SHU JU JIE GOU JIAO CHENG

数据结构教程

数据结构教程

黄育潜　滕少华

华中理工大学出版社

SHU JU JIE GOU JIAO CHENG

数据结构教程

黄育潜　滕少华

华中理工大学出版社

图书在版编目(CIP)数据

数据结构教程/黄育潜
武汉:华中理工大学出版社，1996.5
ISBN 7-5609-1285-0

Ⅰ.数…
Ⅱ.①黄… ②滕…
Ⅲ.数据结构-教材
Ⅳ.TP311.12

数据结构教程
黄育潜 滕少华 编
责任编辑 陈培斌 周筠
*
华中理工大学出版社出版发行
(武昌喻家山 邮编:430074)
新华书店湖北发行所经销
华中理工大学出版社照排室排版
武汉市青联彩印厂印刷
*
开本:787×1092 1/16 印张:14.75 字数:338 000
1996年5月第1版 1999年1月第3次印刷
印数:9 001—12 000
ISBN 7-5609-1285-0/TP·77
定价:15.00

内 容 简 介

本书用精练、流畅的语言详述了数据结构的基本概念、基本思想、基本原理及实际背景。共分十章，内容包括：绪论，线性表，栈和队列，特殊链表和特殊线性表，内、外排序，树，图，检索，文件。

书中以大量的例子来突出这样一个思想：数据结构是算法设计和描述的基础与工具，并采取了“对象描述、关键一步和总体控制”的算法讲解模式等多项化解难点的创新作法，在教学中深受学生欢迎。另外，本书采用实用的 PASCAL 语言作为数据结构和算法的描述工具，这将便于读者自学，也有利于帮助读者在今后的实践中应用所学的知识。

前　言

数据结构是计算机科学中一门研究数据元素之间关系的学科分支。计算机科学的飞速发展，使计算机从局限于解决数值计算的各种问题迅速扩展到非数值计算的各个领域，如企业管理、情报检索、文件系统、办公室自动化、数据库、人工智能等社会生活的各个领域。在这些领域中，计算机加工处理的对象已不是纯粹的数值，而是有着复杂结构关系的数据。数据结构就是研究这些数据内部的结构关系（包括逻辑结构和存储结构），研究如何对各种结构的数据定义各种运算（操作），设计各种算法，分析各种算法的效率，研究各种数据结构在计算机科学及其它重要应用领域中的应用，探讨数据查找和排序等方面的技术。

“数据结构”课程普遍被列为计算机系各专业的核心课程。因为它不仅是“操作系统”、“编译原理”、“数据库”、“算法设计与分析”、“人工智能”、“计算机图形学”等课程的基础，也是各行各业的计算机软件工作人员应具备的基本知识。

目前国内外关于数据结构的教材颇多，少说也有几十种。编者在多年教学实践中，曾采用过其中多种作为教材或教学参考书，自觉受益非浅。但在教学之中，又常常感到种种不足。本书就是编者根据多年教学经验，企图弥补不足而编写的，其主要特点是：

1. 根据学科的发展和应用，对教材内容作了认真选择和组织，如考虑到现代高级程序设计语言普遍都已实现了数组结构、提供很强的字符串处理功能及愈来愈重视指针和动态存储管理等现实，精简了有关数组和字符串结构的一些内容，加强了链表结构及其应用的内容。

2. 全书突出这样一个思想：数据结构是算法设计和描述的基础与工具。书中多次以例子说明，数据结构的选择如何影响到算法的描述和效率。

3. 在数据的存储结构讲述中，把重点放在关系的存储表示上，突出区别了关系的隐式表示和显式表示，让读者更自觉地认识到，在各种基本运算和算法的实现中，如何利用相应关系及进行相应调整。

