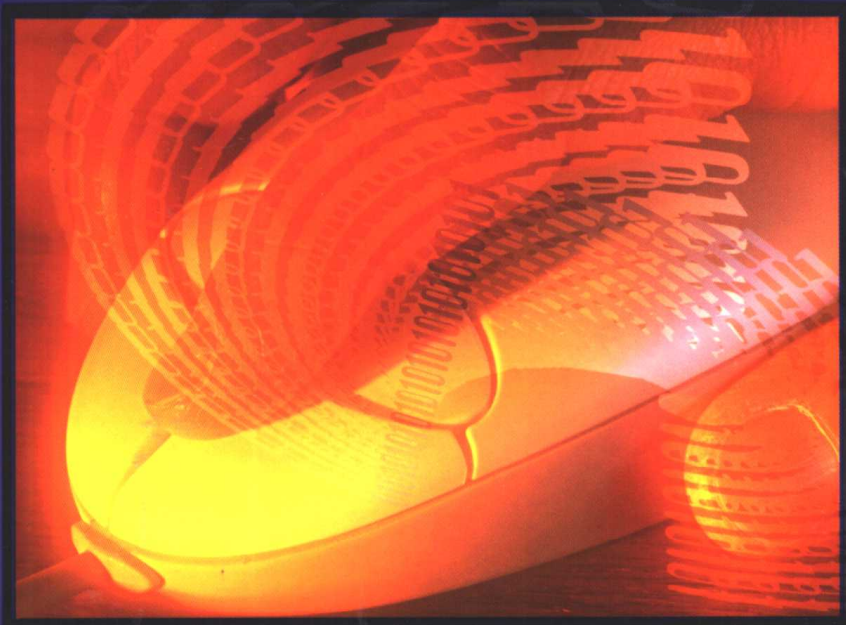


高职高专计算机类规划教材

数据结构

(C 语言版)

姚 菁 主编



机械工业出版社
China Machine Press

中国计算机学会推荐教材

数据结构

第四版

图 像 工 程



清华大学出版社

高职高专计算机类规划教材

数 据 结 构

(C 语言版)

主 编 姚 菁

副主编 刘喜勋

参 编 葛永明

史志强

万雅静

主 审 韩 穗



机械工业出版社

本书介绍了各种类型的数据结构和查找、排序的各种方法。对于每一种类型的数据结构都详细阐述了基本概念、逻辑特性和存储结构。书中采用 C 语言作为算法描述语言,清晰易读,学生只需根据自己的计算机系统的特点,对算法作少量的修改或不作修改即可上机实现。

本书概念清楚,内容丰富,深入浅出。每章都有具体实例,帮助读者理解,具有一定的实用性。各章都附有习题,全书结尾还附有上机实验及相关程序,供读者练习,以巩固、加深对课程内容的理解。

本书为高职高专计算机专业的教材,也可以作为中专计算机专业的教材,亦可供工程技术人员参考。

图书在版编目(CIP)数据

数据结构: C 语言版/姚菁主编. —北京:机械工业出版社, 2000.2
高职高专计算机类规划教材
ISBN 7-111-08637-6

I. 数… II. 姚… III. 数据结构—高等学校; 技术学校—教材
N. TP311.12

中国版本图书馆 CIP 数据核字 (2000) 第 81123 号

机械工业出版社 (北京市百万庄大街 22 号 邮政编码 100037)

责任编辑: 王小东 版式设计: 霍永明 责任校对: 申春香

封面设计: 姚毅 责任印制: 何全君

北京第二外国语学院印刷厂印刷·新华书店北京发行所发行

2001 年 2 月第 1 版·第 1 次印刷

1000mm×1400mmB5·7.125 印张·277 千字

0 001—5000 册

定价: 18.00 元

凡购本书,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

前 言

数据结构是计算机程序设计的重要理论技术基础,是计算机学科的核心课程,也是其他理工专业学生进一步学习计算机知识必备的选修课程。本教材系高职高专计算机类规划教材。

本课程的参考学时数为 72。全书共分九章,第一章综述了数据结构的基本概念,并对算法、算法描述及算法分析作了简单说明;第二章至第六章分别介绍了线性表、栈、队列、数组、串、链表、树和图等基本类型的数据结构及其应用;第七章、第八章介绍了查找和排序的基本算法并简单进行了性能分析;第九章介绍了文件的概念和文件的几种组织。

