

# 数据结构

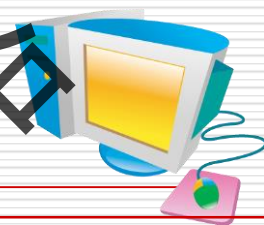
---

## 第四章 串、数组和广义表

人工智能学院  
刘运



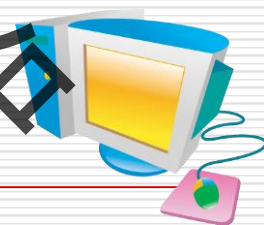
## 4.4 数组



本节所讨论的数组与高级语言中的数组区别：

- 高级语言中的数组是顺序结构；
- 而本章的数组既可以是顺序的，也可以是链式结构，用户可根据需要选择。

## 4.4.1 数组的定义



数组是由一组个数固定，类型相同的数据元素组成的阵列。

**一维数组的特点：**1个下标， $a_i$ 的直接前驱是 $a_{i-1}$ ，直接后继是 $a_{i+1}$ 。

**二维数组的特点：**2个下标，每个元素 $a_{ij}$ 受到两个关系（行关系和列关系）的约束。

$$A_{m \times n} = \begin{pmatrix} a_{00} & a_{01} & \dots & a_{0,n-1} \\ a_{10} & a_{11} & \dots & a_{1,n-1} \\ \dots & \dots & \dots & \dots \\ a_{m-1,0} & a_{m-1,1} & \dots & a_{m-1,n-1} \end{pmatrix}$$

**在行关系中**

$a_{ij}$ 直接前趋是  $a_{ij-1}$

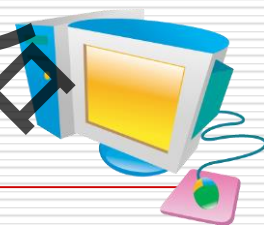
$a_{ij}$ 直接后继是  $a_{ij+1}$

**在列关系中**

$a_{ij}$ 直接前趋是  $a_{i-1,j}$

$a_{ij}$ 直接后继是  $a_{i+1,j}$

# N维数组的数据类型定义



**ADT ARRAY{**

**数据对象:**  $j_i=0, \dots, b_i-1, i=1, 2, \dots, n$

$D = \{a_{j_1, j_2, \dots, j_n} \mid n(>0) \text{ 称为数组的维数, } b_i \text{ 是数组第 } i \text{ 维的长度, } j_i \text{ 为数组元素的第 } i \text{ 维下标, } a_{j_1, j_2, \dots, j_n} \in \text{Elemset}\}$

**数据元素个数:**  $(b_1 * b_2 * b_3 * \dots * b_n)$

**数据关系:**  $R = \{R_1, R_2, \dots, R_n\}$

**基本操作:** 构造数组、销毁数组、读数组元素、写数组元素

**数据关系：**  $R = \{ R1, R2, \dots, Rn \}$

$$R_i = \{ \langle a_{j_1, j_2, \dots, j_i, \dots, j_n}, a_{j_1, j_2, \dots, j_{i+1}, \dots, j_n} \rangle \mid \\ 0 \leq j_k \leq b_k - 1, 1 \leq k \leq n \text{ 且 } k \neq i, 0 \leq j_i \leq b_i - 2, \\ a_{j_1, j_2, \dots, j_i, \dots, j_n}, a_{j_1, j_2, \dots, j_{i+1}, \dots, j_n} \in D \}$$

每个数据元素  $a_{j_1, j_2, \dots, j_n}$  受到  $n$  个关系的约束： **$R1, R2, \dots, Rn$**

数据元素  $a_{j_1, j_2, \dots, j_n}$  在  $R1$  关系中的直接后继元素： **$a_{(j_1+1), j_2, \dots, j_n}$**

数据元素  $a_{j_1, j_2, \dots, j_n}$  在  $R2$  关系中的直接后继元素： **$a_{j_1, (j_2+1), \dots, j_n}$**

.....