4. 在许多独立、完整算法的讲述中，采用了按对象描述、关键一步和总体控制这种模式来进行，发现用这种模式表述算法更清晰，更便于讲清难点，更易于为学生接受和理解。

5. 抛弃了伪语言，采用了实用的 PASCAL 语言来作数据结构和算法的描述工具，事实证明，这既有利于学生上机实习，加深对所学数据结构和算法的理解，也有利于他们今后在实践中对所学知识的应用。

此外，在概念引入、例题选择、算法（程序）的形式编排和解说等方面也作了一些努力，希望能有助于读者学习和掌握本书内容，但由于编者学识有限，加上时间仓促，书中难免存在错误和缺点，恳切希望读者批评指正。

本书编写过程中，得到江西师范大学计算机系、江西师范大学计算机应用研究所、江西省计算中心以及许多教师的关心和支持，在此一并表示感谢。

编者

1996 年 3 月

目　录

第一章 绪 论

计算机解题离不开程序设计。程序设计的重要问题之一是要建立适应具体问题对象描述的各种新的数据类型。数据结构就是用于表示各种新的数据类型的重要工具。本章将概括地介绍有关的基本概念、基本思想、基本原理及实际背景。

1.1 数据结构和算法

对计算机来说，数据的含义是极为广泛的，它包括数字、字符、字符串、图形、图像、声音、表、文件等等。概括地说，凡是可以输入到计算机中并被计算机程序进行处理的各种信息，都可以称为数据。

通常，实际问题中所涉及的数据都不是单个孤立存在的，而是成组、成批出现的。经验表明，弄清问题中数据间的各种关系，从而合理地组织数据，实是设计问题求解算法(及程序)的基础。

下面通过例子对此作进一步说明。

例 1.1 求以下 5 个整数之和：

32,15,43,11,34.

本问题共涉及 6 个数据：5 个原始数据和一个结果，且结果是 5 个原始数据的算术和。依据这一关系，可以得到解此问题的一个朴素算法：

①结果变量 S 置为初值 0，即 S:=0；

②将第一个整数 32 加入 S 中，即 S:=S+32；

③将第二个整数 15 加入 S 中，即 S:=S+15；

④将第三个整数 43 加入 S 中，即 S:=S+43；

⑤将第四个整数 11 加入 S 中，即 S:=S+11；

⑥将第五个整数 34 加入 S 中，即 S:=S+34；

⑦结束。

算法结束时，结果变量 S 中的数就是所需的和数。

显然，上述算法是笨拙的。事实上，稍有程序语言知识的人都知道，若建立一个一维数组 A[1..5]，把原始数据用作此数组的初值，则可建立如下一个更简洁的算法：

①结果变量 S 置初值 0，即 S:=0；

②对 i=1,2,…,5，重复将 A[i]值加入 S 中的工作

for i:=1 **to** 5 **do** S:=S+A[i]；

本算法还有一个优点是，只需要改变数组 A 的下标界和循环终值，此算法可适用于对任意有限个整数的求和。

仔细分析后一算法，可以看到，此算法利用了问题数据之间存在的两类关系：一类是原始数据与结果之间的运算关系——整数间的四则运算且结果可通过对原始数据使用四则运算得到；另一类是原始数据之间的结构关系——原始数据组成一个一维数组，便于依次访问各数组

元素，把迭代求和运算组织在一循环中。前一算法由于没有建立原始数据间的这种结构关系，故各数据只能单个孤立地引用，导致算法的笨拙。

本例说明，合理地组织数据，建立一定的数据结构，可以简化问题的求解算法。

例 1.2 多项式相加。

众所周知，按升幂形式记写的多项式

$$P(x)=a_0+a_1x+a_2x^2+\cdots+a_{n-1}x^{n-1}+a_nx^n, a_n\neq 0$$

完全由其系数序列$(a_0,a_1,a_2,\cdots,a_{n-1},a_n)$唯一确定。如多项式

$$A(x)=7-2x+3x^2+4x^5$$

可表示为(7,−2,3,0,0,4)。通常，为了更便于处理，还显式地给出多项式最高项的次数 n，如上例 n=5。因而，在 PASCAL 语言中，如上设计的多项式结构可用如下数据类型来描述：

