面向21世纪

高职高专系列教材

数据结构

◎陈付贵　主编

◎鲁　辉　审

机械工业出版社
China Machine Press

面向21世纪高职高专系列教材

数据结构

陈付贵　主编

鲁　辉　审

机械工业出版社

全书采用C语言作为数据结构和算法的描述语言，概念表达准确，逻辑推理严谨，语言精练，通俗易懂，便于教学和自学。

全书共分10章，第1章简要介绍了数据结构中的基本概念；第2章～第7章主要讨论了几种基本数据结构的逻辑特性，以及这些结构在计算机中的存储表示和有关算法及应用；第8、9两章分别介绍了查找和排序的方法及综合分析比较；第10章介绍了文件的概念及文件的组织结构。

本书可作为高职高专计算机专业和信息类相关专业的教材，也可供从事计算机工程与应用的技术人员参考。

图书在版编目（CIP）数据

数据结构/陈付贵主编. —北京：机械工业出版社，2001.6
面向21世纪高职高专系列教材
ISBN 7-111-08282-6

Ⅰ.数… Ⅱ.陈… 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（2001）第033242号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
策　　划：胡毓坚
责任编辑：田　梅
责任印制：路　琳

中国建筑工业出版社密云印刷厂印刷·新华书店北京发行所发行

2001年7月第1版·第1次印刷
1000mm×1400mmB5·6.375印张·290千字
0001－5000册
定价：20.00元

凡购本图书，如有缺页、倒页、脱页，由本社发行部调换
本社购书热线电话：（010）68993821、68326677－2527

面向21世纪高职高专
计算机专业系列教材编委会成员名单

顾问	曾玉崑　王文斌　陈瑞藻　李　奇　凌林海 林　东
主任委员	周智文
副主任委员	周岳山（常务副主任）　詹红军　陈付贵 穆天保　赵佩华　黄甘洲　武文侠　吕何新
委员	郭曙光　王德年　刘瑞新　陈丽敏　孔令瑜 李　玲　鲁　辉　陶书中　赵增敏　马　伟 孙心义　翟社平　廖常武　于恩普　王春红 王娟萍　屈　圭　汤新广　谢　川　姜国忠 汪赵强　董　勇　梁国浚　张晓婷
秘书长	胡毓坚
副秘书长	陈丽敏（兼）

出 版 说 明

积极发展高职高专教育，完善职业教育体系，是我国职业教育改革和发展的一项重要任务。为了深化职业教育的改革，推进高职高专教育的发展，培养21世纪与我国现代化建设要求相适应的，并在生产、管理、服务第一线从事技术应用、经营管理、高新技术设备运作的高级职业技术应用型人才，尽快组织一批适应高职高专教学特色的教材，已成为各高职高专院校的迫切要求。为此，机械工业出版社与高职高专计算机专业、电子技术专业和机电专业教材编委会联合组织了全国40多所院校的骨干教师，共同研究开发了一批计算机专业、电子技术专业和机电专业的高职高专系列教材。

各编委会确立了“根据高职高专学生的培养目标，强化实践能力和创新意识的培养，反映现代职业教育思想、教育方法和教育手段，造就技术实用型人才为立足点”的编写原则。力求使教材体现“定位准确、注重能力、内容创新、结构合理和叙述通俗”的编写特色。

本套系列教材是由高职高专计算机专业、电子技术专业、机电专业教材编委会分别会同各院校第一线专业教师针对高职高专计算机、电子技术和机电各专业的教学现状和教材存在的问题开展研讨，尤其针对目前高职高专教学改革的新情况，分别拟定各专业的课程设置计划和教材选题计划。在教材的编制中，将教学改革力度比较大、内容新颖、有创新精神、比较适合教学、需要修编的教材以及院校急需、适合社会经济发展的新选题优先列入选题规划。在广泛征集意见及充分讨论的基础上，由各编委会确定每个选题的编写大纲和编审人员，实行主编负责制，编委会通过责任编委和主审对教材进行质量监控。

