高等学校计算机基础课程系列教材

艾德才　刘　山　主　编
蒋浩江　刘　山　冯兴杰　编　著

数据结构

中国水利水电出版社
www.waterpub.com.cn

高等学校计算机基础课程系列教材

数据结构

艾德才　刘山　主编

蒋浩江　刘山　冯兴杰　编著

中国水利水电出版社

内 容 提 要

本书共 11 章。内容包括：绪论；线性表、栈和队列；串；递归；数组；树型结构；图；存储管理；查找；排序及文件，概括了数据结构领域内的基本知识和内容。书中在对算法进行描述时采用的是 C 语言，因为 C 语言提供了较强的结构描述功能，允许用基本类型及用户自定义类型来定义复杂的数据结构。本书通俗易懂、实例丰富、实用性强。每一章之后配有习题，供自学自测用。

本书可作为高等学校计算机专业及相关非计算机专业教学用书，也可作为培训教材。

图书在版编目（CIP）数据

数据结构 / 艾德才，刘山主编. —北京：中国水利水电出版社，2002
高等学校计算机基础课程系列教材
ISBN 7-5084-1133-1

Ⅰ.数… Ⅱ.①艾…②刘… Ⅲ. 数据结构—高等学校—教材 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字（2002）第 044498 号

书　　名	**数据结构**
作　　者	蒋浩江　刘山　冯兴杰　编著
出版、发行	中国水利水电出版社（北京市三里河路 6 号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@public3.bta.net.cn（万水） sale@waterpub.com.cn 电话：（010）68359286（万水）　63202266（总机）　68331835（发行部）
经　　售	全国各地新华书店
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	787×1092 毫米　16 开本　14.75 印张　327 千字
版　　次	2002 年 7 月第一版　2002 年 7 月北京第一次印刷
印　　数	0001—5000 册
定　　价	20.00 元

凡购买我社图书，如有缺页、倒页、脱页的，本社发行部负责调换

前　言

为尽快实现教育部提出的21世纪计算机教育要上一个新台阶的宏伟目标，落实新世纪需要培养出一大批具有现代科学知识能独立解决实际问题，勇于创新的高科技人才的宏图，编写了本书，书中内容丰富、系统、完整，是多年教学经验和心得体会的结晶。

本书共有11章。内容包括：绪论；线性表、栈和队列；串；递归；数组；树型结构；图；存储管理；查找；排序及文件。本书概括了当前数据结构领域里的基本知识和内容，是当今计算机专业本科生必须具备的计算机知识。

数据结构是计算机专业学生必学的一门专业技术基础课，通过本书的学习，学会分析研究计算机加工处理的数据对象的特征，初步掌握各种算法在时间和空间的分析技巧。从某种意义上来说，学习数据结构的过程也是对复杂程序设计进行训练的过程。

本书在对算法进行描述时采用的是C语言，因为C语言是一种高效、灵活、实用、精练的高级程序设计语言，并且提供了较强的结构描述功能，允许用基本类型及用户自定义类型来定义复杂的数据结构，为ADT数学模型在计算机上的实现提供了极大的方便，ADT操作可以用C语言提供的基本数据类型、用户自定义数据类型来实现。所以读者在阅读本书时应该已经熟知C语言。

数据结构是一门实验环节很强的计算机技术基础课程，上机实习是不可缺少的一个重要学习过程，书中的举例仅需稍加修改即可在计算机上使用，实用性较强。而且书中每一章都有习题供复习用。

本书第1章至第3章由蒋浩江编写，第4章、第5章由艾德才编写，第6章至第8章由刘山编写，第9章至第11章由冯兴杰编写，全书由艾德才、刘山教授任主编。

出版本教材，是在计算机教育改革上进行的一次尝试，虽力图做好，但由于作者水平有限，难免有不足之处，殷切希望能得到广大同仁和读者的批评指正，以便使本教材的质量得到进一步提高。

编　者

2002年3月于天津

目　　录

第1章 绪 论

1.1 数据结构发展概况

数据结构（Data Structure）是一门研究非数值计算的程序设计问题中，计算机的操作对象以及它们之间关系和操作等的学科。

回顾一下数据结构课程的发展与形成过程，对于理解数据结构的内容和重要性是很有意义的。

数据结构作为一门独立的课程在国外是从1968年才开始设立的。在这之前，它的某些内容曾在其它课程如表处理语言中有所阐述。1968年在美国一些大学计算机系的教学计划中，虽然把数据结构作为一门课程，但对课程的范围没有作明确的规定。当时，数据结构几乎和图论，特别是与表和树的理论为同义语。随后数据结构这个概念被扩充到包括网络、集合代数论、格、关系等方面。由于数据必须在计算机中进行处理，因此不仅考虑数据本身的数学性质，而且还必须考虑数据的存储结构，又随着数据库系统的不断发展，在数据结构课程中又增加了文件管理（特别是大型文件组织）的内容。

