Loops, Multi-dimensional Array

王慧妍

why@nju.edu.cn

南京大学 软件学院 计算机软件研究所

回顾

- Loop
  - For
  - While
  - Do-while

标量和聚合变量

- 标量
  - 保存单一数据项
  - int, float, double, char, bool(int)

- 聚合变量
  - 存储成组的数据：数组，结构

[]: *subscript* operator（下标运算符）

Reverse

- 读入一串数，并按照反向顺序输出这些数

```
Enter 10 numbers: 34 82 49 102 7 94 23 11 50 31
In reverse order: 31 50 11 23 94 7 102 49 82 34
```

数组和循环的使用注意点

- 数组下标从0开始！

```c
#include <stdio.h>
#define N 10
int main(){
    int a[N], i;

    for(i = 1; i<= N; i++)
        a[i] = 0;
    return 0;
}
```

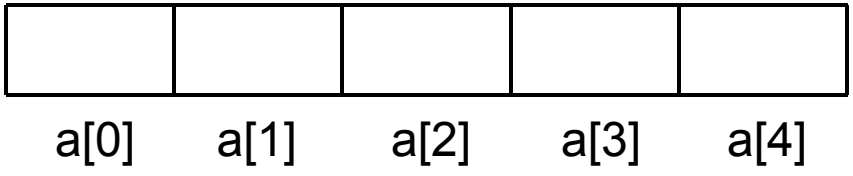
```c
    i = 0;
while( i< N)
    a[i++] = 0;
```

```c
    i = 0;
while( i< N)
    a[i] = b[i++];
```

```c
    i = 0;
while( i< N)
    a[++i] = 0;
```

```c
    i = 0;
for( i = 0; i < N; i++)
    a[i] = b[i];
```

数组：一维数组回顾



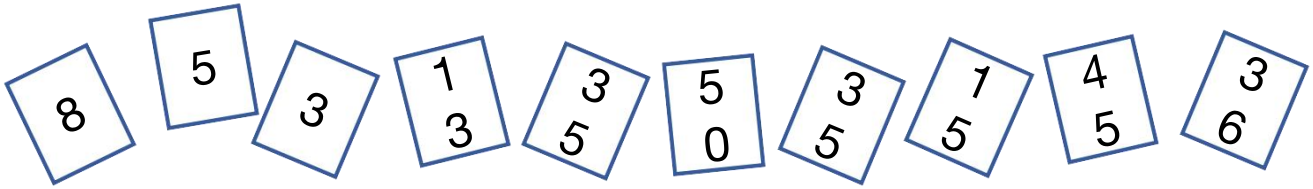
[]: *subscript* operator（下标运算符）

- #define NUM 5
- 初始化
  - int nums[NUM] = {0};
    - {0,0,0,0,0}
  - int nums[] = {0};
    - {0}
  - int nums[NUM] = {1};
    - {1,0,0,0,0}
  - int nums[NUM] = {[2]=1}; //指示器
    - {0,0,1,0,0}