担任本套教材编写的老师们都是来自各高职高专院校教育第一线的教师，他们以高度的责任感和使命感，经过近一年的努力，终于将本套教材呈现在广大读者面前。由于高职高专教育还处于起步阶段，加上我们的水平和经验有限，在教材的选题和编审中可能出现这样那样的问题，希望使用这套教材的教师和学生提出宝贵的意见和建议，以利我们今后不断改进，为我国的高职高专教育事业的繁荣而共同努力。

高职高专系列教材编委会
机 械 工 业 出 版 社

前　言

“数据结构”是计算机类专业的重要理论基础，是计算机学科的核心课程。其培养目标是：学会分析研究计算机处理的数据结构的特性，为物理问题的数据选择适当的逻辑结构、存储结构和相应的算法；并初步掌握算法的时间分析和空间分析的技术。

本书的第 1 章介绍了数据、数据结构等基本概念；第 2 章～第 7 章分别讨论了线性表、栈、队列、串、数组、树和二叉树以及图等基本类型的数据结构及其应用；第 8 章和第 9 章讨论了查找和排序，除了介绍各种实现方法之外，还着重从时间上进行定性或定量的分析和比较；第 10 章介绍了常用的文件结构。

全书以 C 语言作为数据结构和算法的描述语言，讨论了基本类型的数据结构及其应用；表述了查找和排序的实现方法及其综合分析比较。本书具有以下特点：

1．内容选取符合教学大纲基本要求，兼顾不同学科的广度和深度，适应面广；

2．内容精练，系统性强；

3．概念描述准确，逻辑推理严谨；

4．图文并茂，通俗易懂。

本书由陈付贵教授担任主编，鲁辉审稿。其中陈付贵编写第 1 章～第 3 章；张冀红编写第 4 章～第 6 章；孙立威编写第 7 章～第 10 章。在本书的编写出版过程中，得到了周智文、周岳山老师及机械工业出版社、上海电子技术学校有关领导的大力支持，在此一并致谢。

由于水平所限，本书中难免出现错误和缺点，恳切希望读者批评和指正。

作　者

目　录

第1章 引 论

人类社会进入21世纪，计算机的应用已不再局限于科学计算，而更多地用于控制、管理及数据处理等非数值计算的处理工作。与此相应，计算机加工处理的对象由纯粹的数值发展到字符、表格和图像等非数值对象，仅凭程序设计员的经验和技巧已难以设计出效率高、可靠性强的程序。于是，就要求对计算机程序加工的对象进行系统的研究，即研究处理对象的特性以及对象之间存在的关系。

1.1 数据结构的概念

用计算机解决一个问题的关键是从具体问题抽象出一个适当的数学模型，寻求数学模型的实质是分析问题，从中提取操作的对象，找出这些操作对象之间的内在关系，并用数学语言加以描述。

假定有一个学生花名册，记录了某学校所有学生的名称及其他相关信息，现设计一个算法，要求是：当给定任一学生的名称时，计算机能够查出该学生的有关信息，如果没有该学生，计算机也能报告“查无此人”。

如果学生花名册中的姓名是随机排列的，其次序没有任何规律（见表1-1）。那么，当给定一个学生姓名时，就只能从花名册中第一个姓名开始，逐个与给定的姓名进行比较（即线性查找），直至找到相应的学生，这种方法虽然简单，但相当费时，效率较低。

表1-1 某学校学生花名册

姓名	性别	系别	籍 贯
唐珍	女	计科系	河南洛阳
张章	男	经管系	广东梅县
李林	女	计科系	河北邯郸
⋮	⋮	⋮	⋮

如果对学生花名册进行适当的组织，比如按学生所在的系别进行排列，并且再构造一个索引表，这个表用来登记每个系的学生在学生花名册中的起始位置。这样，要查找某学生的信息时，则可先从索引表中查找该学生所在系的学生姓名是从何处开始，然后再从开始处进行查找，而不必查看花名册的其他部分。

由上述的讨论可以看出，“查找”的算法设计完全依赖于学生花名册的组织方式，这就是一个数据结构问题。不同的数据结构，可得出不同的算法，算法与数据结构密切相关——每一个算法无不依赖于具体的数据结构，数据结构直接影响着算法的选择和效率。