本书力求内容与实例相结合,语言的描述深入浅出、简洁明了。书中的算法除了用文字作简明的描述外,还用 C 语言作了较为详尽的描述。为了便于学习和理解,书中每章的结束都有综合应用的例子,同时给出了完整的 C 语言程序。本书每章都配有适量习题,全书结尾还附有上机实验及相关程序。

本书既可作为高、中等职业技术学院计算机专业的教材,也可以作为从事计算机应用的工程技术人员的参考书。

本书由姚菁(第一章至第三章)、刘喜勋(第五章、第六章)、葛永明(第四章、第九章)、史志强(第七章)、万雅静(第八章)编写;姚菁统编全稿,由韩穗高级讲师主审。

由于编者水平有限,加上时间匆促,错误和不当之处在所难免,敬请广大读者批评指正。

编 者

2000 年 10 月

目 录

前 言

第一章 绪论	1
第一节 引言	1
第二节 数据结构的基本概念	1
第三节 算法的概念、算法描述和算法分析	3
一、算法的概念	3
二、算法描述	4
三、算法分析	5
习题	6
第二章 线性表和数组	7
第一节 线性表及其存储表示	7
一、线性表	7
二、线性表的存储表示	8
第二节 线性表的查找、插入和删除	9
一、线性表的查找	10
二、线性表的插入和删除	12
第三节 栈和队列	14
一、栈的结构及运算	14
二、计算表达式——栈的应用举例	17
三、队列的结构及运算	19
第四节 数组的顺序分配	24
第五节 稀疏矩阵	27
第六节 线性表的应用举例	30
习题	37
第三章 链表	39
第一节 线性链表的存储表示	39
第二节 线性链表的建立、查找、插入和删除	40
一、线性链表的建立和查找	40
二、线性链表的插入和删除	45
第三节 链栈和链队列	49

一、链栈的结构及运算	49
二、链队列的结构及运算	50
第四节 循环链表	52
一、循环链表的结构	52
二、循环链表的查找、插入和删除	52
第五节 双向链表	62
一、双向链表的结构	62
二、双向链表的查找、插入和删除	63
第六节 链表的应用	68
一、多项式相加问题	68
二、约瑟夫环问题	74
习题	77
第四章 串	78
第一节 串的基本概念	78
第二节 串的基本运算	79
第三节 串的存储结构	81
一、串的静态存储结构	81
二、串的动态存储结构	82
第四节 串基本操作的实现	84
一、求子串定位函数 $\text{index}(s, t)$	84
二、求替换运算 $\text{replace}(a, b, c)$	85
第五节 串操作应用——文本编辑	86
习题	88
第五章 树	89
第一节 一般树的概念	89
一、一般树的定义	89
二、基本术语	89
三、树的存储结构	90
第二节 二叉树	92
一、二叉树的基本概念	92
二、二叉树的性质	94
三、二叉树的存储	95
第三节 二叉树的遍历	97
一、二叉树的遍历方式	97
二、遍历算法	97
三、由数据序列恢复二叉树	99

第四节 线索二叉树	103
一、线索二叉树的结构	103
二、线索二叉树的检索	104
第五节 二叉树的应用——表达式树	106
一、表达式树的概念	106
二、构造表达式树的方法	106
三、构造表达式树的算法	107
第六节 二叉排序树	109
一、二叉排序树的结构	109
二、建立二叉排序树的算法	109
三、在二叉排序树中删除结点	111
四、二叉查找树	113
第七节 哈夫曼树	113
一、树的路径长度和带权路径长度	113
二、哈夫曼树和哈夫曼算法	115
第八节 二叉树与树、森林之间的转换	117
一、二叉树与树之间的转换	117
二、二叉树与森林之间的转换	119
习题	120
第六章 图	122
第一节 基本术语	122
第二节 图的存储结构	124
一、邻接矩阵	124
二、邻接链表	125
三、多重邻接表	127
第三节 遍历图	128
一、深度优先搜索法	128
二、广度优先搜索法	129
第四节 最短路径	131
一、从某个源点到其他各顶点的最短路径	131
二、求每一对顶点之间的最短路径	135
第五节 拓扑排序	138
一、AOV 网	138
二、拓扑排序的方法	139
第六节 关键路径	141
习题	142

