前驱，简称前驱；称第二个元素为第一个元素的直接后继，简称后继。

由于同一种逻辑结构可以映射成不同的存储结构，如顺序存储结构或非顺序存储结构（或称为链式存储结构），因此，数据的存储结构一定要正确地反映出数据元素之间的逻辑关系。

为了具体说明这一点，下面再举一个例子。

例如，一个具有 20 个记录的反映学生家庭地址情况的数据文件，其记录在文件中的先后顺序是按学生学号从小到大排列的，如表 1-2 所示。

表 1-2 学生家庭地址表

学号	姓名	城市	街道	门牌号
19980801	邱卫国	北京	王府井大街	20
19980802	张　强	上海	淮海中路	31
19980803	南　健	青岛	香港中路	8
…	…	…	…	…
19980820	卞　军	广州	中山路	19

该数据文件中记录之间的逻辑关系是一个线性关系（因此也称该数据文件为一个线性表）。对应于这个逻辑结构，在计算机内可以有两种物理结构，即顺序存储结构和链式存储结构。前者用一块地址连续的存储空间依次存放该文件的 20 个记录，记录物理上的先后关系映射了记录逻辑上的先后关系。也就是说，逻辑上相邻的两个数据元素，其存储位置也相邻。后者则用 20 个称为链表结点的存储空间分别存放这 20 个记录，每个结点物理地址上可以不连续，但是通过链指针来映射记录之间的逻辑关系。见图 1-1 所示。

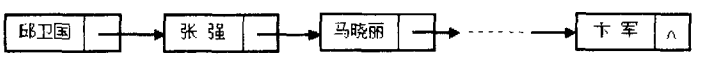


图 1-1　线性链表示意图

由此不难想到，在实现某种操作之前，首先需要选择合适的存储结构，而同一种操作在不同的存储结构中实现的方法不同，有的则完全依赖于所选择的存储结构。可见，数据的逻辑结构与存储结构是紧密相联的两个方面。

综上所述，数据结构所要研究的主要内容，可以简单地归纳为以下 3 个方面：

（1）研究数据元素之间固有的客观联系（逻辑结构）。

（2）研究数据在计算机内部的存储方法（存储结构）。

（3）研究如何在数据的各种结构（逻辑和物理）上实施有效的操作（算法）。

因此，应该说数据结构是一门抽象地研究数据之间结构关系的学科。

1.2 算法的概念

数据结构与算法之间存在着密切的联系。算法的结构设计和选择，在很大程度上依赖于作为其基础的数据结构，即数据结构为算法提供了工具，而算法则是应用这些工具来具体解决问题的方案。凡是从事过程序设计的人都会有这样一个体会：程序设计过程中有相当多的时间花费在构思算法上，一旦有了合适的算法，用具体的程序设计语言来实现（编写程序）并不是一件很困难的事情。有个著名的观点是：算法＋数据结构＝程序。从这个角度上说，要设计出一个好的程序，在很大程度上取决于设计出一个好的算法。

算法（Algorithm）是对特定问题求解步骤的一种描述，是用来解决某个问题的一些指令的集合。它是指令的有限序列，其中每一条指令表示一个或多个操作。由此可以说，程序就是用计算机语言表述的算法，流程图就是图形化的算法，甚至一个公式也可以叫算法。

在计算机领域，一个算法实质上是根据所处理问题的需要，在数据的逻辑结构和物理结构的基础上施加的一种操作。由于数据的逻辑结构与物理结构不是唯一的，在很大程度上可以由设计人员进行选择和设计，因而处理同一个问题的算法也不一定是唯一的；即使具有相同的逻辑结构与物理结构，但如果设计思路和技巧不同，设计出来的算法也不会相同。显然，根据数据处理的需要，为所处理的数据选择合适的逻辑结构与物理结构，进而设计出比较满意的算法，是学习数据结构这门课程的主要目的。

作为一个完整的算法，应该满足下面 5 个标准，通常称之为算法的基本性质：

- 确定性　算法中的每一条指令必须有确定的含义，不应该产生二义性，即不允许出现不同的人对于每个指令有不同理解的情况。并且，在任何条件下，算法只有唯一的一条执行路径，即对于相同的输入只能得出相同的输出。
- 有穷性　一个算法必须在执行有限的步骤之后结束，并且每一步骤也都必须在有限的时间之内完成。
- 可行性　算法中描述的每一个操作，都可以通过已经实现了的基本运算，执行有限次后来完成。
- 输入　一个算法有零个或若干个输入，这些输入都来自于某一特定的对象的集合。
- 输出　一个算法有一个或多个输出。

算法还有一个重要的方面，就是构成这个算法所依据的方法（公式、方案、准则）。有许多问题，只要对数据对象进行细致的分析，就能确定处理方法，有的问题则不然。不过，作为寻找设计思路的基本思想方法，对任何算法设计都是有用的。这些方法通常有枚举法、归纳法、递归法等等。