下面对学生花名册再作进一步讨论。当有新学生进校时，花名册中需要增添新学生的姓名和相关信息；学生毕业后，应从花名册中删除毕业学生的相关信息。因此，这就要

求在原有的花名册上进行插入(insert)和删除(delete)运算。对于一种已生成的结构,如何实现插入和删除运算,把要增添学生的内容插入到表头还是表尾,或者是中间某个合适的位置上,这都要由具体的数据结构决定。此外,插入后,对原有的数据是否有影响,有什么样的影响,删除某学生的信息以后,其他学生的信息在花名册中的位置是否要跟着变动,若需要变动,应如何进行。这一系列的问题说明,为适应数据增减的需要,还必须对每种数据结构定义一套运算。上面只涉及了插入和删除运算,但在实际问题中,还会有一些其他可能的运算。如修改(modify)运算,这就要求我们设计出相应的插入、删除、修改等算法。这些运算均是由算法来完成的。也就是说,数据结构还必须给出每种结构类型所定义各种运算的相应算法。

"数据结构"是一门研究非数值计算的程序设计问题中操作对象以及它们之间的关系和操作的学科。它不仅是计算机专业教学计划中的核心课程,而且是其他非计算机专业的主要选修课程之一。其研究领域不仅涉及到计算机硬件(特别是编码理论、存储装置和存取方法等),同时和计算机软件的研究有着密切的关系。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,是一般程序设计(特别是非数值计算的程序设计)的基础,而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序和大型应用程序的重要基础。

1.2 基本概念和术语

数据(Data)是对客观事物的符号表示,在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称。例如,一个利用数值分析方法解代数方程的程序,其处理对象是整数和实数;一个编译程序或文字处理程序的处理对象是字符串。因此,对计算机科学而言,数据的含义极为广泛,如图像、声音等都可以通过编码而归之于数据的范畴。

数据元素(Data Element)是数据的基本单位,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只有一个数据项,也可以由若干个数据项组成。例如,在学生档案管理系统中,每个学生的情况用一个数据元素表示,而其中的姓名、性别、年龄……则分别为一个数据项。数据项是数据的最小单位。数据元素的同义语有:结点(Node)、顶点(Vertex)和记录(Record)等,在顺序结构中多用"元素",在链式结构中多用"结点",而在图和文件中又分别使用"顶点"和"记录"等术语。

数据对象(Data Object)是性质相同的数据元素的集合。例如,整数数据对象是集合 $N = \{0, \pm 1, \pm 2, \ldots\}$,字母字符数据对象是集合 $C = \{'A', 'B', \cdots, 'Z'\}$。

数据处理(Data Processing)是指对数据进行查找、插入、删除、合并、排序、统计、简单计算、输入、输出等的操作过程。像办公室自动化系统、情报检索系统、经济信息管理系统、物资调配系统、财务管理系统等都是计算机在数据处理方面的具体应用。

数据结构(Data Structure)是指数据以及数据之间的联系和这种联系在计算机中的存储表示。数据结构包括两方面的内容,数据的逻辑结构(Logical Structure)和数据的物理结构(Physical Structure)。在本教材中所说的数据结构专指数据的逻辑结构。

数据的逻辑结构是指各数据元素之间的逻辑关系，是用户按需要建立起来的，并呈现在用户面前的数据元素的结构形式。根据数据元素之间关系的不同特性，通常有下列四种基本结构，如图 1-1 所示。

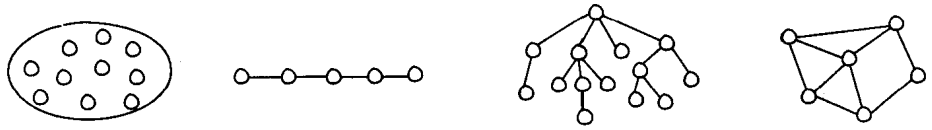

图 1-1　四种基本结构示意图

a) 集合结构　b) 线性结构　c) 树形结构　d) 网状结构

(1)集合结构：结构中的数据元素之间除了“同属于一个集合”的关系外，别无其他关系；

(2)线性结构：结构中的数据元素之间存在一个对一个的关系；

(3)树形结构：结构中的数据元素之间存在一个对多个的关系；

(4)网状结构：结构中的数据元素之间存在多个对多个的关系。

数据的物理结构(Physics Structure)又称为数据的存贮结构，是指数据之间的关系在计算机内实际的存贮形式。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，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，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。

