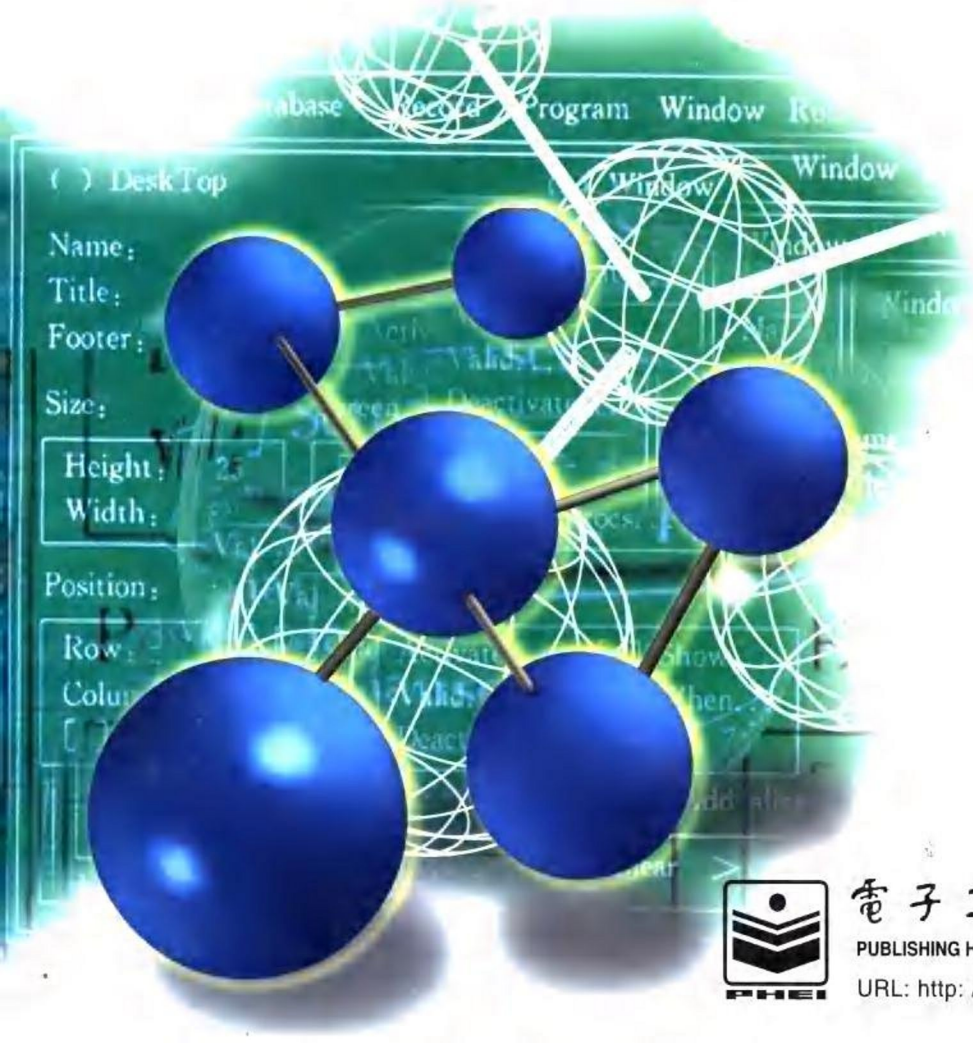
高等专科学校教材

中国计算机学会大专教育学会推荐出版

数据结构

谈春媛　江红　编



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: http: // www. phei. co. cn

高等专科学校教材

数据结构

谈春媛　江　红　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书是中国计算机学会大专教育委员会、大专计算机教材编审委员会组织编写、审定、推荐出版。是为大专院校计算机及电子类专业编写的《数据结构》课程教材。全书共分八章，分别为绪论、线性表、串、数组、树、图、查找和排序。全书用 C 语言作为算法描述语言，详细介绍了各种数据结构的特性，存储表示和有关运算的算法。

本书概念清楚，内容丰富，为便于巩固教学各章后都附有习题，是一本适合用于教学的教材，也可以作为从事计算机工程与应用人员的参考书。

丛 书 名：高等专科学校教材
书　　名：数据结构
编 著 者：谈养媛　江　红
责任编辑：赵家鹏
特约编辑：程　会
排版制作：电子工业出版社计算机排版室
印 刷 者：北京大中印刷厂
出版发行：电子工业出版社　　URL：http://www.phei.com.cn
北京市海淀区万寿路 173 信箱　邮编 100036　发行部电话：68279077
经　　销：各地新华书店
开　　本：787 × 1092 1/16　印张：9.75　字数：243.2 千字
版　　次：1997 年 8 月第 1 版　1999 年 7 月第 4 次印刷
书　　号：ISBN 7-5053-3862-5/G·302
定　　价：12.00 元