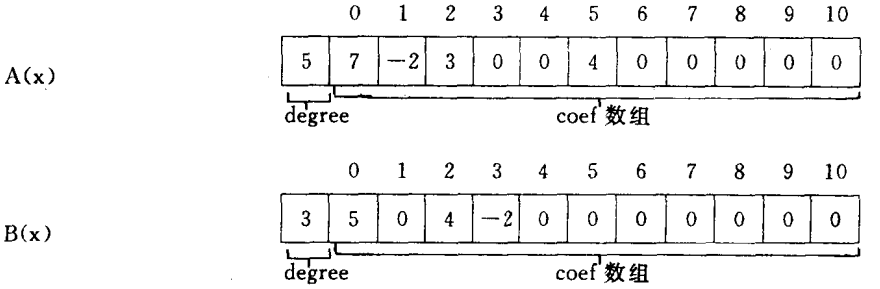
```
type poly=record
           degree:0..maxd;
           coef: array [0..maxd] of real;
         end;
```

其中，maxd 是一整型常数，表示所处理的多项式次数的上界。

设有如上多项式 A(x)及多项式

$$B(x)=5+4x^2-2x^3$$

依据上述数据结构，多项式 A(x)和 B(x)将分别表示为(设 maxd=10)：



这时，引入说明

```
var a,b,c:poly;
```

且设 a、b 中如上存放了多项式 A(x)与 B(x)的信息。现在，多项式相加就可以用以下算法来完成：

①同类项系数相加，即对 i=0,1,…,maxd，重复以下工作：

c.coef[i]:=a.coef[i]+b.coef[i];

②按以下规则确定和多项式 C(x)的次数：

$$c.degree=\begin{cases}a.degree, & 若\ a.degree>b.degree,\\ b.degree, & 若\ a.degree<b.degree,\\ \max\{i\mid c.coef[i]\neq 0\}, & 若\ a.degree=b.degree.\end{cases}$$

本问题的求解算法中，也利用了问题数据之间的两类关系：一类是原始数据与结果数据间的运算关系——实数间的四则运算，且结果可通过实数四则运算得到；另一类是数据间的结构关系——同一多项式的信息可组织成一个记录，且其各项系数按升幂组成一个一维数组，以便按多项式相加法则，将逐次进行同类项系数相加这一操作组织在一个循环中。

本书以后有关章、节中，还将介绍描述多项式对象的多种其它数据类型(数据的组织和存

储方式)。读者将发现,随着描述多项式对象的数据类型的不同,相应多项式运算的实现也将不同。这进一步说明,算法依赖于数据结构。

例1.3　职工档案管理问题。

为了简单起见,假定每个职工的档案只包括以下5个项目:工作证号、姓名、性别、职称和工资。档案管理的主要职能是查阅、复制、插入、删除和更新。为便于完成这些职能,档案管理人员通常将所有档案组织成为表1.1所示的表格形式。

表1.1　职工档案登记表

工作证号	姓名	性别	职称	工资/元
0001	陈建新	男	工程师	470
0002	张　燕	女	助　工	380
0003	李天德	男	高　工	590
0004	王小红	女	工程师	470
⋮	⋮	⋮	⋮	⋮

在用计算机解决上述管理问题时,人们可以保留数据的这种组织结构,即把每人的档案数据组织成一个记录,把一个单位或部门全体职工的档案数据组织成个人档案记录的数组(或文件)。亦即,人们可用以下数据类型来描述个人档案及档案登记表的结构:

```
type person=record
              no:integer;
              name:string[20];
              sex:(male,female);
              techpost:(assiseng,eng,higheng,…);{助工,工程师,高工,…}
              salary:real
            end;
      pstable=array [1..n] of person;
```