数据元素  $a_{j_1, j_2, \dots, j_n}$  在  $Rn$  关系中的直接后继元素： **$a_{j_1, j_2, \dots, (j_n+1)}$**

例子：数组 `int a[3][7][5]`

---

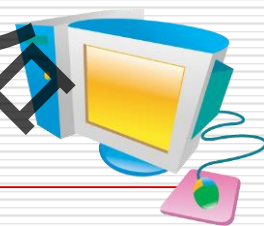
数据元素个数： **105**

每个数据元素 `a[j1][j2][j3]` 受到3个关系的约束：**R1,R2,R3**

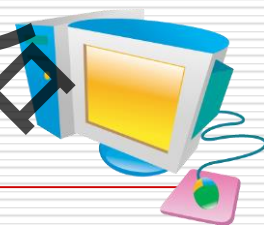
数据元素 `a[2][3][0]` 在R1关系中的直接后继元素：无

数据元素 `a[2][3][0]` 在R2关系中的直接后继元素：**`a[2][4][0]`**

数据元素 `a[2][3][0]` 在R3关系中的直接后继元素：**`a[2][3][1]`**



# 数组的基本操作



1 初始化操作 `InitArray(&A, n, bound1, ..., boundn)`

功能：参数合法，构造数组A，并返回OK；

2 销毁操作 `DestroyArray(&A)`

功能：销毁数组A；

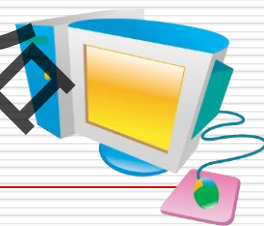
3 读元素操作 `Value(A, &e, index1, ..., indexn)`

功能：若指定下标不越界，读指定下标的元素，用e返回并返回OK；

4 写元素操作 `Assign(A, e, index1, ..., indexn)`

功能：若指定下标不越界，将e赋值给A指定的下标元素并返回OK。

# 二维数组



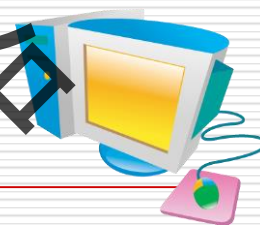
可看作是是一个定长线性表，且它的每个数据元素也是一个定长线性表。

例：

$$A_{m \times n} = \begin{pmatrix} a_{00} & a_{01} & a_{02} \dots & a_{0,n-1} \\ a_{10} & a_{11} & a_{12} \dots & a_{1,n-1} \\ \dots & \dots & \dots & \dots \\ a_{m-1,0} & a_{m-1,1} & a_{m-1,2} \dots & a_{m-1,n-1} \end{pmatrix}$$

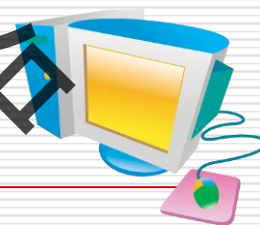


# 二维数组



$$(1) A=(\alpha_0, \alpha_1, \dots, \alpha_{m-1}) \quad \alpha_i=(a_{i0}, a_{i1}, \dots, a_{in-1}) \quad 0 \leq i \leq m-1$$

$$A_{m \times n} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \dots \\ \alpha_{m-1} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \dots & a_{0,n-1} \\ a_{10} & a_{11} & a_{12} \dots & a_{1,n-1} \\ \dots & \dots & \dots & \dots \\ a_{m-1,0} & a_{m-1,1} & a_{m-1,2} \dots & a_{m-1,n-1} \end{bmatrix}$$



(2)  $A=(\alpha_0,\alpha_1,\dots,\alpha_{n-1})$   $\alpha_j=(a_{0j},a_{1j},\dots,a_{m-1,j})$   $0\leq j\leq n-1$

$$A_{m\times n}=\begin{bmatrix} \alpha_0 & \alpha_1 & \dots & \alpha_{n-1} \end{bmatrix}=$$

$$\begin{bmatrix} \begin{pmatrix} a_{00} \\ a_{10} \\ \dots \\ a_{m-1,0} \end{pmatrix} & \begin{pmatrix} a_{01} \\ a_{11} \\ \dots \\ a_{m-1,1} \end{pmatrix} & \begin{pmatrix} a_{02} \\ a_{12} \\ \dots \\ a_{m-1,2} \end{pmatrix} & \dots & \begin{pmatrix} a_{0,n-1} \\ a_{1,n-1} \\ \dots \\ a_{m-1,n-1} \end{pmatrix} \end{bmatrix}$$