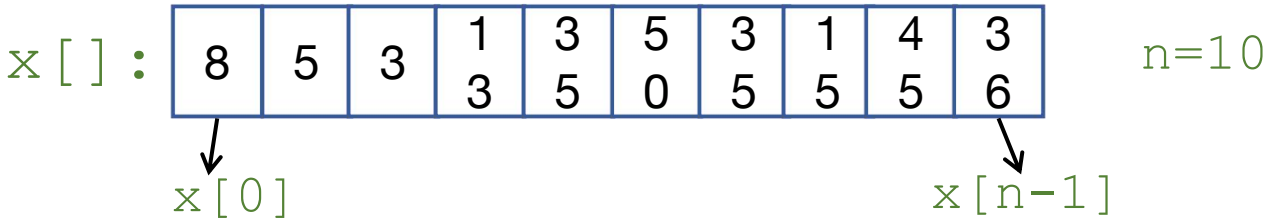
数组和循环的应用

- 排序类
  - 选择排序，冒泡排序，快速排序……

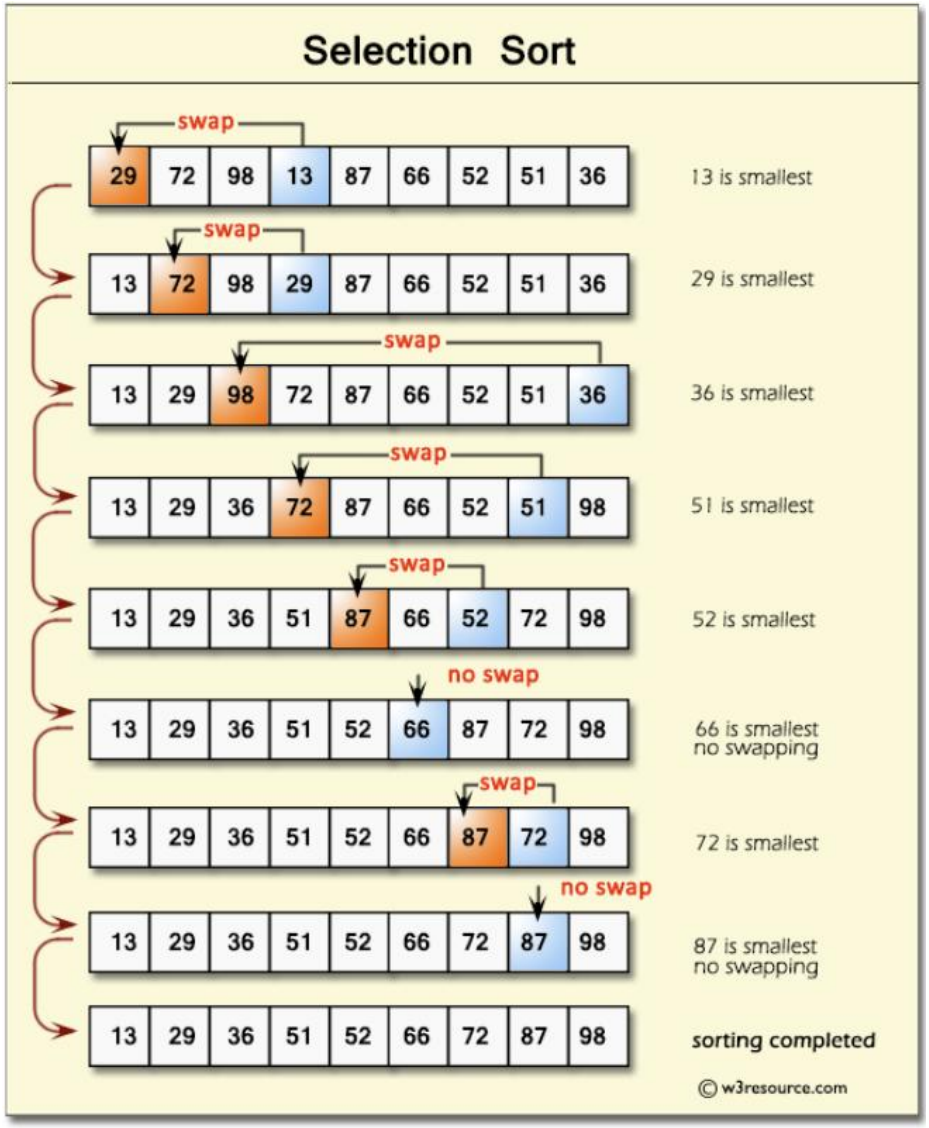
- 数组排序问题
  - 无序数组变为有序数组
    - “将一组n个整型数从小到大排列”



  - 如何组织此n个整形数？
    - 以一维数组方式组织



选择排序


Selection Sort
swap
29 72 98 13 87 66 52 51 36
13 is smallest
swap
13 72 98 29 87 66 52 51 36
29 is smallest
swap
13 29 98 72 87 66 52 51 36
36 is smallest
swap
13 29 36 72 87 66 52 51 98
51 is smallest
swap
13 29 36 51 87 66 52 72 98
52 is smallest
no swap
13 29 36 51 52 66 87 72 98
66 is smallest
no swapping
swap
13 29 36 51 52 66 87 72 98
72 is smallest
no swap
13 29 36 51 52 66 72 87 98
87 is smallest
no swapping
13 29 36 51 52 66 72 87 98
sorting completed
© w3resource.com

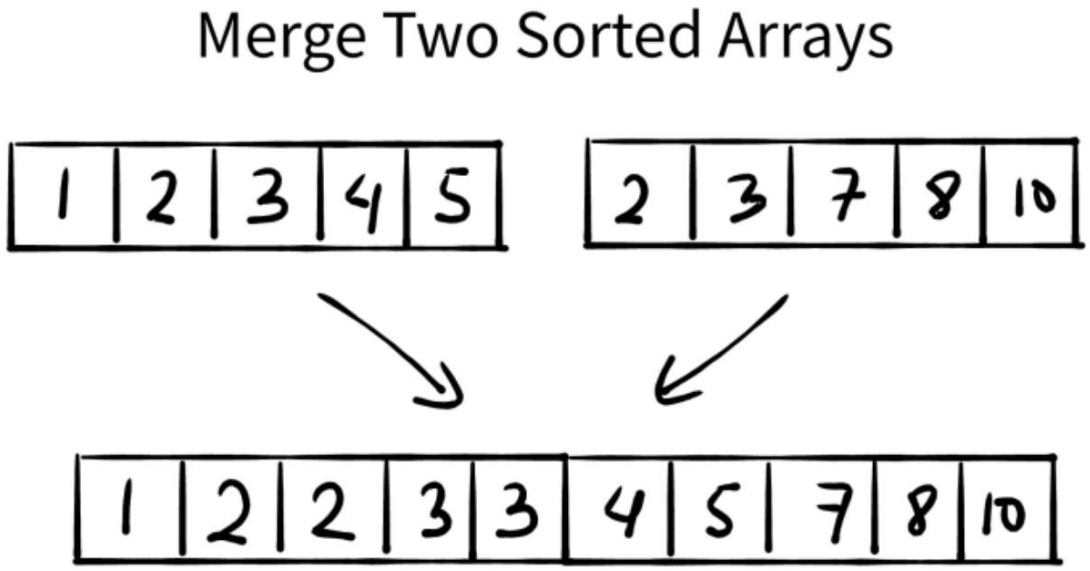
冒泡排序

- 基本思想
  - 重复地走访过要排序的元素列，依次比较两个相邻的元素并按需交换（目标：从小到大排列）
    - 若`x[i]>x[i+1]`，则交换
  - 直观表达，每一轮遍历，将一个最大的数移到序列末尾