数据结构的形式定义为：数据结构是一个二元组

$$DataStructure = (D,S)$$

其中：D 是数据元素的有限集，S 是 D 上关系的有限集。

例如：L = (P,S)其中，

$$P = \{1,2,3,4,5,6,7,8,9\}$$
$$S = \{s\}$$
$$S = \{\langle 1,2\rangle,\langle 2,3\rangle,\langle 3,4\rangle,\langle 4,5\rangle,\langle 5,6\rangle,\langle 6,7\rangle,\langle 7,8\rangle,\langle 8,9\rangle\}$$

对应的图形描述为：

$$1\rightarrow 2\rightarrow 3\rightarrow 4\rightarrow 5\rightarrow 6\rightarrow 7\rightarrow 8\rightarrow 9\rightarrow 0$$

在该数据结构中，每个元素有且只有一个前趋元素(除第一个元素外)，有且只有一个后继元素(除最后一个元素外)。这种数据结构的特点是使数据元素之间具有一对一的联系，即线性关系，具有这种特点的数据结构叫做线性数据结构，简称为线性结构。

数据类型(Data Type)数据类型是一个值的集合和定义在这个值集上的一组操作的总称。例如，在高级语言中，每个变量、常量或表达式都有一个它所属的确定的数据类型。

类型显式或隐含地规定了在程序执行期间变量或表达式所有可能取值的范围，以及在这些值上允许进行的操作。按“值”的不同特性，高级语言中的数据类型可分为两类：一类是非结构的原子类型，原子类型的值是不可分解的。如：C 语言中的基本类型（整型、实型、字符型和枚举类型）、指针类型和空类型；另一类是结构类型，结构类型的值是由若干成分按某种结构组成的，因此是可以分解的，并且它的成分可以是非结构的，也可以是结构的。例如数组的值由若干分量组成，每个分量可以是整数，也可以是数组等。

1.3 算法与算法分析

1.3.1 算法

算法（arithmetic）是求解问题的步骤，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。算法具有下列五个特性：

（1）有穷性——一个算法必须（对任何合法的输入）在执行有穷步骤之后结束，且每一步都可在有限时间内完成；

（2）确定性——算法中每一条指令必须有确切的含义，不会产生二义性理解。并在任何条件下，算法只有唯一的一条执行路径；

（3）可行性——算法中描述的操作都可以通过已经实现的基本运算执行有限次来实现；

（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。

1.3.2 算法设计的要求

一个“好”的算法应达到以下目标：

（1）正确性。正确性是设计和评价一个算法的首要条件，如果一个算法不正确，其他方面就无从谈起。一个正确的算法是指在合理的数据输入下，能在有限的运行时间内得出正确的结果。“正确”一词的含义大体可分为以下四个层次：

①程序不含语法错误；

②程序对几组输入数据能够得出满足规格说明要求的结果；

③程序对精心选择的典型、苛刻而带有刁难性的几组输入数据能够得出满足规格说明要求的结果；

④程序对于一切合法的输入数据都能得出满足规格说明要求的结果。

显然，达到第④层意义下的正确是极为困难的，因为所有不同输入数据的数量大得惊人，逐一验证的方法是不现实的。因此，通常以第③层意义的正确性作为衡量一个程序是否合格的标准。

（2）易读性。算法主要是为了人们的阅读与交流，其次才是机器执行。一个清晰的算法有助于人们对算法的理解；晦涩难懂的程序易于隐藏较多错误，难以调试和修改。

(3)健壮性。当输入数据非法时,算法也能适当地作出反应或进行处理,应返回一个表示错误或错误性质的值,而不是打印错误信息或异常终止程序的执行。

(4)高效率与低存储量需求。同一个问题如果有多个算法,执行时间短的算法效率高。存储量需求指算法执行过程中所需的最大存储空间。效率与存储量需求及问题的规模有关。显然,对 100 个数据处理与对 1000 个数据处理所花的时间和空间有一定的差别。