第七章 查找	144
第一节 顺序查找	145
第二节 折半查找	146
第三节 分块查找	150
第四节 哈希法	151
一、哈希法查找的概念	151
二、哈希函数的构造	152
三、解决冲突的方法	155
第五节 各种查找方法的比较及应用	160
一、各种查找方法的比较	160
二、应用举例	161
习题	165
第八章 排序	166
第一节 插入排序	167
一、直接插入排序	167
二、希尔排序	169
第二节 选择排序	171
一、直接选择排序	171
二、堆排序	172
第三节 交换排序	177
一、冒泡排序	177
二、快速排序	178
第四节 归并排序	181
第五节 各种排序方法的比较及应用	184
一、各种排序方法的比较	184
二、应用举例	185
习题	189
第九章 文件	191
第一节 文件的基本知识	191
一、文件的概念	191
二、文件的存储介质	191
三、文件的种类	193
四、文件的操作	193
第二节 文件的结构	194
一、顺序文件	194

二、索引文件	195
三、索引顺序文件 ISAM	196
习题	199
实验	200
实验一 线性表的查找、插入、删除	200
实验二 线性链表的查找、插入、删除	200
实验三 循环链表的操作	204
实验四 串操作运算	209
实验五 线索二叉树的检索	213
实验六 二叉排序树的建立及中序遍历	215
实验七 最短路径的实现	216
实验八 查找练习	217
实验九 学生成绩管理	218
参考文献	220

第一章 绪 论

计算机技术的飞速发展已远远地超出了人们对它的估计，它的应用范围也日益扩大。计算机已不仅仅用于科学计算，而是更多地用于进行数据处理、实时控制等的非数值计算处理工作。同时，计算机加工处理的对象，也从简单的数值发展到字符、表格和图像等各种具有一定结构关系的数据，这就给程序设计带来一些新的问题。如何使计算机能高效率、高可靠性的准确处理数据，编写出一个“好”程序，就需要分析待处理的数据的特性以及各处理数据之间存在的关系，这就是“数据结构”的形成和发展的背景。

第一节 引 言

“数据结构”作为一门独立的课程是从 1968 年才开始设立的。在这之前，它的某些内容曾在其他课程，如表处理语言中有所阐述。1968 年在美国一些大学的计算机系的教学计划中，虽然把“数据结构”规定为一门课程，但对课程的范围仍没有作明确规定。当时，数据结构几乎和图论，特别是和表、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、代数结构、集合代数论关系等方面。这就是目前形成的两大课程；数据结构和离散数学。近年来，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的内容。

1968 年美国唐·欧·克努特 (D.K.Knuth) 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 60 年代末到 70 年代初，出现了大型应用程序和系统软件，结构程序设计思想成为程序设计方法的主要内容，人们越来越重视数据结构。从 70 年代中期到 80 年代初，各种版本的数据结构著作相继问世。我国从 1978 年开始，各院校先后开设了“数据结构”课。“数据结构”是我国计算机专业教学计划中的核心课程之一，这一门课的内容不仅是一般程序设计的基础，而且是实现编译程序、操作系统、数据库系统及其他系统程序的基础。

第二节 数据结构的基本概念

数据 (Data) 是人们利用文字符号、数字符号及其他规定的符号对现实世界

的事物及其活动所做的描述。凡是能被计算机输入、存储、处理和输出的一切信息都叫数据。因此，数字、字符、图像、图形、声音等都是属于数据的范畴。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等等。

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。一个数据元素通常的表现形式为由一个或若干个数据项组成，数据项是具有独立含义的最小标识单位。例如，一张名片就是一个数据元素，组成它的数据项可以是姓名、职务、职称、电话、邮编、地址和电子信箱等。数据元素有时也称为结点或记录。

数据对象 (Data Object) 是性质相同的数据元素的集合，是数据的一个子集。例如，整数数据对象是集合 $\{0, +1, +2, +3 \dots\}$ ，字母字符数据对象是集合 $\{'A', 'B', 'C', \dots, 'Z'\}$ 。

数据处理是指对数据进行录入、查找、修改、合并、删除、统计、计算和输出等的操作过程。

为了给出数据结构的概念，下面先讨论两个例子。

例 学生成绩表如图 1-1 所示。

在图 1-1 中，每一行都描述了一个人的有关信息，由学号、姓名、性别等数据项组成，即每一行表示一个数据元素。表中数据元素之间的逻辑关系是顺序的，除了第一个元素和最后一个元素之外，其余元素有且仅有唯一的一个与它相邻且在它前面的元素（称为直接前趋）和唯一的一个与它相邻且在它后面的元素（称为直接后继）。例

学号	姓名	性别	课名	成绩
...	...	...	...	...
97011	李军	男	英语	65
97012	赵红	女	英语	78
97013	王刚	男	英语	90
...	...	...	...	...