出 版 说 明

根据国务院关于高等学校教材工作的有关规定，在电子工业部教材办的组织与指导下，按照教材建设适应“三个面向”的需要和贯彻国家教委关于“以全面提高教材质量水平为中心、保证重点教材，保持教材相对稳定，适当扩大教材品种，逐步完善教材配套”的精神，大专计算机专业教材编审委员会与中国计算机学会教育专业委员会大专教育学会密切合作，于1986～1995年先后完成了两轮大专计算机专业教材的编审与出版工作，共出版教材48种，从而较好地解决了全国高等学校大专层次计算机专业教材需求问题。

为及时使教材内容更适应计算机科学与技术飞速发展的需要以及在管理上适应国家实施“双休日”后的教学安排；在速度上适应市场经济发展形势的需要，在电子工业部教材办的指导下，大专计算机专业教材编委会、中国计算机学会大专教育学会与电子工业出版社密切合作，从1994年7月起经过两年的努力制定了1996～2000年大专计算机专业教材编审出版规划。

本书就是规划中配套教材之一。

这批书稿都是通过教学实践，从师生反映较好的讲义中经学校选报，编委会评选择优推荐或认真遴选主编人，进行约编的。广大编审者，编委和出版社编辑为确保教材质量和如期出版，作出了不懈的努力。

限于水平和经验，编审与出版工作中的缺点和不足在所难免，望使用学校和广大师生提出批评建议。

中国计算机学会教育委员会大专教育学会
电子工业出版社

附：先后参加全国大专计算机教材编审工作和参加全国大专计算机教育学会学术活动的学校名单：

上海科技高等专科学校
上海第二工业大学
上海科技大学
上海机械高等专科学校
上海化工高等专科学校
复旦大学
南京大学
上海交通大学
南京航空航天大学
扬州大学工学院
济南交通专科学校
山东大学
苏州市职工大学
国营 734 厂职工大学
南京动力高等专科学校
南京机械高等专科学校
南京金陵职业大学
南京建筑工程学院
长春大学
哈尔滨工业大学
南京理工大学
上海冶金高等专科学校
杭州电子工业学院
上海电视大学
吉林电气化专科学校
连云港化学矿业专科学校
电子工业部第 47 研究所职工大学
福建漳州大学
扬州工业专科学校
连云港职工大学
沈阳黄金学院
鞍钢职工工学院
天津商学院
国营 738 厂职工大学
北京广播电视大学
天津职业技术师范学院
天津市计算机研究所职工大学
山西大众机械厂职工大学
河北邯郸大学
沈阳机电专科学校
北京燕山职工大学
国营 761 厂职工大学
山西太原市太原大学
大连师范专科学校
江苏无锡江南大学
上海轻工专科学校
上海仪表职工大学
常州电子职工大学
国营 774 厂职工大学
西安电子科技大学
电子科技大学
河南新乡机械专科学校
河南洛阳大学
郑州粮食学院
江汉大学
武钢职工大学
湖北襄樊大学
郑州纺织机电专科学校
河北张家口大学
河南新乡纺织职工大学
河南新乡市平原大学
河南安阳大学
河南洛阳建材专科学校
开封大学
湖北宜昌职业大学
中南工业大学
国防科技大学
湖南大学

湖南计算机高等专科学校
中国保险管理干部学院
湖南税务高等专科学校
湖南二轻职工大学
湖南科技大学
湖南怀化师范专科学校
湘穗电脑学院
湖南纺织专科学校
湖南邵阳工业专科学校
湖南湘潭机电专科学校
湖南株洲大学
湖南岳阳大学
湖南商业专科学校
长沙大学
长沙基础大学
湖南零陵师范专科学校
湖北鄂州职业大学
湖北十堰大学
贵阳建筑大学
广东佛山大学
广东韶关大学
西北工业大学
北京理工大学
华中工学院汉口分院
烟台大学计算机系
安徽省安庆石油化工总厂职工大学
湖北沙市卫生职工医学院
化工部石家庄管理干部学院
西安市西北电业职工大学
湖南邵阳师范专科学校

前　言

《数据结构》是计算机专业的技术基础课。如何根据实际应用的要求，对数据进行有效地组织、存储和处理，进而编制出相应的算法，是数据结构这门课程所要研究并要加以解决的问题。通过数据结构课程的学习，应使学生能运用数据结构的知识和技巧更好地进行算法和程序的设计，也为学习操作系统和数据库等后继课程打下良好的基础。