1968年美国唐·欧·克努特教授开创了数据结构的最初体系。他所著的《计算机程序设计技巧》第一卷《基本算法》，是第一本较系统地阐述数据的逻辑结构和存储结构及其操作的著作，从20世纪60年代末到70年代初，出现了大型程序，软件也相对独立，结构程序设计成为程序设计方法学的主要内容，数据结构的地位显得更为重要，人们认为程序设计的实质是对确定的问题选一个好的结构，加上一种好的算法。目前在我国，数据结构早已成为计算机专业的必修课之一。

数据结构在计算机科学中是一门综合性的专业基础课。数据结构的研究不仅涉及到计算机硬件的研究范围，而且和计算机软件的研究有着密切的关系。无论是编译程序还是操作系统，都涉及到数据元素在存储器中的分配问题。在研究信息检索时也必须考虑如何组织数据，以便查找和存储数据元素更为方便，可以认为数据结构是介于数学、计算机硬件和计算机软件三者之间的一门核心课程，如图1-1所示。在计算机科学中，数据结构不仅是一般程序设计的基础，而且是设计和实现编译程序、操作系统、数据库系统及其它系统程序和大型应用程序的重要基础。

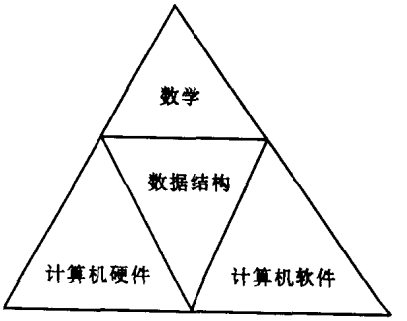

图1-1 数据结构所处的地位

目前数据结构的发展仍在继续，一方面，在面向各专门领域中特殊问题的数据结构得到研究和发展，如多维图形数据结构等；另一方面，从抽象数据类型的观点来讨论数据结构已经成为一种趋势，越来越被人们所重视。

1.2 数据结构的基本概念和数据类型

1.2.1 基本概念

1. 数据（Data）

在计算机中，数据这个术语含义非常广泛，可以认为它是描述客观事物的数字、字符以及所有能输入到计算机中并能为计算机所接受的符号集。

2. 数据元素（Data Element）

数据的基本单位，是计算机进行输入输出操作的最小单位。在大多数情况下，一个数据元素由若干个数据项组成。数据项是数据不可分割的最小单位。

3. 数据结构（Data Structure）

是指计算机处理的数据元素的组织形式和相互间的关系。定义数据结构是一个二元组，即

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

其中，D 为数据结构有限集；S 为 D 上关系的有限集。

从数据元素之间的不同特性划分，数据结构有 4 种基本类型：集合、线性结构、树型结构、图形结构。

（1）集合。结构中的数据元素除了同属于一个集合关系外，别无其它关系。

（2）线性结构。数据元素之间存在一个对一个的关系。

（3）树型结构。数据元素之间存在一个对多个的关系。

（4）图形结构。数据元素之间存在多个对多个的关系。

图 1-2 中示出了 4 种基本结构关系图。

从不同的抽象层次划分，数据结构可以分为逻辑数据结构和物理数据结构（或称存储结构）。存储结构主要有两种基本类型：顺序存储结构和链式存储结构。

（1）逻辑数据结构。是从操作对象抽象出来的数学模型，结构定义中的“关系”描述是数

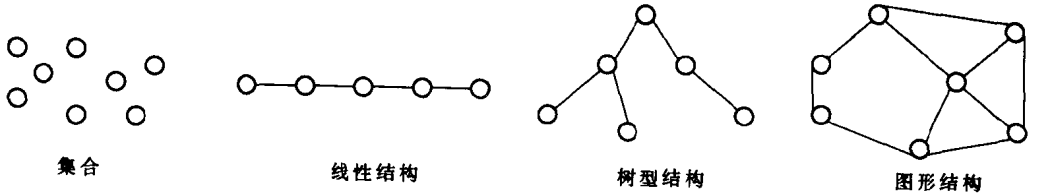

图 1-2 4 种基本结构关系图

据元素之间的逻辑关系。

(2) 物理数据结构。数据在计算机中的表示，又称存储结构。它包括数据元素的表示和关系的表示。在计算机中表示信息的最小单位是二进制数的一位，叫做位。在计算机中，可以用若干个位组合起来形成一个位串，用以表示一个数据元素，通常称这个位串为元素或结点。当数据元素由若干个数据项组成时，位串中对应于各个数据项的子位串称为数据域。