这时,所需的职工档案管理系统将由以下主控结构和相应的命令处理子程序组成:

```
初始化;
出口标志 Exit_flag 置为 false;
while not Exit_flag do
  bdgin
    显示命令菜单;
    提示并读入用户打入的命令;
    case 命令 of
      查阅:处理查阅命令;
      复制:处理复制命令;
      插入:处理插入命令;
      删除:处理删除命令;
      更新:处理更新命令;
      退出:处理退出命令(置 Exit_flag 为 true);
```

```
        end
    end;
```

本管理系统的运作也利用了问题数据之间的两类关系：一类是数据之间的运算关系——档案记录可查阅、复制、插入、删除、更新，及档案记录中各数据项上允许的有关操作；另一类是数据间的结构关系——档案数据按人组成记录，然后依单位或部门组成记录数组。

细心的读者可以发现，在本管理系统中，主控部分是把一个个完整的档案记录作为基本单位来操作的，不难理解，在各命令处理子程序中，则把一个档案记录看成由若干数据项组成。对整个记录的操作，实际上是在相应命令处理子程序中，通过对该记录的各有关数据项操作来完成的。这种分层次地组织数据和描述算法的思想正是近代结构化程序设计和面向对象程序设计的精髓。

现在，可以把上述几例所揭示的事实归纳如下。

①弄清问题数据间的关系，合理地组织数据，是设计解题算法和程序的基础。

②问题数据间的关系可分为两大类：反映数据间因果变化的运算关系和服务于算法描述的结构关系。

③运算与结构是相互影响、相互依赖的：运算以结构为基础来建立，结构要适于运算的实现。

④分层次地组织数据和描述算法是近代软件设计实践中常用的思想。

1.2 数据的逻辑结构和存储结构

前面说到，在计算机领域内，人们把一切能输入计算机中并被其处理的信息通称为数据。为了有效地设计解题算法，有必要对数据的结构关系进行深入的研究。为此，先来解释两个有关的基本术语：数据元素和数据项。

数据元素是数据的基本单位，它在算法和程序中作为一个整体而被加以考虑和处理。换句话说，数据元素被当作运算的基本单位，通常具有完整的确定的含义。例如，在例1.3主控部分中，每一管理命令都是以档案记录作为基本单位来处理的，故在这里档案记录被看成是数据元素。

在很多情况下，数据元素又是由数据项组成的。如，例1.3第一个档案记录是由“0001”、“陈建新”、“男”、“工程师”、“470”等5个数据项组成的。数据项通常不具有完整确定的含义，或不被当作一个整体对待。例如，上述5个数据项脱离档案记录而独立使用就将失去它们在档案记录中所具有的完整确定的含义。

值得注意的是，在实际问题中，数据元素和数据项常常是相对的。在某层次被看作数据元素的一些对象可能在更高层次上被看作数据项；反之，在某层次被看作数据项的对象，在更低层次上也可能被看作数据元素。

数据结构指的是数据元素之间的结构关系。严格地说，一个数据结构 B 是一个二元组 B=(K,R)，其中 K 是数据元素的集合，R 是 K 上关系的集合。这里，R 集合中的关系指的是数据元素之间的逻辑关系，因而这种结构又称之为数据的逻辑结构。如，例1.2中多项式的系数序列表示就是一种数据的逻辑结构，其中数据元素集 K 是多项式的系数集，关系集 R 此时只包含一个关系——系数间的线性关系（按对应项次数升幂顺序存放）。