## 二维数组的C语言表示:

---

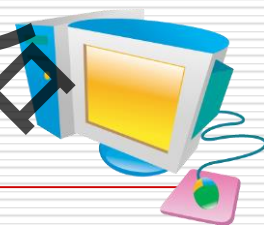
**Typedef elemtype Array1[n];**

**Typedef Array1 Array2[m];**



**Typedef elemtype Array[m][n];**

## 4.4.2 数组的顺序存储



用一组连续的存储单元存放数组的数据元素。

二维数组的两种顺序存储方式：

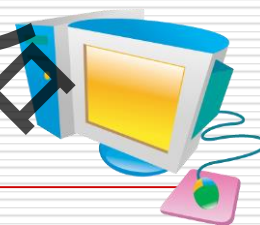
(1) 行序为主序

(2) 列序为主序

例：

$$A_{m \times n} = \begin{pmatrix} a_{00} & a_{01} & a_{02} \dots & a_{0,n-1} \\ a_{10} & a_{11} & a_{12} \dots & a_{1,n-1} \\ \dots & \dots & \dots & \dots \\ a_{m-1,0} & a_{m-1,1} & a_{m-1,2} \dots & a_{m-1,n-1} \end{pmatrix}$$

行序为主序:



$a_{00}$	$a_{01}$	...	$a_{0n-1}$	$a_{10}$	$a_{11}$	...	$a_{1n-1}$	...	$a_{m-1,0}$	$a_{m-1,1}$	...	$a_{m-1,n-1}$
----------	----------	-----	------------	----------	----------	-----	------------	-----	-------------	-------------	-----	---------------

$A_{m \times n} =$

$a_{00}$	$a_{01}$	$a_{02}$	...	$a_{0,n-1}$
$a_{10}$	$a_{11}$	$a_{12}$	...	$a_{1,n-1}$
...	...	...	...	...
$a_{m-1,0}$	$a_{m-1,1}$	$a_{m-1,2}$	...	$a_{m-1,n-1}$

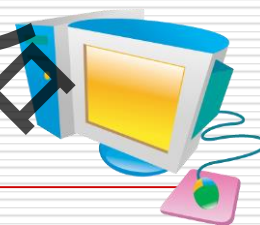
数据元素的存储位置:

$$\text{loc}(i,j) = \text{loc}(0,0) + a_{ij} \text{ 之前的元素个数} \times L$$

↓

$(i*n+j)$

列序为主序:



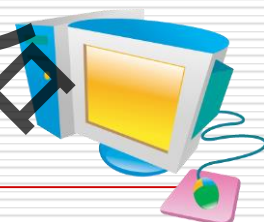
$a_{00}$	$a_{10}$	$\dots$	$a_{m-1,0}$	$a_{01}$	$a_{11}$	$\dots$	$a_{m-1,1}$	$\dots$	$a_{0,n-1}$	$a_{1,n-1}$	$\dots$	$a_{m-1,n-1}$
----------	----------	---------	-------------	----------	----------	---------	-------------	---------	-------------	-------------	---------	---------------

$$A_{m \times n} = \begin{pmatrix} a_{00} & a_{01} & a_{02} \dots & a_{0,n-1} \\ a_{10} & a_{11} & a_{12} \dots & a_{1,n-1} \\ \dots & \dots & \dots & \dots \\ a_{m-1,0} & a_{m-1,1} & a_{m-1,2} \dots & a_{m-1,n-1} \end{pmatrix}$$