①顺序存储结构：借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系。所占用的地址空间是连续的地址空间。

②链式存储结构：借助指示元素存储地址的指针表示数据元素之间的逻辑关系。所占用的地址空间可以是连续的地址空间，也可以是不连续的地址空间。

如图 1-3 所示，对于一组数据如 {1，3，5，7}，我们可以采用顺序存储结构或链式存储结构分配内存单元进行存储。

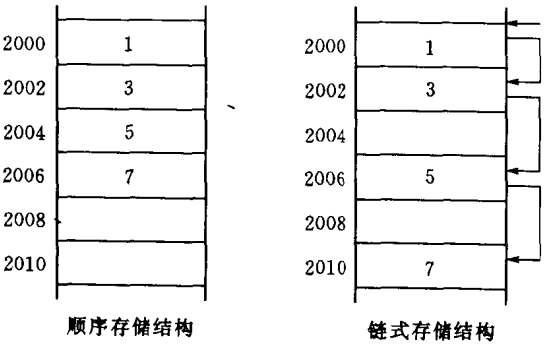

图 1-3 数据在存储器中的存储器方式

一个算法的设计取决于选定的数据逻辑结构，而算法的实现依赖于采用的存储结构。

4. 操作

是指数据结构上需要或能够进行的处理。对一个具体问题来说，存储结构不同，则操作的具体实现算法就不同；相同的存储下也可以有多种不同的操作实现算法。

1.2.2 数据类型和抽象数据类型

1. 数据类型

数据类型这个概念最早出现在高级程序设计语言的实现中。

最初，计算机硬件系统并不能直接处理如整数、浮点数、双精度数等，需要按照一定的规则转换成二进制码来表示和处理。从计算机硬件系统的角度来看，计算机能处理的数据类型只包括位、字节和字。

高级程序设计语言的设计者们引入的整数、浮点数、双精度数等，对计算机硬件系统来说就是一种新的数据结构。对这种数据结构的操作，需要通过编译器转化成机器语言的位、字节和字这样的计算机硬件能够接受的数据类型的操作来实现。从高级程序设计语言用户的角度来

看，数据结构实现了信息的隐蔽，即将一切用户不必要了解的细节都封装在数据结构之中，从而实现了数据的抽象。例如硬件系统直接支持的机器语言程序需要考虑整数的具体机器表示和整数运算的机器实现算法，而高级程序设计语言用户在使用整数时，就不需要了解整数在计算机内部是如何表示的，以及各种运算是如何实现的。如对两个整数求和这样的问题，高级程序设计语言的用户只需要了解其数学上求和的抽象操作含义，这样在硬件系统机器语言的数据类型支持下，高级程序设计语言实现了整数这样的数据结构。对高级程序设计语言来说，整数就是它的固有数据类型，高级程序设计语言用户可以直接使用。在用高级程序设计语言编写的程序中，每个变量、常量和表达式都有一个它所属的确定的数据类型，并且规定了在程序执行期间变量、常量和表达式所有可能的取值范围，以及允许进行的操作。

因此我们可以给数值类型作如下定义：

数据类型是一个数据的集合和定义在这个数据集合上的一组操作的总称。或者说，数据类型是某种程序设计语言中已实现的数据结构。

2. 抽象数据类型

抽象数据类型（简写为 ADT）是用户在数据类型基础上新定义的数据类型。抽象数据类型定义包括数据组成和对数据的处理操作。抽象数据类型是数据和数据使用者的一个接口。

程序设计方法学研究指出，一个软件系统的框架应建立在抽象数据类型之上。从抽象数据类型的角度入手设计软件系统可大大提高软件构件的复用率。

抽象数据类型的定义主要包括：数据对象的定义、数据关系的定义和基本操作定义 3 个部分。其书写格式为：

抽象数据类型：　抽象数据类型名
数据对象：　数据对象的定义
数据关系：　数据逻辑关系的定义
基本操作：　基本操作的定义

其中，数据对象定义应该是在已有数据类型或已定义数据对象的基础上对新的数据对象作出的定义，基本操作定义主要包括操作名、参数表、初始条件和操作结果四部分内容的定义和描述。其书写格式为：