数据的逻辑关系是指数据之间的关联方式或称“邻接关系”。在实际应用中常遇到的4类基

本逻辑关系是：空关系、线性关系、树型关系和图型关系。图1.1给出了由这4类基本逻辑关系组成的基本逻辑结构示意图，图中连线表示逻辑关系。

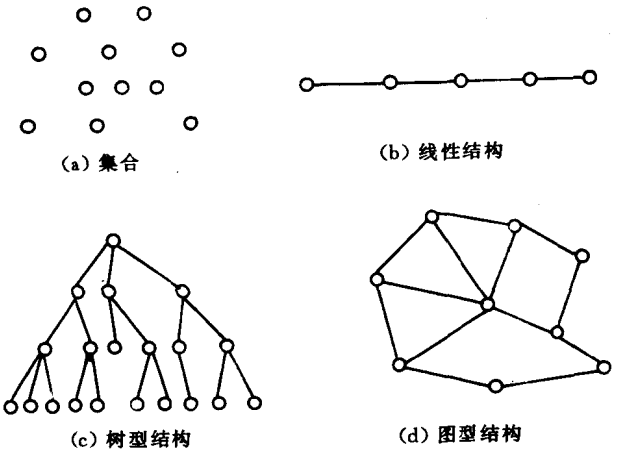


图1.1　四种基本逻辑结构示意图

从图1.1中易见，带空关系的数据结构就是通常的集合，它表明集合中任意两个数据元素之间都没有逻辑关系，是一种最松散的组织形式。线性结构中，数据元素是按线性关系（或说是前后关系）来组织的，犹如日常生活中遇到的一路队形。树型结构具有分支及层次特性，其形态有点像一棵倒挂的树。图型结构最复杂，一般，其中的各个数据元素按逻辑关系互相缠绕，任何两个数据元素都可以邻接。

逻辑结构在计算机中的实现称为数据的存储结构（物理结构），它由数据元素集的存储表示和逻辑关系的存储表示两者组成。例1.2中，多项式系数序列（多项式的逻辑结构）的存储实现为一实型数组 coef[0..maxd]。这时，系数序列中每一系数在机内表示成一实型数组元素，系数间的线性关系则隐式地表示为数组元素的顺序连续存放。

存储结构虽也受数据元素的存储表示的影响，但本质上来说，它更取决于逻辑关系的存储表示。通常，逻辑关系的存储表示有如下4种方式：

(1)顺序存储方式

在这种方式下，每一数据元素对应有一存储表示，常称为结点。所有存储结点物理相继地存放在一个连续的存储区里。也就是说，在顺序存储方式下，元素间的逻辑关系隐含地体现在结点间的位置关系中。按这种方式表示逻辑关系的存储结构称为顺序存储结构。

(2)链式存储方式

在这种方式下，每个数据元素对应的存储结点一般由两部分组成：数据元素存储表示和用于显式表示关系的指针部分。数据元素间的逻辑关系则由对应结点中指针部分的某一指针的具体指向（指针值）来显式地表示。按这种方式表示逻辑关系的存储结构称为链式存储结构。

(3)索引存储方式

在这种方式下，每个数据元素对应一个存储结点，所有存储结点不必连续存放。此外，增设一个索引表，使索引表的第 i 项的值就是第 i 个结点的存储地址。也就是说，在索引存储方式下，数据元素间的逻辑关系体现为索引表中对应项的位置关系，按这种方式表示逻辑关系的存储结构称为索引存储结构。

(4)散列存储方式

在这种方式下，每个数据元素对应一个存储结点，各结点均匀分布在一片存储区里。数据元素的逻辑关系（即一数据元素的逻辑后继）是通过经预先设计好的散列函数的计算得到的（求出其存储地址）。按这种方式表示逻辑关系的存储结构称为散列存储结构。

本书将着重讨论前两种存储方式。

1.3 算法和算法分析

在1.1中，我们已看到一些算法实例。一般来说，一个算法就是一个步骤的有穷序列，其目的是要解决某一问题。从前面实例可见，每一算法必涉及以下三个要素：算法所处理的对象、对象上可进行的操作及操作流程的控制（即操作的执行顺序）。

