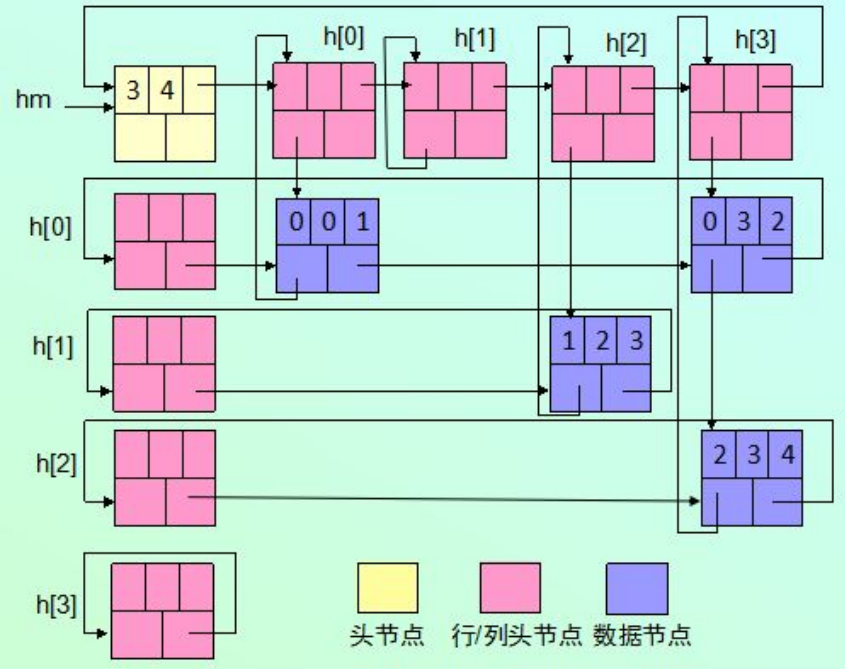

本节主题：

数组的基本概念与存储结构

# 本章内容

- 数组的基本概念与存储结构
- 特殊矩阵的压缩存储
- 稀疏矩阵的三元组表示
- 稀疏矩阵的十字链表表示
- 广义表
- 广义表的存储结构及基本运算的实现

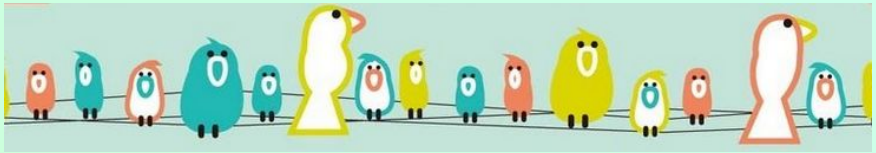


# 不同语境中的数组

- 数据结构视角下的数组
  - 数组A是n（n＞1）个相同类型数据元素$a_1$、$a_2$、…、$a_n$ 构成的有限序列
  - 数组的逻辑表示： A=（$a_1$，$a_2$，…，$a_n$）
  - 其中，$a_i$（1≤i≤n）表示数组A的第i个元素
- C/C++ 语言中的数组

```
#define N 100

int a[N]
```