全书共分八章。第一章绪论，介绍数据结构和算法的基本概念和常用术语；第二章至第六章介绍基本的数据结构如线性表、堆栈、队列、串、数组和树、图结构，分别讨论了数据的逻辑结构和存储结构，以及相应运算的算法；最后两章为查找和排序，介绍了常用的几种查找方法和内部排序方法。本书讲授时数为60学时左右。本教材也可作为大专院校非电子类专业学生学习数据结构的选修教材和培训教材。还可作为从事计算机工作的技术人员自学或参加计算机各级等级考试的参考用书。

本书由河海大学计算机及信息工程学院濮伯泉教授主审。濮伯泉教授在百忙当中，对全部书稿进行认真仔细地审定，提出了宝贵的意见及独到的见解，在此表示真挚的谢意。同时向曾对本书进行过指导的东南大学计算机系孙志辉教授表示由衷的感谢。

本书第一章绪论、第二章线性表、第三章串、第五章树和第七章第二节树表查找由谈春媛编写；第四章数组和广义表、第六章图、第七章查找和第八章排序由江红编写。谈春媛统编全稿。由于作者水平有限，书中难免有误，希望读者批评指正。

作者　　一九九六年于南京

第一章　绪　　论

自从世界上第一台计算机问世以来，计算机科学和计算机软、硬件技术得到飞速地发展，计算机的应用领域也从最初的科学计算逐步发展到人类社会的各个领域。计算机加工处理的对象由简单的数值、字符发展到文字、图像、声音等各种复杂的带有不同类型及相互有不同关联的数据。为了更好地进行程序设计，必须要深入地研究待处理的数据的特性以及数据之间存在的关系，这就是“数据结构”这门学科形成和发展的背景。

第一节　数据结构与算法

一、基本概念

数据(data)是对客观事物所进行的描述，而这种描述是采用了计算机系统能够识别、存储和处理的形式来进行的。数据是计算机程序加工处理的对象。例如一个编译程序所处理的数据是字符串。对计算机而言，数据的含义是很广泛的，其形式可有多种如图形、声音、光和电等等。

数据元素(data element)是数据的基本单位，即数据集合中的个体。在计算机程序中通常作为一个整体进行考虑和处理。每一个数据元素可以是单项数据，也可以由多项数据复合组成。数据项是数据的不可分割的最小单位，也是数据集合的最小可命名单位。例如：在学生档案管理系统中，可以把一个学生的有关信息作为一个数据元素，它由学号、姓名、年龄等数据项组成。

数据对象(data object)是性质相同的数据元素的集合，是数据集合的一个子集。例如整型数据的对象是集合{0, ±1, ±2, …}，字母字符的数据对象是集合{'A','B',...,'Z','a','b',...,'z'}。

数据结构(data structure)是相互之间存在一种或多种特定关系的数据元素的集合。数据结构要研究的数据不是一、二个孤立的数据，而是一批相互关联的数据。描述数据元素之间存在的相互关系的方法称为结构。抽象化地描述数据元素之间的相互关系称为数据的逻辑结构，这种相互关系可用一组运算及相应的运算规则来描述。通常，把这种数据逻辑结构简称为数据结构。数据的物理结构是指数据的逻辑结构在计算机中的表示(或映象)，它包含数据元素的映象和关系的映象。通常，把数据的物理结构称为存储结构。

从集合论的观点出发，可以对数据结构作出形式化的描述。数据结构是一个二元组，即

D.S = (D,R)

其中：D是数据元素的有限集，R是D上关系的有限集。

例如：复数可被定义为一种数据结构

Complex = (D,R)

其中：D = {x | x 是实数}；R = { < x,y > | x,y∈D，x 称为实部，y 称为虚部}。

又如，某城市的电话号码本可定义为一种数据结构

telephe_book = (D,R)

其中，$D=\{(a_1,b_1),(a_2,b_2),\cdots,(a_n,b_n)\}$

n 为用户个数，a_i 为用户名，b_i 为其电话号码。

$$R=\{<N_i,N_{i+1}>\mid N_i,N_{i+1}\in D,i=1,2,\ldots,n-1\}$$

$$N_i=(a_i,b_i)$$

可以定义若干运算及相应的运算规则。如进行查找，插入或删除某个数据元素等。这样，就构成了上述电话号码本的数据结构。

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**(bit)。可以用一个由若干位组合起来形成的一个位串表示一个数据元素，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**。结点的域按其性质可以分成两大类，一类是存放数据元素本身的域，一般称为**数据域**；另一类用来存放该结点与其它结点的连接信息，一般称为**指针域**。因此，元素或结点可看成是数据元素在计算机中的映象。

数据元素之间的关系在计算机中有两种不同的表示方法：顺序映象和非顺序映象，并由此得到两种不同的存储结构：顺序存储结构和链式存储结构。顺序映象的特点是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；非顺序映象的特点是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。数据的逻辑结构和物理结构是密切相关的两个方面。任何一个算法的设计取决于选定的数据的逻辑结构，而算法的实现依赖于采用的存储结构。