图 1-1 学生成绩表

如，表中“赵红”所在的这个数据元素的直接前趋为“李军”这个数据元素，直接后继为“王刚”这个数据元素。对于满足这种顺序关系的表在计算机中如何进行存储表示则是存储结构研究的内容，根据不同的方式可采用顺序存储与非顺序存储。另外，在这张表中可能要经常查阅某一学生的成绩，如有新生加入时要增加数据元素，或有学生退学时要删除相应元素。因此，进行查找、插入和删除就是数据的运算问题。把图 1-1 中数据的逻辑关系、存储结构和运算这三个问题搞清楚，也就弄清了学生成绩表这个数据结构，从而可以有针对性地进行问题的求解。

例 研究城市交通网络问题，如图 1-2 所示。

图中的圆圈表示城市，边表示城市间的交通。在这个问题中，每一个城市就是一个数据元素，它们之间的逻辑关系不再是顺序的，而是一种称为图的结构。这样一种复杂的结构如何在计算机中存储表示也是有关存储结构研究的问题。此

外, 仍然会有数据的运算, 如求城市间的距离, 求某两个城市间的最短路径等。可见, 数据元素间的逻辑结构、存储结构和运算这三个问题也是我们要在求解问题之前首先搞清楚的, 这些问题弄清楚, 问题也就易于解决了。

通过以上例题, 我们发现在实际问题中, 有些问题的数学模型不是数值方程, 而是一些表、图、树等, 因此可以通过下面的描述来理解数据结构的内涵。

现实的世界是信息世界, 所谓信息就是客观存在的反映, 而数据就是信息的表现形式和描述, 这种描述是为了能被计算机所识别、存储和处理。这种描述一般可归结为数字、字符和各种符号的集合。如上例中的学号、姓名、课名、成绩等。值得注意的是

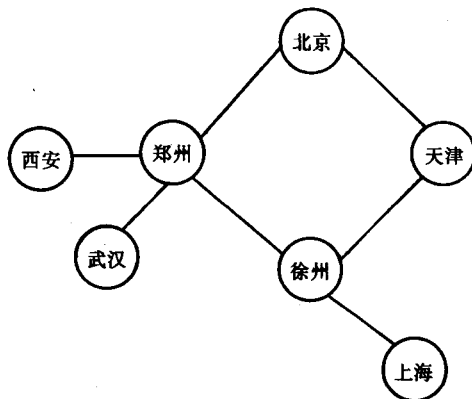


图 1-2 城市交通网例图

的是数据结构中所研究的数据不是孤立的, 而是表现为相互关联的数据。如学生姓名与成绩相对应, 这种数据元素之间存在的相互关系就是数据的结构, 也称为数据之间的逻辑结构。在图 1-1 学生成绩表中, 学生成绩与课名的联系表现为数据之间的线性关系; 在图 1-2 城市交通网例图中, 结点 (圆圈表示) 之间的关系可以是任意的, 图中任意两个数据元素之间都可能相关。把数据和表示数据之间的关系即结构在计算机中的存储形式, 称作数据的物理结构或存储结构。

至此可以认为: 数据结构就是研究数据的逻辑结构和物理结构, 以及它们之间的相互关系和所定义的算法在计算机上运行的学科。本书中将要讨论的逻辑结构有线性关系的表、栈、队列、数组、串和链, 非线性关系的树和图的结构。它们分别以顺序和非顺序的映像关系, 在内存中表示不同的存储结构。本书还将讨论并给出上述结构的算法和算法的基本应用, 同时附有相应的 C 语言程序。