算法是计算机科学的一个基本概念，也是程序设计的一个核心概念。任何算法总是用某种语言描述的，即用该语言表达算法中的所有处理对象、可进行的操作和操作流程控制。考虑到学习本课程必须与上机练习紧密结合起来，因而抛弃了许多教材中常用的伪语言，而采用了实用的PASCAL语言来描述。事实证明，这既有利于读者上机实践，加深对所学算法的理解，也有利于今后把所学数据结构和算法知识应用于实际。

一般地，对同一个问题可以设计出求解它的多种不同算法。这就产生了如何评价这些算法及在实际应用中如何选择合适算法的问题。

严格地说，算法的选择依赖于许多因素。不过，最重要的有如下三个：执行需求（时间），存储需求（空间）和程序设计需求（难度、可读性和模块性）。由于程序设计需求难于严格分析，故本书评价算法时仅考虑执行需求和存储需求，并分别称它们为算法的时间复杂性和空间复杂性，把确定某一算法的时、空复杂性工作称为算法分析。又由于在实用中，人们对时间复杂性的关心常比对空间复杂性的关心更迫切，故在算法分析中，常把重点放在算法的时间复杂性上。

一个算法的时、空复杂性是指该算法的执行需求（即时间需求）和存储需求（即空间需求）。说得更明确些，前者是算法包含的计算量，后者是算法需要的存储量。

应指出的是，为了评价算法的时间复杂性，并不需要知道计算量的精确值，只要能反映出求解同一问题的不同算法的计算量之间的差异即可。通常采用下述办法来估算某类问题的各个算法在给定输入下的计算量：

①根据该类问题的特点合理地选择一种或几种操作作为“标准操作”；

②确定每个算法在给定输入下共执行了多少次标准操作，并将此次数规定为该算法在给定输入下的计算量。

还应注意的是，一个算法在不同输入下的计算量常是不同的。人们发现，这主要与问题规模（即所包含的数据元素的数目）有关。为便于讨论和比较，今后总把问题规模假定为n。算法在问题规模为n的输入下的计算量将记为T(n)。

前面说到，在评价算法的时、空复杂性时，并不需要知道计算量的精确值，而只关心能否反映不同算法的差异。为此，引进了便于比较这种差异的所谓O( )记法（阶或数量级记法）。

严格地说，一个函数g(n)是O(f(n))的，假如存在正常数c和i，使之对所有n>i，有g(n)<cf(n)。也就是说，对几乎所有n值，函数g(n)以函数f(n)为上界（在相差一个常数因子c的意义下），或说函数g(n)的增长不会比函数f(n)快一个常数因子。有时简单地说，函数g(n)是“阶f(n)”的，下面的例子可帮助进一步弄清这一点。

g(n)	O(f(n))	量级
20	O(1)	常数阶
$\frac{1}{2}n^2+3$	$O(n^2)$	平方阶
$500n^2-17$	$O(n^2)$	平方阶
n^3+n^2-1	$O(n^3)$	立方阶

此外，很多算法的时间复杂性不仅依赖于问题的规模，还依赖于问题数据原有的性态。例如，有的排序算法对某些原始数据(已有序状态)，其时间复杂性为 O(n)，而对另一些数据(如逆序状态下)可达 $O(n^2)$。这就提出了算法在最好情况下、最坏情况下和平均情况下的时间复杂性问题。不过，实用中真正有意义的还是最坏情况下和平均情况下的时间复杂性。

①算法在最坏情况下的时间复杂性是指此算法在所有可能输入下求得的计算量中的最大值。

②算法在平均情况下的复杂性是指此算法在所有可能输入下求得的计算量通过加权求得的平均值。

存储量和空间复杂性的概念与计算量与时间复杂性的概念类似，但通常在实用中，人们更关心一个算法除初始数据占用的存储空间之外所需的附加存储量。

最后，再介绍今后算法分析中要用到的两个记号⌊x⌋和⌈x⌉，其中 x 为任一实数。记号⌊x⌋意指不大于 x 的最大整数，记号⌈x⌉意指不小于 x 的最小整数。

习　题

1—1　试举一例说明问题数据中的运算关系和结构关系。

1—2　简要回答以下问题。

(1)数据项和数据元素的区别和关系。

(2)何谓数据的逻辑结构?4种基本逻辑结构的特点是什么?