操作名（参数表）
操作结果：操作结果描述

1.3 算法描述和分析

1.3.1 算法描述

所谓算法是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每一条指令表示

一个或多个操作。

算法具有以下5个重要特性。

1. 有穷性

一个算法必须总是对任何合法的输入值在执行有穷步之后结束，且每一步都可在有穷时间内完成。

2. 确定性

算法中每一条指令必须有确切的含义，读者理解时不会产生二性。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。

3. 可行性

一个算法是能行的，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的。

4. 输入

一个算法有零个或多个的输入，这些输入取自于某个特定的对象的集合。

5. 输出

一个算法有一个或多个的输出。这些输出是同输入有着某些特定关系的量。

算法设计过程中要求一个“好”的算法应考虑达到以下目标。

1. 正确性

算法应当满足具体问题的要求。通常一个大型问题的需求，要以特定的规格说明方式给出，而一个实习问题或练习题，往往就不那么严格，目前多数是用自然语言描述需求，它至少应当包括对于输入、输出和加工处理等的明确的无歧义性的描述。设计或选择的算法应当能正确地反映这种需求，否则，算法的正确与否的衡量准则就不存在了。

2. 可读性

可读性有助于人们对算法的理解，有助于算法中隐藏错误的排除，也有助于算法的交流和移植。

3. 健壮性

当输入数据非法时，算法应该能作出适当的处理，而不应该产生不可预测的结果。

4. 高效率

算法的效率指算法的执行时间。对于同一个问题如果有多个算法可供选择，应该尽可能选择执行时间较短的算法。算法的效率在数据结构中称作算法的时间复杂度。

5. 低存储量

算法的存储量要求指算法执行过程中所需要的最大存储空间。对于同一个问题如果有多个算法可供选择，应该尽可能选择存储量需求低的算法。算法的存储量需求在数据结构中称作算法的空间复杂度。通常算法的高效率和低存储量的需求是互相矛盾的。

算法需要用一种语言来描述，同时算法可以有各种描述方法以满足不同的需求，例如一个需要在计算机上运行的程序（程序也是算法），必须是严格按照语法规定用机器语言或汇编语

言或高级语言程序编写的，同时为了人类阅读和交流算法的方便，还可以用框图等其它形式来进行描述。

本书采用了C语言对算法进行描述，因为C语言作为一种高效、灵活、实用、精练的高级程序设计语言而被广泛采用，C语言提供了较强的结构描述功能，允许用基本类型及用户自定义类型定义复杂的数据结构，为ADT数学模型在计算机上实现提供了方便，ADT操作可以用C语言提供的基本数据类型、用户自定义数据类型来实现。读者在阅读本书的时候应该已经熟练掌握C语言。

1.3.2 算法分析

我们已经讲过一个好的算法应该达到如上所述的5个基本目标，其中高效率和低存储量又是其中重要的衡量标准。也就是所谓合理的时间复杂度和空间复杂度。

1. 时间复杂度

一个用高级程序语言编写的程序在计算机上运行时所消耗的时间取决于下列因素：

(1) 依据的算法是采用何种策略。

(2) 问题的规模，例如是求100以内还是10 000以内的素数。

(3) 书写程序的语言，对于同一算法，实现语言的级别越高，执行效率就越低。

(4) 编译程序所产生的机器代码的质量。

(5) 机器执行指令的速度。

显然同一个算法用不同的语言实现，或者用不同的编译程序进行编译，或者在不同的计算机上运行时，其效率是不相同的。这表明使用绝对的时间单位衡量算法的效率是不合适的。我们应该不考虑这些与计算机硬件、软件有关的因素，可以认为一个特定算法的运行工作量的大小，只依赖于问题的规模（通常用整数量n表示），或者说它是问题规模的函数。

一个算法是由控制结构（顺序、分支和循环3种）和原操作（指固有数据类型的操作）所构成的，则算法时间取决于两者的综合效果。为了便于比较同一问题的不同算法，通常的做法是，从算法中选取一种对于所研究的问题（或算法类型）来说是基本操作的原操作，以该基本操作重复执行的次数作为算法的时间量度。

例如，在如下所示的两个$N \times N$矩阵相乘的算法中，乘法运算是矩阵相乘问题的基本操作。整个算法的执行时间与该基本操作(乘法)重复执行的次数n^3成正比，可以记作$T(n)=O(n^3)$。

```
for (i=1; i<=n; ++i)
    for (j=1; j<=n; ++j)
        {
          c [i] [j] =0;
          for (k=1; k<=n; ++k)
              c [i] [j] +=a [i] [k]* b [k] [j];
        }
```