1.3.3 算法效率

算法效率需通过由该算法编制的程序在计算机上运行时所消耗的时间来度量。一个程序在计算机上运行时所消耗的时间取决于下列因素:

①选取何种算法;

②问题规模的大小;

③书写程序所用的语言;

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。

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,其运行时所消耗的时间均不相同。这表明使用绝对的时间来衡量算法的效率是不合适的。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)来说是基本的操作,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 $n \times n$ 矩阵相加的算法中,"加法"运算是"矩阵相加问题"的基本操作。整个算法的执行时间与该基本操作(加法)重复执行的次数 n^2 成正比。

$$\boldsymbol{A}=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \quad \boldsymbol{B}=\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{pmatrix}$$

```
for(i = 1;i< = n;i + + )
for(j = 1;j< = n;j + + )
c[i][j] = a[i][j] + b[i][j];
```

一般情况下,精确地计算出算法中基本操作的执行次数不太现实,只要大致计算出相应的数量级即可。

设 $T(n)$的一个辅助函数为 $f(n)$,当 n 大于等于某一足够大的正整数时,存在着两个正的常数 a 和 $b(a < b)$使得 $a < T(n)$和 $f(n) < b$ 均成立,则称 $f(n)$是 $T(n)$的同数量级函数,记为:

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

这表示 $f(n)$同 $T(n)$只差一个常数倍。

设 $f(n)$为问题规模的某个函数,则算法执行时间的增长率和 $f(n)$的增长率相同,

称作算法的渐近时间复杂度(Asymptotic Time Complexity),简称时间复杂度,记为 $O(f(n))$。算法的时间复杂度采用数量级的形式表示后,将给求一个算法的时间复杂度带来很大的方便。

在下列三个程序段中,

```
(a){s+ =1;}
(b)for(i=1;i<=n; i++)
    s+ =1;
(c)for(i =1; i <=n; i++)
  for(j =1; j <=n; j++)
   s+ =1;
```

若其中基本操作"$s+=1$"的语句的执行次数分别为 1、n 和 n^2,则这三个程序段的时间复杂度分别为 $O(1)$、$O(n)$和 $O(n^2)$,分别称为常量级、线性级和平方级。有些算法还呈现的时间复杂度有:对数级 $O(\log_2 n)$,指数级 $O(2^n)$等。不同数量级时间复杂度的性状如图 1-2 所示。从图中可以看出:

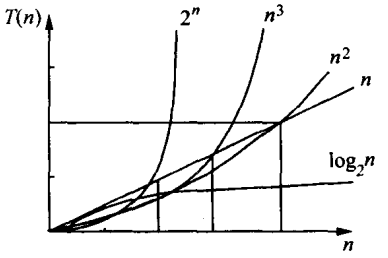

图 1-2 常见函数的增长率

(1)随着值的增大,各种 $T(n)$函数所对应的运行时间的增加速度大不相同。当 n 足够大后,各种不同数量级的 $T(n)$函数存在着下列关系:

$$O(\log_2 n) < O(n) < O(n\log_2 n) < O(n^2) < \ldots < O(2^n) < O(n!)$$

(2)当 $T(n)$或它的数量级为对数函数、幂函数或它们的乘积时,算法的运行时间是可接受的,称这些算法为有效算法,当 $T(n)$或它的数量级为指数函数或阶乘函数时,算法的运行时间是不可接受的,称这些算法是坏的或无效的算法。

一般情况下,算法中的基本操作重复执行的次数还与该问题输入的数据集有关,例如,从 n 个数据中寻找某一确定的数据,若所查找的数据恰是该组数据中的第一个(最好情况),则查找一次即可完成,若查找的恰是最后一个数据(最坏情况),则需执行 n 次,平均查找次数为 $n/2$,即平均查找的时间复杂度为 $O(n)$。多数情况下,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,常用的办法是讨论算法在最坏情况下的时间复杂度,即分析最坏情况下算法执行时间的一个上界。在本书以后各章中讨论的时间复杂度,除特别指明外,均指最坏情况下的时间复杂度。