数据元素的存储位置:

$$\text{loc}(i,j) = \text{loc}(0,0) + \text{a}_{ij} \text{ 之前的元素个数} \times L$$

$$j * m + i$$




**例1【软考题】：**一个二维数组A，行下标的范围是1到6，列下标的范围是0到7，每个数组元素用相邻的6个字节存储，存储器按字节编址。这个数组的体积是 288 个字节。

**答：**  $\text{Volume} = m * n * L = (6 - 1 + 1) * (7 - 0 + 1) * 6 = 48 * 6 = 288$

**例2：**已知二维数组 $A_{m,m}$ 按行存储的元素地址公式是：  
 $\text{Loc}(a_{ij}) = \text{Loc}(a_{11}) + [(i-1)*m + (j-1)]*K$ ，请问按列存储的公式相同吗？

**答：**尽管是方阵，但公式仍不同。应为：

$$\text{Loc}(a_{ij}) = \text{Loc}(a_{11}) + [(j-1)*m + (i-1)]*K$$



**例3：** 设数组 $a[1...60, 1...70]$ 的基地址为2048，每个元素占2个存储单元，若以列序为主序顺序存储，则元素 $a[32,58]$ 的存储地址为 **8950**。

解：根据列优先公式  $Loc(a_{ij}) = Loc(a_{11}) + [(j-1)*m + (i-1)]*K$

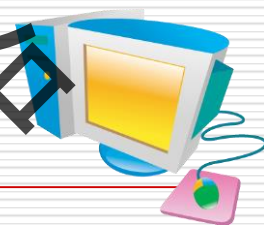
得：  $LOC(a_{32,58}) = 2048 + [(58-1)*60 + (32-1)]*2 = 8950$

想一想：若数组是 $a[0...59, 0...69]$ ，  
结果是否仍为8950？

维界虽未变，但此时的 $a[32,58]$ 不再是原来的 $a[32,58]$



n维数组的数据元素存储位置的计算公式:



$$\text{Loc}(j_1, j_2, \dots, j_n) = \text{Loc}(0, 0, \dots, 0) + \text{Loc}(j_1, j_2, \dots, j_n) \text{ 之前的元素个数} \times L$$

$$(\mathbf{b_2 \times \dots \times b_n \times j_1} + \mathbf{b_3 \times \dots \times b_n \times j_2} + \dots + \mathbf{b_n \times j_{n-1} + j_n})$$

例子: `int a[3][5][7]`

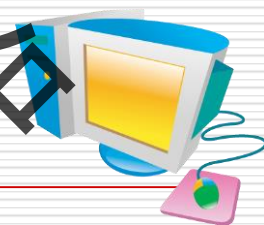
`a[2][3][0]`的存储位置为:

$$\text{Loc}(0, 0, 0) + (\mathbf{5 \times 7 \times 2} + \mathbf{7 \times 3} + \mathbf{0}) \times L$$

$$= \text{Loc}(0, 0, 0) + (\mathbf{70} + \mathbf{21}) \times L$$

$$= \text{Loc}(0, 0, 0) + 91 \times L$$

## 4.4.3 特殊矩阵的压缩存储



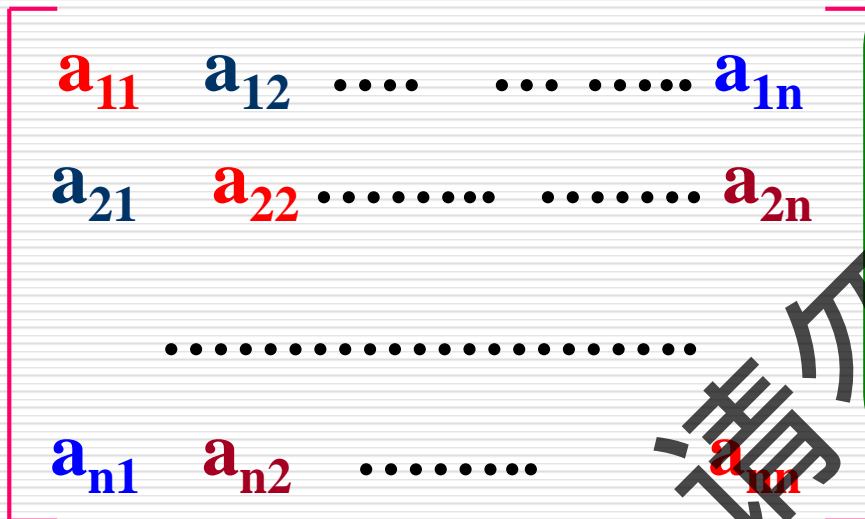
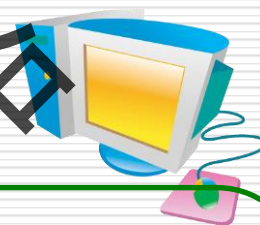
值相同元素或者零元素分布有一定规律的矩阵称  
**特殊矩阵**。