第三节 算法的概念、算法描述和算法分析

学习数据结构是为了更有效地提高程序设计的效率, 使所设计的程序结构合理、层次清晰、内存占有量少、运行时间短。其中, 关键环节之一是数据结构算法的设计和对核算法在运行中的时间分析。有专家曾经概括过这样的公式: 程序 = 算法 + 数据结构。这表明要设计出一个好的程序, 首先要选择好的数据结构, 然后在此数据结构上设计出好的算法。

一、算法的概念

确切地说,算法是对待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计算机的一个或多个操作。算法满足以下性质:

- 1) 输入性:具有零个或若干个输入量。
- 2) 输出性:至少产生一个输出。
- 3) 有穷性:每一条指令的执行次数是有限的。
- 4) 确定性:每一条指令的含义明确,无二义性。
- 5) 可行性:每一条指令都应在有限的时间内完成。

二、算法描述

研究数据结构的目的在于有效地进行程序设计,主要是非数值性程序设计,因此,在讨论各种数据结构的基本运算时,都将给出相应的算法。为了便于理解和掌握算法的设计思想和实质,选择一种合适的算法语言来描述是一个重要问题。本教材中大多数算法除了用文字作简明的描述外,还用 C 语言作了较为详尽的描述。学生只需根据自己的计算机系统的特点,对算法作少量修改或不作修改即可上机实现。

例 计算 $f = 1! + 2! + 3! + \cdots + n!$, 用 C 语言描述。

```
void factorsum(n)
{
    int n;
    int i,j;
    int f,w;
    f=0;
    for (i=1,i<=n;i++)
    {
        w=1;
        for (j=1,j<=i;j++)
            w=w*j;
        f=f+w;
    }
    return(f);
}
```

上述算法所用到的运算有乘法、加法、赋值和比较,其基本运算为乘法操作。在上述算法的执行过程中,对外循环变量 i 的每次取值,内循环变量 j 循环 i 次。因为内循环每执行一次,内循环体语句 $w = w * j$ 只作一次乘法操作,即当内循环变量 j 循环 i 次时,内循环体的语句 $w = w * j$ 作 i 次乘法。所以,整个算法所作的乘法操作总数是: $f(n) = 1 + 2 + 3 + \cdots + n = n(n-1)/2$ 。

三、算法分析

在计算机程序设计中，算法分析是十分重要的，同时也是相当复杂的。本教材就此内容仅作一般的介绍。对于某一算法如何取得算法分析的结果不作进一步深入地推导，而将重点放在对已经获得的算法分析的结果如何在应用程序中使用。

那么，如何衡量和评价一个算法的优劣呢？通常有三个方面的考虑因素：

- 1) 执行算法后，在计算机中运行所消耗的时间，即所需的机器时间。
- 2) 执行算法时，在计算机中所占存储量的大小，即所需的存储空间。
- 3) 所设计的算法是否易读、易懂，是否容易转换成其他可运行的程序语言。

显然我们希望所设计的算法占用空间小，运行时间短，其他性能也好，但实际上是很困难的。计算机的时间和空间这两大资源往往相互抵触，缩短算法的运行时间往往以牺牲较多的空间为代价，而为了节省空间又可能以耗费更多时间为条件。因此，到底如何选取应根据具体算法的应用而定。若程序只使用一次或若干次，则力求算法简明易懂，容易转换成执行的语言，同时便于上机调试；若该程序需要反复多次运行或多次调用，则应选用执行时间尽可能短的算法；若待处理的数据量甚大，而所使用的计算机存储空间相对较小，则选择的算法应主要考虑如何节省空间。本教材主要考虑时间的特性。

一个算法的执行时间等于算法所有语句执行时间的总和，而任一语句的执行是该语句的总执行次数与执行一次所需时间的乘积。由于精确的计算出该算法的总执行时间是相当复杂的，而且是困难的，所以，只能根据给定的计算模式，粗略地估算该算法的总执行时间。

设有程序段表示两个 $n \times n$ 矩阵相乘，其算法描述如下：

语句	执行次数
for (i = 1; i <= n; i++)	$n + 1$
for (j = 1; j <= n; j++)	$n(n + 1)$
{	
c[i][j] = 0;	n^2
for (k = 1; k <= n; k++)	$n^2(n + 1)$
c[i][j] = c[i][j] + a[i][k] * b[k][j];	n^3
}	

以上算法总执行时间为

$$\begin{aligned} T(n) &= n + 1 + n(n + 1) + n^2 + n^2(n + 1) + n^3 \\ &= 2n^3 + 3n^2 + 2n + 1 \end{aligned}$$