1.3.4 算法的空间需求

类似于算法的时间复杂度,本书中以空间复杂度(*Space Complexity*)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寄存本身所用指令、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)。

1.4 小结

数据结构是指数据及数据之间的联系和这种联系在计算机中的存储表示,包括数据的逻辑结构和物理结构。数据的逻辑结构是指各数据元素之间的逻辑关系,是用户按需要建立起来的,并呈现在用户面前的数据元素的结构形式。数据的物理结构是指数据在计算机内实际的存储形式。算法的设计取决于数据的逻辑结构,算法的实现依赖于采用的物理结构。数据结构形式定义为一个二元组:

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

其中:D 是数据元素的有限集,S 是 D 上关系的有限集。

算法是求解问题的步骤,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。具有有穷性、确定性、可行性、有零个或多个输入和有一个或多个输出等特性。一个"好"的算法应达到正确性、易读性、健壮性和高效率与低存储量需求等目标。衡量一个算法的效率用时间复杂度来实现,当时间复杂度的数量级为常量级、对数函数级、线性函数级、幂函数级或它们的乘积时,为可接受算法。

本章学习要点:

(1) 掌握数据结构中有关基本概念,掌握算法的概念;

(2) 会分析算法的时间复杂度。

1.5 习题

1. 解释术语:数据、数据元素、数据结构、数据的逻辑结构和物理结构。
2. 举例说明在算法正确性中为什么达到第③层便可以了?
3. 指出下面程序中 S = S + I 的执行次数为:

```
FOR(I = 0;I < 10;I + + )
  FOR(J = 0;J < = I;J + + )
    S = S + I;
```

第2章 线 性 表

线性结构是最常用且最简单的一种数据结构，其特点是：在数据元素的非空有限集中，存在唯一的一个被称做“第一个”的数据元素；存在唯一的一个被称做“最后一个”的数据元素；除第一个元素之外，集合中的每个数据元素均只有一个直接前驱；除最后一个元素之外，集合中每个数据元素均只有一个直接后继。

2.1 线性表的类型定义

线性表(Liner-List)：线性表是若干个数据元素的有限序列。每个数据元素的具体含义，在不同的情况下各不相同，它可以是一个数、或一个符号，甚至其他更复杂的信息。

例如，26个英文字母的字母表，(A，B，C，…，Z)是一个线性表，表中的数据元素是单个字母字符。

再如，基本数字可以用线性表的形式给出：(0，1，2，3，4，5，6，7，8，9)表中的数据元素是整数的基本数字。

又如，一个学校的学生档案(如表2-1)也是一个线性表，一个数据元素可以由若干个数据项(Item)组成。这时，常把数据元素称为记录(Record)，含有大量记录的线性表又称为文件(File)。

表2-1 学生档案

学 号	姓名	性别	出生日期	班级	成绩
9910001	张峰	男	01/01/70	计991	90
9823102	李林	女	11/23/72	计992	95
…	…	…	…	…	…

表中每个学生的情况为一个数据元素(也称为记录)，它由学号、姓名、性别、出生日期、班级和成绩等数据项组成。

综上所述，线性表中的数据元素的具体含义虽不相同，但同一线性表中的元素必定具有相同特性，即属同一数据对象，相邻数据元素之间存在着序偶关系。

一般地，一个线性表是 $n \geqslant 0$ 个数据元素 $a_1, a_2, \cdots, a_n$ 的有限序列。如果 $n > 0$，则 a_1 为线性表的第一个元素，a_i 为线性表的第 i 个元素，它在数据元素 a_{i-1} 之后，在 a_{i+1} 之前，而 a_n 是线性表的最后一个元素；如果 $n = 0$，则为空表。这就是说，线性表或者是一个空表，或者可以写成：

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

称 a_{i-1} 是 a_i 的直接前驱元素，a_{i+1} 是 a_i 的直接后继元素。当 $i = 1, 2, \cdots, n-1$ 时，