数据类型(data type)是指某种程序设计语言所允许使用的变量种类。如在 PASCAL 语言和 C 语言中，有整型、实型等简单数据类型，还有数组、记录等复杂数据类型和指针类型。一个数据类型不仅定义了相应变量可以设定的值的集合和存储方法，而且还规定了对变量允许进行的一组运算及其规则。所以，我们可以把数据类型看作是程序设计语言中已经实现了的数据结构。

随着计算机科学的不断发展，特别是面向对象的程序设计语言的研究和发展，提出了抽象数据类型(Abstract Data Type，简记 ADT)的概念。通俗地说，抽象数据类型是程序设计语言中数据类型概念的推广，是定义了一组运算的数学模型。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，即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。

二、算法的概念和特性

算法(algorithm)是对特定问题求解步骤的描述，它是指令的有限序列。在计算机科学中，算法则是描述计算机解决给定问题的操作过程。此外，一个算法还具有下列性质。

1. **有穷性**　一个算法必须在执行有限的步骤以后结束。

2. **确定性**　算法的每一步骤必须确切定义，无二义性。对于每种情况，有待执行的运算必须被严格地和清楚地规定。

3. **可行性**　算法的每一条指令都必须是基本的，可执行的，原则上可由人仅用笔和纸做有穷项运算或用计算机可实现。

4. **输入**　一个算法有零个或零个以上的输入，它们是在算法开始前对算法给定的量。这些输入取自于特定的对象的集合。

5. **输出**　一个算法有 1 个或 1 个以上的输出，它们是与输入有某种特定关系的量。

算法的设计可以避开具体的计算机程序设计语言，但算法的实现必须借助程序设计语言

中提供的数据类型及其算法。数据结构和算法是计算机科学的两个重要支柱。它们是一个不可分割的整体。著名的计算机科学家,PASCAL 语言的发明者 N. 沃思(Niklaus Wirth)教授曾提出一个有名的公式:

算法 + 数据结构 = 程序

它清楚地揭示了算法和数据结构这两个计算机科学的重要支柱的重要性和统一性,不能离开数据结构去抽象地分析求解的算法,也不能脱离算法去孤立地研究程序的数据结构。

第二节　算法的描述和算法分析

数据结构的表示和操作都涉及到算法,如何描述数据的结构和讨论有关的算法,又涉及到程序设计语言。C 语言不仅具有丰富的数据类型,也是一种较好地体现结构程序设计原则的语言。在计算机的各个应用领域中,C 语言被作为系统开发和应用的首选语言。本书采用 C 语言描述数据结构及相应的算法,其中算法可以直接应用于实际。描述算法的书写规则如下:

1. 所有算法均以函数形式给出,算法的输入数据一般来自参数表。每个算法都是功能相对独立的模块,并且具有较强的实用价值。

2. 参数表中的参数在算法之前均进行类型说明。

3. 有关结点结构的类型定义,全局变量的说明等均在算法之前进行说明。

对算法进行分析和评估,通常在算法正确性的前提下考虑下面几个方面:一是根据算法编制成程序后在计算机中运行时所耗费的时间;二是根据算法编制成程序后在计算机中所占有存储量的大小,其中主要考虑程序运行时需辅助存储量的大小;其它方面如算法是否易读,是否容易转换成任何其它语言编制的可执行的程序,以及是否易被测试等等。

一个程序在计算机上运行时所耗费的时间由下列因素所决定:程序运行时所需输入的数据总量,对源程序进行编译所需时间,计算机执行每条指令所需时间,程序中的指令重复执行的次数。前三条取决于实现算法的计算机软、硬件系统,习惯上常常把语句重复执行的次数作为算法运行时间的相对量度,称做算法的时间复杂性。

若解决一个问题的规模为 n,例如在树结构中,n 表示树中的结点数;在线性表中,n 表示数据元素的个数;在排序问题中,n 表示待排序元素的个数。那么,算法的时间复杂性就是 n 的一个函数,通常记为 T(n)。

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

f(n)表示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,上式表示随问题规模 n 的增大算法执行时间的增长率和 f(n)的增长率相同。f(n)和 T(n) 是同数量级的函数,大写字母 O 表示的意思是指 f(n)同 T(n)只相差一个常数倍。

算法的时间复杂性采用数量级的形式表示后,将对求一个算法的 T(n)带来很大方便,这时,只需要分析影响一个算法运行时间的主要部分即可,不必对每一步都进行详细地分析;同时,对主要部分的分析也可简化,一般只要分析清楚循环体内简单操作的执行次数即可。如对下面三个简单程序段:

1. `x++;`

2. `for(i=1; i<=n; i++) x++;`

3. `for(i=1; i<=n; i++) for(j=1; j<=n; j++) x=x+1;`

在程序段 1 中,语句 x++ 不包含在任何循环中,则此语句只执行一次,其时间复杂性为

O(1);在程序段 2 中,基本操作 x++ 在 for 循环中,根据循环次数 n,就可求出其时间复杂性为 O(n);同理,在程序段 3 中,基本操作 x++ 在两个 for 循环中。它的重复执行次数为 $n \times n$,则时间复杂性为 $O(n^2)$。

一个算法的时间复杂性除了与问题的规模 n 有关外,还与输入的具体数据以及多数据情况下的输入次序有关,当输入的具体数据与次序不同时,其算法的时间复杂性也可能不同。对这类算法,除特别指明外,均根据各种可能出现的数据集中最坏的情况来估算算法的时间复杂性,有时也在某种约定下讨论算法的平均时间复杂性。

算法在运行过程中需辅助存储空间的大小称为算法的空间复杂性,用 S(n) = O(f(n))表示,算法在运行过程中临时占用的辅助存储空间随算法的不同而异,有的算法只需占用少量的临时工作单元,而且不随问题规模的大小而改变;有的算法需要占用的临时工作单元数随着问题规模 n 的增大而增大,此时按最坏情况来分析。

习 题 一

一、什么是算法(algorithm),它与程序(procedure)有何区别?

二、请计算下列程序段中 x++ 的执行次数。

1.
```
for(i = 1; j <= n; i++)
    for(j = 1; j <= n; j++)
      for(k = 1; k <= n; k++)
        x++;
```

2.
```
i = 1;
while(i <= n){ x++; i++;}
```

3.
```
for(i = 1; i <= n; i++)
    for(j = 1; j <= i; j++)
      for(k = 1; k <= j; k++)
        x++;
```

4.
```
for(i = 1; i <= n; i++)
  { j = i;
    for(k = j + 1; k <= n; k++) x++;
  }
```

第二章　线　性　表

从本章至第四章开始研究线性结构。线性结构的特点是在数据元素的非空有限集中：1.存在唯一的一个被称作“第一个”的数据元素；2.存在唯一的一个称作“最后一个”的数据元素；3.除第一个外，集合中的每一个数据元素均只有一个直接前趋；4.除最后一个外，集合中的每一个数据元素都只有一个直接后继。

线性表(lines list)是一种最基本的、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。本章首先介绍它们的逻辑结构和物理结构。并重点讨论对于不同物理结构的线性表的插入和删除运算。最后讨论线性表应用的典型例子，即一元多项式的存储和相加。

第一节　线性表的逻辑结构

以下是几个线性表的例子：

例 2.1　(1,2,3,4,5)是一个线性表。其中的数据元素是数字，共有五个数据元素。

例 2.2　(A,B,C,...,Z)是一个线性表。其中的数据元素是英文大写字母，共有二十六个数据元素。

例 2.3　图 2-1 所示的学生《数据结构》成绩表也是一个线性表，其中的数据元素是每个学生所对应的一行信息，它是由姓名、学号、性别、成绩共四个数据项组成。通常把这种数据元素称为记录，含有大量记录的线性表又称文件。

学号	姓名	性别	成绩
001	陈华	男	85
002	王丽丽	女	74
003	沈国辉	男	93
⋮	⋮	⋮	⋮

图 2-1　某班学生《数据结构》成绩表

综合上述三个例子可对线性表作如下描述。

一个线性表是 n 个数据元素 $a_1,a_2,\ldots,a_n$ 的有限序列。表中每个数据元素，除第一个和最后一个外，有且仅有一个直接前趋和一个直接后继。也就是说，线性表可写成如下形式

$$(a_1,a_2,\ldots,a_i,\ldots,a_n)$$

其中，a_i 是属于某个数据对象的元素。线性表中所有元素的性质是相同的。a_1 是第一个数据元素，a_n 是最后一个数据元素。数据元素在表中的位置只取决于它自身的序号，数据元素之间的相邻关系是线性的。即 a_{i-1} 领先于 a_i，a_i 领先于 a_{i+1}，则称 a_{i-1} 是 a_i 的直接前趋元素，a_{i+1} 是 a_i 的直接后继元素。线性表中数据元素的个数 $n(n\geqslant 0)$ 定义为线性表的长度，$n=0$ 时称为空表。

对于线性表进行的基本操作有如下几种：

1. **存取**　指存取或更新线性表中第 i 个数据元素；
2. **插入**　在线性表的第 i 个元素前，插入一个新的数据元素；
3. **删除**　删除线性表中第 i 个数据元素；