例 对称矩阵、上（下）三角矩阵都是特殊  
矩阵

**对称矩阵**是满足下面条件的 $n$  阶矩阵

$$a_{ij} = a_{ji} \quad 1 \leq i, j \leq n$$

# 1. 对称矩阵



若  $i \geq j$ , 则  $a_{ij}$  在下三角形中。  
 $a_{ij}$  之前的  $i-1$  行 (从第1行到第  $i-1$  行)  
 一共有  $1+2+\dots+i-1 = i(i-1)/2$  个元素, 在第  $i$  行上,  $a_{ij}$  之前恰有  $j-1$  个元素 (即  $a_{i1}, a_{i2}, \dots, a_{i,j-1}$ ), 因此有:  $k = i*(i-1)/2 + j - 1$

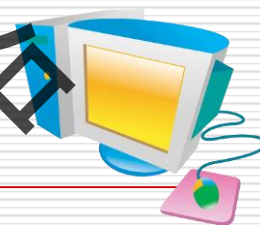
按行序为主序:

a11	a21	a22	a31	a32	.....	an1	.....	ann
k=0	1	2	3	4		$n(n-1)/2$		$n(n+1)/2-1$

$$k = \begin{cases} i(i-1)/2 + j - 1 & \text{当 } i \geq j \\ j(j-1)/2 + i - 1 & \text{当 } i < j \end{cases}$$

若  $i < j$ , 则  $a_{ij}$  是在上三角矩阵中。  
 因为  $a_{ij} = a_{ji}$ , 所以只要交换上述对应关系式中的  $i$  和  $j$  即可得到:  
 $k = j*(j-1)/2 + i - 1$

## 2. 三角矩阵



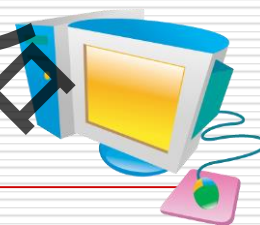
$a_{11}$	0	0	.....	0
$a_{21}$	$a_{22}$	0	.....	0
.....	.....	.....	.....	0
$a_{n1}$	$a_{n2}$	$a_{n3}$	.....	$a_{nn}$

按行序为主序：

$a_{11}$	$a_{21}$	$a_{22}$	$a_{31}$	$a_{32}$	.....	$a_{n1}$	.....	$a_{nn}$
k=0	1	2	3	4		$n(n-1)/2$		$n(n+1)/2-1$

$$\text{Loc}(a_{ij}) = \text{Loc}(a_{11}) + \left[ \frac{i(i-1)}{2} + (j-1) \right] * L$$

### 3. 对角矩阵



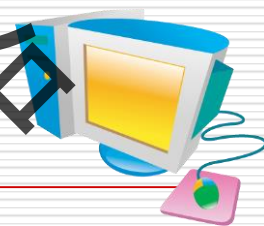
$$\begin{array}{ccccccc}
 a_{11} & a_{12} & 0 & \dots & \dots & \dots & 0 \\
 a_{21} & a_{22} & a_{23} & 0 & \dots & \dots & 0 \\
 0 & a_{32} & a_{33} & a_{34} & 0 & \dots & 0 \\
 \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
 0 & 0 & \dots & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} & \\
 0 & 0 & \dots & \dots & a_{n,n-1} & a_{nn} & 
 \end{array}$$