# 维数不同的数组

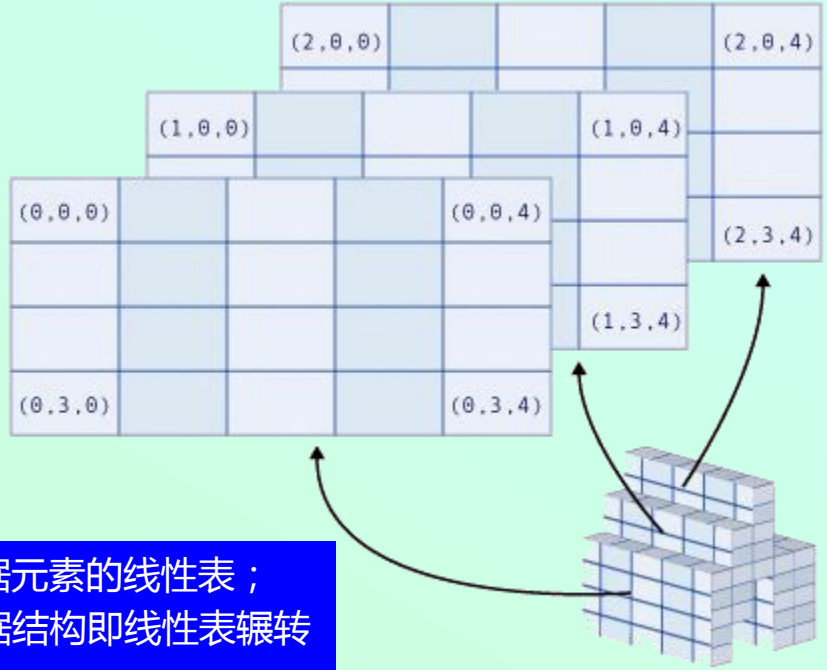
| (0) | (1) | (2) | (3) | (4) |
|---|---|---|---|---|

一维数组，由多个数据元素构成

| (0,0) | | | | (0,4) |
|---|---|---|---|---|
| (1,0) | | | | |
| | | | | |
| (3,0) | | | | (3,4) |

二维数组：每个数据元素都是相同类型的一维数组的一维数组。

- 多维数组都可以看作一个线性表，线性表中的每个数据元素也是一个线性表。



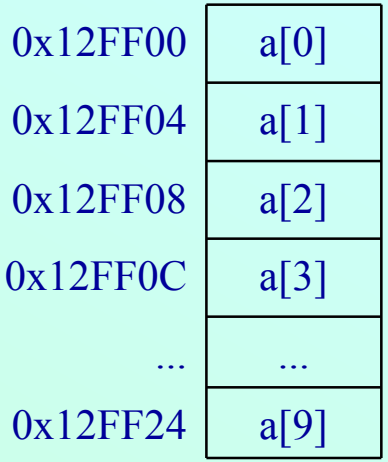

- d(d≥3)维数组，看作一个由d-1维数组作为数据元素的线性表；
- 数组是一种较复杂的线性表结构，由简单的数据结构即线性表辗转合成而得。

# 数组的基本运算

- Value(A,$index_1$,$index_2$,...,$index_d$)——返回各下标指定的A中的对应元素的值
  - A是已存在的 d 维数组，index1，index2，...，indexd是指定的d个下标值，且这些下标均未越界。
  - 例：对二维数组： Value(A, i, j)
- Assign(A,e,$index_1$,$index_2$,...,$index_d$)——将e赋值给由各下标指定的A中元素
  - A是已存在的d维数组，e为元素变量，index1，index2，...，indexd 是指定的d个下标值，且这些下标均未越界。
  - 例：对二维数组：Assign(A, e, i, j)
- ADisp(A,$b_1$,$b_2$,...,$b_d$)：输出d维数组A的所有元素值
- ……

# 数组的存储结构

- 要求
  - 数组的所有元素存储在一块地址连续的内存单元中。
- 实现
  - 几乎所有的计算机语言都支持数组类型
- 数组数据类型的性质（以C/C++语言为例）




（1）数组中的数据元素数目固定，一旦定义,其数据元素数目不再有增减变化。

（2）数组中的数据元素具有相同的数据类型。

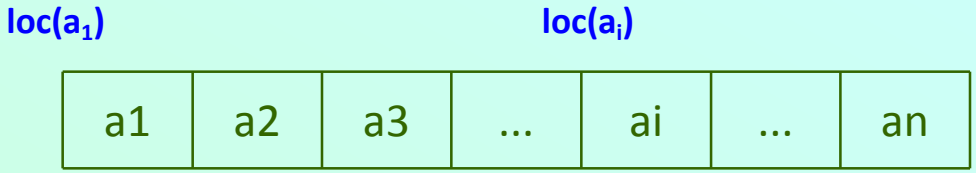
（3）数组中的每个数据元素都和一组唯一的下标值对应。

# 一维数组的存储地址

- 条件
  - 一维数组
  - $a_1$ 的存储地址：$LOC(a_1)$
  - 每个数据元素占用的存储单元数：k
- 任一数据元素$a_i$的存储地址$LOC(a_i)$为