4. **合并** 将二个或二个以上的线性表合并成一个线性表；

5. **分解** 将一个线性表拆成多个线性表；

6. **查找** 在线性表中查找满足某种条件的数据元素，如找出某个数据项具有给定值的数据元素；

7. **排序** 对线性表中的数据元素按其中一个数据项的值（如关键字）递增或递减的次序重新进行排列；

8. **求线性表的长度**

第二节 线性表的顺序存储结构

一、顺序分配

在计算机内，可以用不同的方式表示线性表，其中最简单和最常用的方式是顺序存储结构，即用一组地址连续的存储单元依次存放线性表的数据元素，这就是线性表的顺序分配。

假设线性表的每个数据元素需占用 l 个存储单元，并以所占的第一个单元的存储地址作为数据元素的存储位置，则第 i 个数据元素的存储位置为：

$$LOC(a_i) = LOC(a_1) + (i-1) * l \tag{2-1}$$

式中 $LOC(a_1)$ 是线性表的第一个数据元素 a_1 的存储位置，通常称做线性表的起始位置或者称作首地址。如图 2-2 所示。

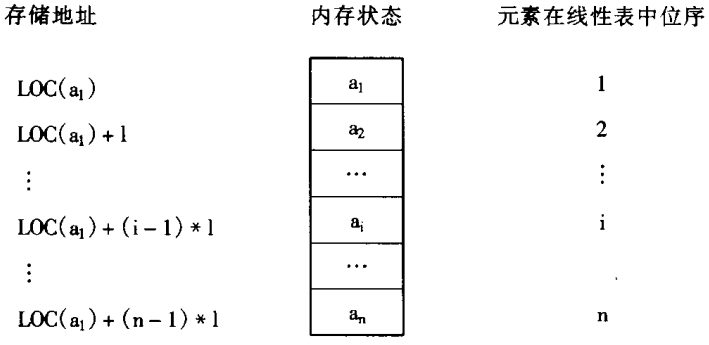


图 2-2 线性表的顺序存储结构示意图

线性表的这种机内表示称作线性表的顺序存储结构或顺序映象。

顺序存储结构的特点是，表中逻辑上相邻的数据元素存储在相邻的存储位置，换句话说，以元素在计算机内“物理位置相邻”来表示线性表中数据元素之间的逻辑关系。对于这种存储方式，只要确定了存储线性表的起始位置，线性表中任一数据元素都可随机存取，所以线性表的顺序存储结构是一种随机存取的存储结构。

顺序表可用 C 语言的一维数组实现，数组的类型随数据元素的性质而定。

为简单起见，以后若不作特别说明，数据元素的取值范围均为整数，即可用 C 语言的整型数组描述。

```
# define MAX 100
int v[MAX];
```

数组定义为 v，该数组有 MAX 个存储单元（MAX = 100），下标从 0 开始。设线性表为 $(a_1, a_2, a_3, \ldots, a_n)$，$n < MAX$，一个数据元素占用一个存储单元。线性表在数组中的存储示意图为

图 2-3。其中，从 v[n]开始到 v[MAX - 1]是备用空间，数据元素的存储位置可用数组的下标值来表示。

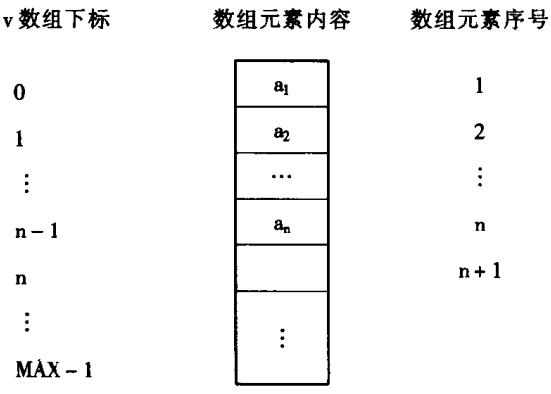


图 2-3　线性表的数组实现示意图

二、线性表的操作

在线性表的顺序存储结构中，线性表的某些操作容易实现。下面着重讨论线性表的插入和删除操作。

(一) 插入

线性表的插入操作是指在线性表的第 $i(1 \leqslant i \leqslant n+1)$ 个数据元素之前插入一个新的数据元素 x，使长度为 n 的线性表

$$(a_1, \ldots, a_{i-1}, a_i, \ldots, a_n)$$

变成长度为 n + 1 的线性表

$$(a_1, \ldots, a_{i-1}, x, a_i, \ldots, a_n)$$

数据元素 a_{i-1}和 a_i 之间的逻辑关系发生了变化。在线性表的顺序存储结构中，由于逻辑上相邻的数据元素在物理位置上也是相邻的，因此，除非 i = n + 1，否则必须移动元素才能反映这个逻辑关系的变化。