按行序为主序

$a_{11}$	$a_{12}$	$a_{21}$	$a_{22}$	$a_{23}$	.....		...	$a_{nn}$
$k=0$	1	2	3	4				$3(n-1)$

$$\text{Loc}(a_{ij}) = \text{Loc}(a_{11}) + (2*i + j - 3)*L$$

# 考研真题



## 【2016年计算机联考真题】

有一个100阶的三对角矩阵M，其元素 $m_{ij}$  ( $1 \leq i \leq 100, 1 \leq j \leq 100$ )按行优先依次压缩存储下标从0开始的一维数组N中。元素 $m_{30,30}$ 在N中的下标是 ( B )。

A. 86

B. 87

C. 88

D. 89

## 4.5 广义表

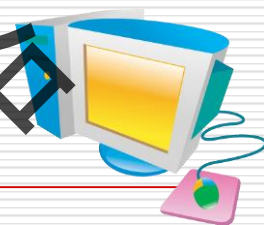


### 4.5.1 广义表的定义

广义表也称为列表，是线性表的一种扩展，也是数据元素的有限序列。记作： $LS = (a_1, a_2, \dots, a_n)$ 。其中  $a_i$  可以是单个元素，也可以是广义表。

#### 说明

- 1) 广义表的定义是一个递归定义，因为在描述广义表时又用到了广义表；
- 2) 在线性表中数据元素是单个元素，而在广义表中，元素可是以单个元素称为原子，也可以是广义表，称为广义表的子表；
- 3)  $n$  是广义表长度；



4) 下面是一些广义表的例子;

$A = ()$  空表, 表长为**0**;

$B = (e)$   $B$ 中只有一个元素 $e$ , 表长为**1**;

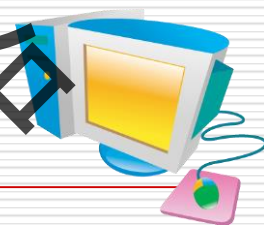
$C = (a, (b, c, d))$   $C$ 的表长为**2**, 两个元素分别为  $a$  和子表  $(b, c, d)$ ;

$D = (A, B, C)$   $D$  的表长为**3**, 它的三个元素  $A$ ,  $B$ ,  $C$  广义表;

$E = (a, E)$  这是一个递归的表, 它的长度为**2**。  $E$ 相当于一个无限的列表:  $E = (a, (a, (a, \dots)))$

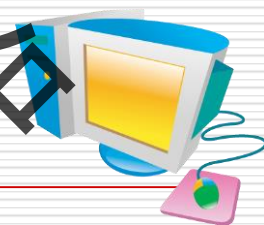


# 广义表的基本操作



- 1) 创建空的广义表L;
- 2) 销毁广义表L;
- 3) 已有广义表L, 由L复制得到广义表T;
- 4) 求广义表L的长度;
- 5) 求广义表L的深度;
- 6) 判广义表L是否为空;
- 7) 取广义表L的表头;
- 8) 取广义表L的表尾;
- 9) 在L中插入元素作为L的第一个元素;
- 10) 删除广义表L的第一个元素, 并e用返回其值;
- 11) 遍历广义表L。

# 广义表的基本运算



(1) 取表头GetHead(LS): 非空广义表的第一个元素, 可以是一个单元素, 也可以是一个子表。

(2) 取表尾GetTail(LS): 非空广义表除去表头元素以外其它元素所构成的表。表尾一定是一个表。

$B = (e)$

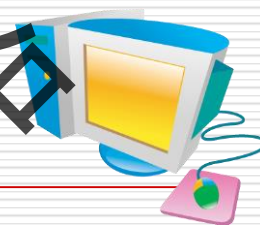
$\text{GetHead}(B) = e$      $\text{GetTail}(B) = ()$

$C = (a, (b, c, d))$

$\text{GetHead}(C) = a$      $\text{GetTail}(C) = ((b, c, d))$

$D = (A, B, C)$

$\text{GetHead}(D) = A$      $\text{GetTail}(D) = (B, C)$



若广义表不空，则可分成表头和表尾，反之，一对表头和表尾可唯一确定广义表。

$A = (a, b, (c, d), (e, (f, g)))$

$\text{GetHead}(\text{GetTail}(\text{GetHead}(\text{GetTail}(\text{GetTail}(A))))))$