(3)何谓数据的存储结构?4种常用存储方式的特点是什么?

(4)何谓算法的时间复杂性和空间复杂性?

1—3　设计求解下列问题的算法，并分析其最坏情况下的时间复杂性及其量级(假定取比较操作为标准操作)。

(1)在数组 A[1..n]中查找值为 k 的元素，若找到则输出其位置 i($1\leqslant i\leqslant n$)，否则输出0。

(2)找出数组 A[1..n]中元素的最大值和次最大值。

1—4　证明以下各断言为真：

(1)17是 O(1)的；

(2)$n(n-1)/2$是 $O(n^2)$的；

(3)$\max(n^3,10n^2)$是 $O(n^3)$的；

(4)假如 p(x)是一 k 次多项式，则 p(n)是 $O(n^k)$的。

1—5　考虑 n 的以下各函数：

$f_1(n)=n^2$，　$f_2(n)=n^2+1000n$

$$f_3(n)=\begin{cases}n,\text{若 } n \text{ 为奇数},\\ n^3,\text{若 } n \text{ 为偶数}。\end{cases}\qquad f_4(n)=\begin{cases}n,\text{若 } n\leqslant 100,\\ n^3,\text{若 } n>100。\end{cases}$$

试指出，对每一不同的 i 与 j，$f_i(n)$是否为 $O(f_j(n))$的。

第二章　线性表

线性表是一种应用十分广泛的数据结构，也是许多其它有用的数据结构的基础。本章介绍线性表的概念、运算及两种常见的实现方法：顺序存储实现和链式存储实现。

2.1　线性表及其基本运算

2.1.1　线性表

简单地说，线性表是一组同类型元素通过线性关系（或称前后关系）组织起来的数据结构，即

线性表＝有限元素集＋（元素间的）线性关系

更严格地说，线性表是一种具有如下特征的数据结构：

①它由 n≥0个同类型元素组成；

②当 n≠0时，存在第一元和最后元；

③除第一元和最后元外，每一元恰有一个直接前驱和一个直接后继；

④当 n≥2时，第一元无前驱，且仅有一个直接后继；最后元无后继，且仅有一个直接前驱。

通常，n 元线性表记为$(a_1,a_2,\cdots,a_n)$，这时，元素间的线性关系就是书写顺序关系。

线性表中元素的个数，称作线性表的长度，简称表长。长度为0的线性表为空表。

现实世界中存在大量可用线性表来描述或组织的数据。

常见的英文字母表(A,B,C,…,Z)是英文字母集按字母顺序关系组成的线性表。

学生成绩登记表可把各学生成绩情况按学号顺序组成线性表。

商店的商品信息可按商品编号组成线性表，也可先按品种，再按编号来组成线性表。

机关的人事记录可先按部门，再在部门内按姓名笔划（或参加工作时间）等来组成线性表。

一家族的成员可先按字辈，在同辈中再按长幼来组成线性表。

2.1.2　线性表的基本运算

利用线性表中给定的线性关系，可在线性表上组织各种各样的运算与操作。线性表的基本运算，也就是常规操作有：

①求表长——求线性表中元素个数；

②遍历——从左到右（或从右到左）扫描（读及处理）表中各元素；

③按编号查找——找出表中第 i 个元素；

④按特征查找——按某特定值查找线性表；

⑤插入——在第 i 个位置上（即原第 i 元素前）插入一新元素；

⑥删除——删除原表中第 i 个元素；

⑦排序——按元素某成分值的递增（或递减）序，重排表中各元素。

利用上述这些基本运算，人们就可以根据需要组织有关线性表的各种更复杂的运算，如：