算法独立于具体的计算机与具体的程序设计语言。在设计一个算法时，应选择一种合适的方式来表达算法思想，或者说，有了解决问题的算法思想，应选择一种合适的语

言来描述算法的各个步骤。在计算机发展的初期，人们往往用自然语言来表达自己的算法思想。下面看一个简单例子。

例如，判断正整数 M 和 N 哪个大。

若采用自然语言描述解决该问题的各个步骤，可表达为：

① M 减 N，将差值赋给临时变量 R。

② 判断 R 是否为零。

（a） 若 R 等于零，则 M 和 N 一样大。

（b） 若 R 大于零，则 M 大于 N。

（c） 若 R 小于零，则 M 小于 N。

类似的简单问题用自然语言表达还是可以的，但很快就会发现，采用自然语言描述算法很不方便，也不直观，更谈不上有良好的可读性，稍微复杂一些的算法就难以表达，甚至无法表达。另外，由于自然语言本身的一些限制，用它描述的算法可能会出现二义性。采用流程图的形式来描述算法（见图 1-2），比采用自然语言表达直观了一些，但依然没有解决复杂算法的表达问题，而且移植性也不好。

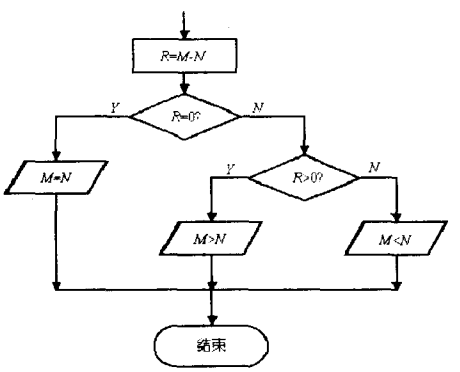


图 1-2 判断正整数 M 和 N 哪个大的流程图

如果算法直接采用某种具体的程序设计语言来描述，则将受到具体语言语法细节的限制（如繁琐的变量说明、语句的书写规则等等）。本书在进行算法描述时，采用“类 C 语言”作为工具。所谓类 C 语言就是在 ANSI C 语言的基础上所做的取舍，它忽略了 ANSI C 语言中语法规则的一些细节，描述出的算法清晰、直观，便于阅读和分析。这样书写的算法，虽然不能直接在计算机上执行，但经过稍加修改和补充，就很容易变成计算机所能执行的程序了。

对于上面的问题，用类 C 语言描述的方法为：

```
void comp (int M, int N)
{
    R = M—N;
    if (R == 0)
        printf ("M = N");
```

```
    else if (R > 0)
        printf ("M > N");
    else
        printf ("M < N");
}
```

1.3 算法描述

本书采用的“类C语言”是ANSI C语言的一部分。以下作简要说明。

（1）预定义常量

```
#define     TRUE     1
#define     FALSE    0
```

（2）数据元素类型

约定为elemtype，同学们在使用该数据类型时自行定义。

（3）基本操作的算法用以下形式的函数描述：

```
函数类型  函数名（函数参数表）{
/* 算法简单说明 */
语句体;
}
```

（4）赋值语句

```
变量名 = 表达式;
变量名[ ] = 表达式;
```

（5）条件语句

```
条件语句1    if（表达式） 语句体;
条件语句2    if（表达式） 语句体;
             else 语句体;
```

（6）循环语句

```
for（赋初值表达式序列; 条件; 修改表达式序列） 语句体;
while（条件） 语句体;
do {
    语句体;
} while（条件）;
```

（7）结束语句

```
函数结束语句     return 表达式;
异常结束语句     exit（异常代码）;
```

（8）输入和输出语句

输入语句　　scanf（[格式串]，变量1，变量2，…，变量n）;

输出语句　　printf（[格式串]，表达式1，表达式2，…，表达式n）;

（9）逻辑运算

逻辑“与”　　对于$A\&\&B$，当A的值为0时，不再对B求值。

逻辑“或”　　对于$A||B$，当A的值为非0时，不再对B求值。

（10）符号和函数

符号$\lfloor x \rfloor$表示不大于x的最大整数，如$\lfloor 2.7 \rfloor = 2$;

符号$\lceil x \rceil$表示不小于x的最小整数，如$\lceil 3.1 \rceil = 4$;

$m \% n$ 表示m对n取模。

（11）注释

单行或多行注释　　/* 说明文字 */

单行注释　　// 说明文字

1.4 算法分析

对同样一个问题，不同的人能够写出不同的算法。对算法进行分析，可以从解决同一个问题的不同算法中，选择出较为合适的一个，也能知道如何对现有算法进行改进，从而有可能设计出更好的算法来。

算法分析的目的在于改进算法的设计。

对一个算法进行评价时，首先要看算法是否正确，这是前提条件。所谓算法的正确性，是指当输入一组合法的数据时，算法能够在有限的时间内，得出正确的结果，而对于不合理的数据输入，也能够给出相应的错误提示信息。通常要验证一个算法是否正确，可以通过输入不同的数据来进行测试，特别是可输入一些极端情况下的数据来进行验证。然而，要从理论上证明一个算法是否正确却不是一件容易的事情。

对算法进行分析时，除了要考虑算法的正确性外，还需从以下三个方面来考察：

（1）根据该算法编写出的程序，在计算机中运行时间的度量，即所谓的时间复杂度。它是一个算法运行时间的相对度量。

（2）根据该算法编写出的程序，占用计算机存储空间的度量，即所谓的空间复杂度。

（3）其他方面。诸如算法的可读性、可移植性、简单性以及容易测试等等。

从理论上讲，一个好的算法既不占用很多的存储空间，运行时间又短，并且在其他方面性能也好。然而，实际上时间与空间的占用往往是一对矛盾，因此，十全十美的算法实际上是极少的，甚至是不存在的。有时候，一个形式上看起来很简单的算法，其对