显然它是矩阵的阶 n 的函数，并且当 $n \rightarrow \infty$ 时， $T(n)/n^3 \rightarrow 2$ ，这表示当 n 充分大时， $T(n)$ 和 n^3 是同阶的。引入“O”记录（读作大“O”）、可记 $T(n)$

$= O(n^3)$, 称为算法的时间复杂度。

常见的时间复杂度有常数阶 $O(1)$ 、对数阶 $O(\log_2 n)$ 、线性阶 $O(n)$ 、平方阶 $O(n^2)$ 、立方阶 $O(n^3)$ 、指数阶 $O(2^n)$ 等。

本教材在一些主要的算法中都直接给出了该算法的时间复杂度, 以便读者在选择算法时就算法的时间因素在应用程序中的位置给出参考的数据, 便于算法的使用。

习 题

1. 简述下列术语: 数据、数据元素、数据对象、数据结构。
2. 什么是逻辑结构? 什么是存储结构?
3. 举一个数据结构的例子, 叙述其逻辑结构和存储结构两方面的内容。
4. 什么叫算法? 评价算法优劣的主要指标是什么?
5. 设 n 为正整数, 求出下列算法基本操作的执行时间 $T(n)$ 。

(1) $i = 1; k = 0;$

```
while ( $i < = n - 1$ )
```

```
{
```

```
     $k = k + 10 * i;$ 
```

```
     $i = i + 1;$ 
```

```
}
```

(2) $x = 1;$

```
for ( $i = 1; i < = n; i++$ )
```

```
    for ( $j = 1; j < = i; j++$ )
```

```
         $x = x + 1;$ 
```

(3) $x = 91; n = 100;$

```
while ( $n > 0$ )
```

```
{
```

```
    if ( $x > 100$ )
```

```
    {
```

```
         $x = x - 10;$ 
```

```
         $y = y - 1;$ 
```

```
    }
```

```
    else
```

```
         $x = x + 1;$ 
```

```
}
```


第二章 线性表和数组

线性表是一种最基本、最简单且应用十分广泛的数据结构。实际应用中经常要用到一些表，如工资表、学生成绩表等等，这些表中的数据元素之间往往存在着某种结构关系，怎样有效地实现对这些表的操作呢？本章将叙述线性表的逻辑结构和存储结构，将讨论线性表的基本操作，记录元素的查找、插入和删除，最后将讨论线性表的基本应用。

第一节 线性表及其存储表示

一、线性表

线性表 (Linear List) 是一种最简单、最常用的数据结构。直观地讲，它是性质相同的一组数据元素的序列，至于数据元素的具体含义，在不同的具体情况下可以不同。它可以是一个数，一个符号，一串文字，甚至是其他更复杂的信息。例如英文字母表：

(A, B, C, ..., Z)

是一个线性表，表中的数据元素是单个字母。又如，我国有 30 个省、市、自治区：

(北京, 广州, 上海, ..., 台湾)

是一个线性表，表中的数据元素是省、市、自治区的名称。较复杂的线性表中，一个数据元素可以由若干个数据项组成。我们把含有多个项组成的数据元素称作记录 (Record)。例如，前一章中讨论过的学生成绩表 (图 1-1) 也是一个线性表，表中每一个学生的成绩就是一个记录，每一个记录包含五个数据项：学号、姓名、性别、课名、成绩。

综合上述的例子，可以这样来描述线性表：

一个线性表是 $n \geq 0$ 个数据元素 $a_0, a_1, a_2, \dots, a_{n-1}$ 的有限序列。如果 $n > 0$ ，则除 a_0 和 a_{n-1} 外，有且仅有一个直接前趋和一个直接后继数据元素， a_i ($0 \leq i \leq n-1$) 为线性表的第 i 个数据元素，它在数据元素 a_{i-1} 之后，在 a_{i+1} 之前。 a_0 为线性表的第一个数据元素，而 a_{n-1} 是线性表的最后一个数据元素；若 $n=0$ ，则为一个空表，表示无数据元素。这就是说，线性表或者是一个空表 ($n=0$)，或者可以写成：

($a_0, a_1, a_2, \dots, a_{i-1}, a_i, a_{i+1}, \dots, a_{n-1}$)