例如，图 2-4 表示了一个线性表在进行插入操作前和后的情况，其数据元素在向量空间中的位置变化。为了在线性表的第 5 个元素之前插入一个值为 25 的数据元素，则需将第 5 个至第 8 个数据元素在向量空间中依次往后移动一个位置。

一般情况下，在第 $i(1 \leqslant i \leqslant n)$个元素之前插入一个元素时，需将第 n 至第 i(共 n - i + 1)个元素向后移动一个位置。

在顺序表插入算法中，p 是指针变量，指向存放表长的变量 n。插入成功，函数返回值为 1，否则函数返回值为 0，算法如下。

算法 2.1　顺序表插入

```
/* Algorithm 2.1 Insert sequential list */
int ins_slist(int i, int x, int v[], int *p);
{ /* 在顺序存储结构的线性表 v 中，第 i 个数据元素之前插入数据元素 x, */
  /* p 为指向存放表长的变量 n 的指针变量。 */
  int j,n;
```

```
    n = * p;
    if((i < 1) || (i > n + 1)) return(0);
    else
    { for(j = n; j >= i; j -- ) v[j] = v[j - 1];
      v[j] = x;
      * p = ++ n;
      return(1);
} }
```

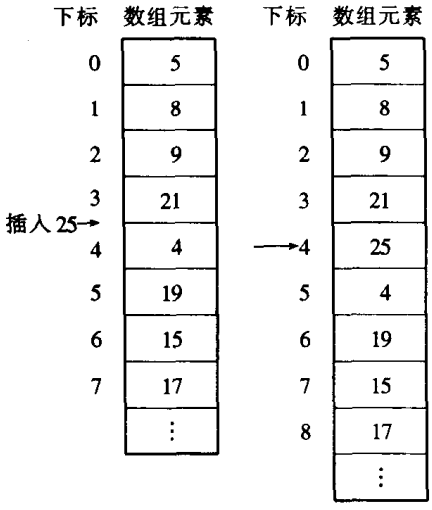


图 2-4 线性表插入前后的状况

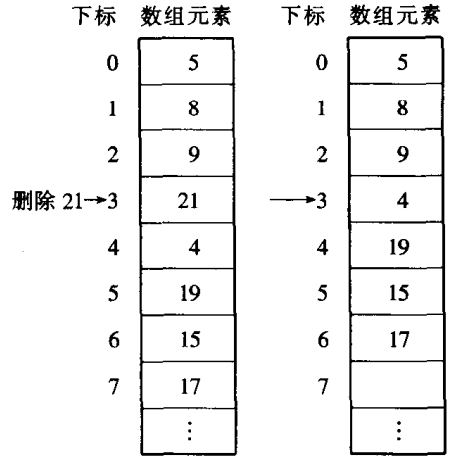


图 2-5 线性表删除前后的状况

(二) 删除

线性表的删除是指将表中第 i 个数据元素删去,使长度为 n 的线性表

$$(a_1, a_2, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n)$$

变成长度为 n - 1 的线性表

$$(a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$$

数据元素 a_{i-1}, a_i 和 a_{i+1}之间的逻辑关系发生变化,为了在存储结构上反映这个变化,同样需要移动元素。如图 2-5 所示,为了删除第 4 个数据元素,必须将从第 5 个至第 8 个元素都依次往前移动一个位置。

一般情况下,删除第 i($1 \leqslant i \leqslant n$)个元素时需将第 i + 1 至第 n(共 n - i)个元素依次向前移动一个位置,删除算法如下。

算法 2.2 顺序表删除

```
/* Algorithm 2.2 Delete seqential list */
int del_slist(int i, int v[], int * p)
{ /* 在顺序存储结构的线性表 v 中删除第 i 个数据元素 */
  int j,n;
  n = * p;
  if(i < 1 || i > n) return(0);
  else
```

```
{ for (j = i + 1;j < n;j++ ) v[j - 1] = v[j];
  * p = --n;
  return(1);
} }
```

(三) 插入、删除运算的时间分析

从线性表的顺序存储结构的插入、删除算法可见,当在顺序表中某个位置上插入或删除一个数据元素时,其时间主要花费在移动元素上,而移动元素的个数取决于插入或删除元素的位置。