$$LOC(a_i)=LOC(a_1)+(i-1)*k \quad (0\le i\le n)$$

- 特点
  - 一维数组中任一数据元素的存储地址可直接计算得到，即一维数组中任一数据元素可直接存取
  - 一维数组是一种随机存储结构。




怎么看？

- 顺序存储结构用数组实现

  ——从占用存储空间角度
- 数组是随机存储结构

  ——从存取数据角度

# 二维数组的表示和存储

$$A_{m\times n}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & & a_{ij} & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

A是一维数组：

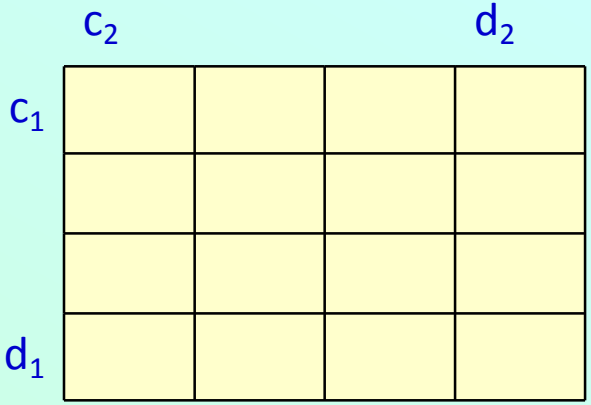
$A=(a_1,a_2,\ldots,a_i\ldots,a_m)$

其中，

$a_i=(a_{i1},a_{i2},\ldots,a_{in})$ $(1\le i\le m)$

- 以行序为主序的存储方式：即先存储第1行，然后紧接着存储第2行，最后存储第m行
  - 行序为主序的线性排列次序：$a_{11},a_{12},\ldots,a_{1n},a_{21},a_{22},\ldots,a_{2n},\ldots,a_{m1},a_{m2},\ldots,a_{mn}$
  - 任一数据元素$a_{ij}$的存储地址：$LOC(a_{ij})=LOC(a_{11})+[(i-1)*n+(j-1)]*k$
- 以列序为主序的存储方式：即先存储第1列，然后紧接着存储第2列，最后存储第m列
  - 列序为主序的线性排列次序：$a_{11},a_{21},\ldots,a_{m1},a_{12},a_{22},\ldots,a_{m2},\ldots,a_{1n},a_{2n},\ldots,a_{mn}$
  - 任一数据元素$a_{ij}$的存储地址：$LOC(a_{ij})=LOC(a_{11})+[(j-1)*m+(i-1)]*k$

# 对数组A[$c_1$..$d_1$, $c_2$..$d_2$]




- 特点
  - 行下标：$c_1$至$d_1$
  - 列下标：$c_2$至$d_2$
- 数据元素$a_{ij}$的存储地址
  - 以行序为主序：$LOC(a_{ij})=LOC(a_{c1c2})+[(i-c_1)*(d_2-c_2+1)+(j-c_2)]*k$
  - 以列序为主序：$LOC(a_{ij})=LOC(a_{c1c2})+[(j-c_2)*(d_1-c_1+1)+(i-c_1)]*k$
- 例：C/C++中数组float a[5][4]，起始地址2000，数据元素长度4，a[3][2]地址？
  - $c_1=0$, $d_1=4$, $c_2=0$, $d_2=3$, $LOC(a_{c1}a_{c2})=2000$, $k=4$, $i=3$, $j=2$
  - 以行序为主序：$LOC(a_{32}) = 2000+[3*4+2]*4 = 2056$
- 推论：C/C++中数组a[m][n]中元素a[i][j]的地址——$LOC(a_{00})+(i*n+j)*k$