一般情况下，算法中基本操作重复执行的次数是问题规模 n 的某个函数 f（n），算法的时间量度记作：

$$T(n) = O(f(n))$$

它表示随问题规模 n 的增大，算法执行时间的增长率和 f（n）的增长率相同，称做算法的时间复杂度。

显然，被称做问题的基本操作的原操作应是其重复执行次数和算法的执行时间成正比的原操作，多数情况下它是最深层循环内的语句中的原操作，它的执行次数和包含它的语句的频度相同。语句频度指的是该语句重复执行的次数。例如，在下列 3 个程序段中：

```
(1) {++x; s = 0;}
(2) for (i=1; i<=n; ++i) {++x; s+=x; }
(3) for (j=1; j<=n; ++j)
        for (k = 1; k<=n; ++k) {++x; s+=x; }
```

含基本操作“x 增 1”的语句频度分别为 1、n、n^2，则这 3 个程序段的时间复杂度分别为 O（1）、O（n）和 O（n^2），分别称为常量阶、线性阶和平方阶。算法还可能呈现的时间复杂度有：对数阶 O（logn），指数阶 O（2^n）等。我们应该尽可能选用多项式阶 O（n^k）的算法，而不希望用指数阶的算法。

一般情况下，对一个问题（或一类算法）只需选择一种基本操作来讨论算法的时间复杂度，有时也需要同时考虑几种基本操作，甚至可以对不同的操作赋以不同的权值，以反映执行不同操作所需要的相对时间，这种做法便于综合比较解决同一问题的两种完全不同的算法。

由于算法的时间复杂度考虑的只是对于问题规模 n 的增长率，则在难以精确计算的基本操作执行次数（或语句频度）的情况下，只需求出它关于 n 的增长率或阶即可。例如，在下列程序段中：

```
for (i=2; i<=n; ++i)
    for (j=2; j<=i - 1; ++j)
        {++x; a [i, j] =x;}
```

语句 ++x 的执行次数关于 n 增长率为 n^2，它是语句频度的表达式（n − 1）（n − 2）/2 中增长最快的项。

有的情况下，算法中基本操作重复执行的次数还随问题的输入数据集不同而不同。例如，在下列起泡排序的算法中：

```
/*自小到大排序*/
main ( )
{
 int a [11];
 int i, j, t;
 for ( i=1; i<11; i++)
          scanf ( "%d", &a [i]);
```

```
for (j=1; j<=9; j++)
    for (i=1; i<=10-j; i++)
        if (a[i] >a[i+1])
          {
           t=a[i];
           a[i] =a[i+1];
           a[i+1] =t;
          }
for (i=1; i<11; i++)
    printf ("%d", a[i]);
}
```

“交换序列中相邻两个整数”为基本操作。当 a 中初始序列为自小到大有序，基本操作的执行次数为 0；当初始序列自大到小有序时，基本操作的执行次数为 n（n－1）/2。对这类算法的分析，一种解决的方法是计算它的平均值，即考虑它对所有可能的输入数据集的期望值，此时相应的时间复杂度为算法的平均时间复杂度。例如，假设 a 中初始输入数据可能出现 n! 种的排列情况的概率相等，则起泡排序的平均时间复杂度为 T_{arg}（n）=O（n^2），然而，在很多情况下，各种输入数据集出现的概率难以确定，算法的平均时间复杂度也就难以确定。因此，另一种更可行也是更常用的办法是，讨论算法在最坏情况下的时间复杂度，即分析最坏情况以估算算法执行时间的一个上界。例如上述起泡排序的最坏情况为 a 中初始序列为自大到小有序，则起泡排序算法在最坏情况下的时间复杂度为 T（n）=O（n^2）。

2. 空间复杂度

类似于算法的时间复杂度，本书中以空间复杂度作为算法所需存储空间的量度，记作：

$$S(n) = O(f(n))$$

其中，n 为问题的规模或大小。

一个上机执行的程序除了需要存储空间来存储本身所用的指令、常数、变量和输入数据外，也需要一些对数据进行操作的工作单元和存储一些为实现计算所需信息的辅助空间。若输入数据所占空间只取决于问题本身，和算法无关，则只需要分析除输入和程序之外的额外空间，否则应同时考虑输入本身所需空间（和输入数据的表示形式有关），若额外空间相对于输入数据量来说是常数，则称此算法为原地工作。

习 题 一

1－1 什么是数据、数据元素、数据结构？

1－2 什么是逻辑数据结构，什么是存储结构？

1－3 什么是算法，算法优劣的评价标准是什么？

1－4 设 n 为正整数，求出下列算法基本操作的执行时间 T（n）

(1) i=1; k=0;