假设,p_i 是在第 i 个元素之前插入一个元素的概率,则在长度为 n 的线性表中插入一个元素时所需移动数据元素的平均次数为

$$E_i = \sum_{i=1}^{n+1} p_i(n - i + 1) \tag{2-2}$$

同理,假设 q_i 是删除第 i 个元素的概率,则在长度为 n 的线性表中删除一个元素时所需移动数据元素的平均次数为

$$E_d = \sum_{i=1}^{n} q_i(n - i) \tag{2-3}$$

为了不失一般性,可以假定在线性表的任何位置上插入或删除元素都是等概率的,即 $p_i = \frac{1}{n+1}$,$q_i = \frac{1}{n}$,则式(2-2)和(2-3)可分别做如下化简

$$E_i = \frac{1}{n+1} \sum_{i=1}^{n+1} (n - i + 1) = \frac{n}{2} \tag{2-4}$$

$$E_d = \frac{1}{n} \sum_{i=1}^{n} (n - i) = \frac{n+1}{2} \tag{2-5}$$

由此可见,在顺序存储结构的线性表中插入或删除一个数据元素,平均所需移动表中一半元素。若表长为 n,则算法 ins_slist 和 del_slist 的时间复杂性为 O(n),所以,当 n 很大时,算法的效率是很低的。

第三节　线性表的链式存储结构

线性表的顺序分配结构简单、直观,便于随机存取表中任一数据元素,但作插入或删除操作时,需移动大量的元素。由于顺序表需要一组连续的存储单元,对于长度可变的线性表就需要预先分配足够的空间,有可能使一部分存储空间长期闲置不能充分利用,还可能造成表的容量难以扩充。为了克服上述缺点,本节讨论另一种存储结构——链式存储结构。

一、线性链表

线性表的链式存储结构的特点是用一组任意的存储单元存储线性表的数据元素(这组存储单元可以是连续的,也可以是不连续的)。因此,为了表示每个数据元素 a_i 与其直接后继的数据元素 a_{i+1} 之间的逻辑关系,对数据元素 a_i 来说,除了存储数据元素自身信息外,还需存储一个指示其后继元素的信息,这信息即为直接后继元素的存储位置。这两部分信息组成数据元素 a_i 的存储映象,称为**结点**。一个结点由两个域组成:存放数据元素信息的数据域和存放直接后继结点存储位置的指针域。指针域中存储的信息称作**指针**或**链**。用指针相连接的结点

序列称为**链表**，即为线性表$(a_1, a_2, a_3, \ldots, a_n)$的链式存储结构。若链表中的每个结点只含有一个指针域，则称此链表为**线性链表**或**单链表**。

例如，图 2-6 所示为线性表(5,8,9,21,4,19,15,17)的线性链表存储结构，整个链表的存取必须从头指针开始进行，头指针指示链表中第一个结点(即第一个数据元素的存储映象)的存储位置。同时，由于最后一个数据元素没有直接后继，则线性链表中最后一个结点的指针为空，用∧或 NULL 表示。若线性表为空表时，则头指针为空。

用线性链表表示线性表时，数据元素之间的逻辑关系是由结点中的指针指示的，逻辑上相邻的两个数据元素其存储的物理地址不要求相邻，由此，这种存储结构为非顺序映象或链式映象。

通常把链表画成用箭头相链接的结点的序列，结点之间的箭头表示链域中的指针，如图 2-6的线性表可画成如图 2-7 所示的形式，其中 head 是头指针，指示链表中第一个结点的存储位置。

头指针 h

31

存储地址	数据域	指针域
1	21	43
7	8	13
13	9	1
19	17	∧
25	19	37
31	5	7
37	15	19
43	4	25

图 2-6　线性链表示例

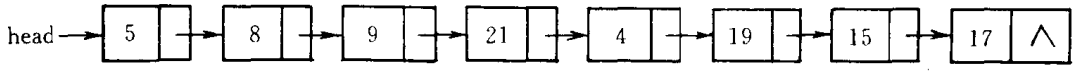


图 2-7　线性链表的逻辑状态

由此可见，单链表可由头指针唯一确定，在 C 语言中可用"指针"数据类型来描述。

```
struct node { int data;
              struct node * link;
            };
typedef struct node NODE;
```

假设 head 为链表的头指针，它指向表中第一个结点。若 head 为"空"(head = NULL)，则所表示的线性表为"空"表，其长度 n 为"零"。

有时为了操作方便，在单链表的第一个结点之前增加一个表头结点，通常表头结点的结构与表中结点结构相同，头结点的数据域可以不存储任何信息，也可存储如线性表的长度等类的附加信息，头结点的指针域存放指向第一个结点的指针。如图 2-8(a)所示，此时，单链表的头指针指向头结点。若线性表为空表，则头结点的指针域为"空"，如图 2-8(b)所示。

在线性链表中，每个元素的存储位置都包含在其直接前趋结点的信息之中。假设 p 是指向线性表中第 i 个数据元素的指针，该结点称为 p 结点或结点 a_i。则 p->link 是指向第 i + 1