  - bubblesort.c

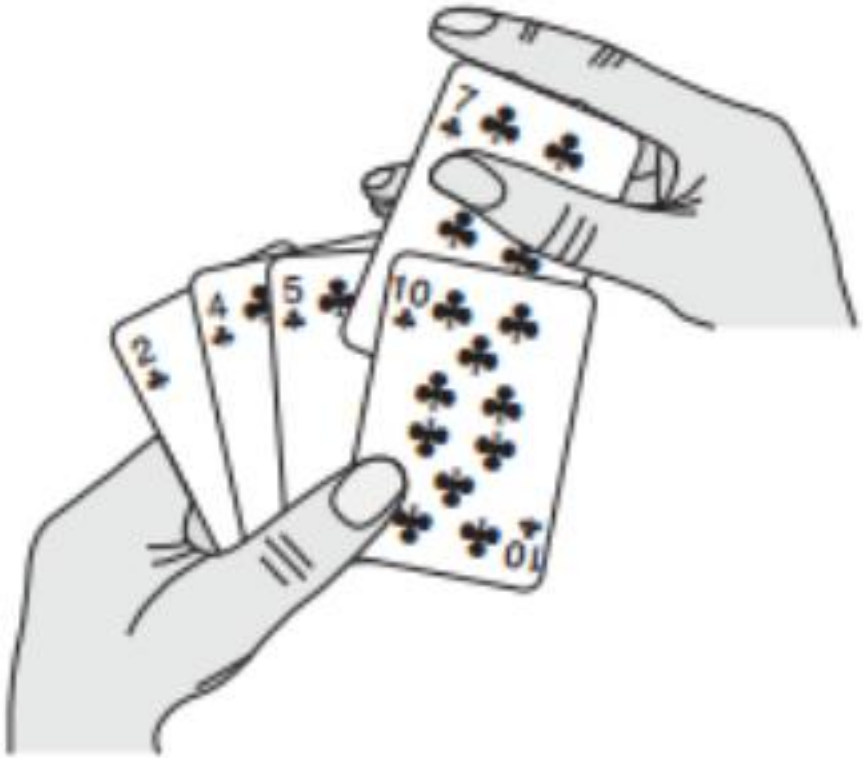
Merge Sort

- mergesort.c

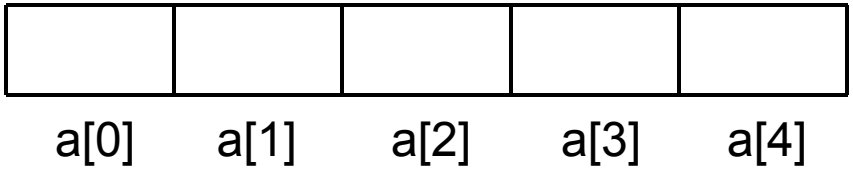


插入排序

insertsort.c

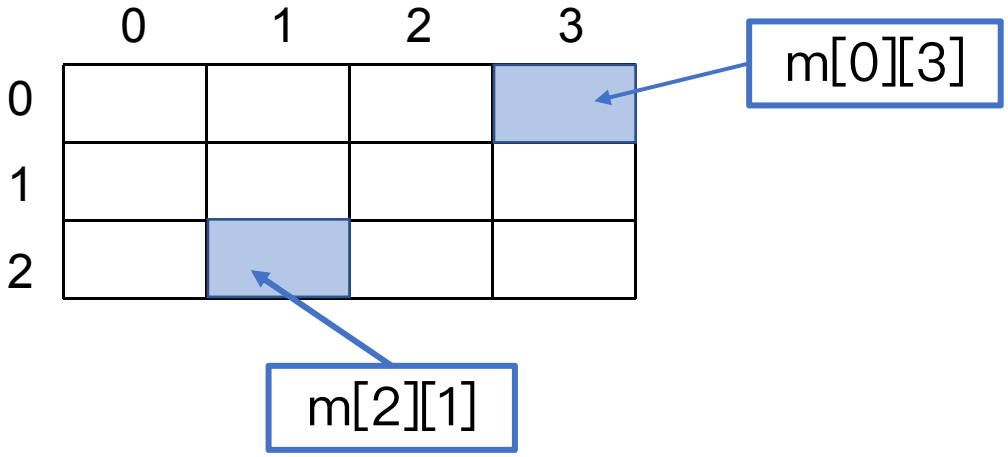


多维数组



[]: *subscript* operator (下标运算符)

- 一种特殊的一维数组
- 如：二维数组 int m[3][4];

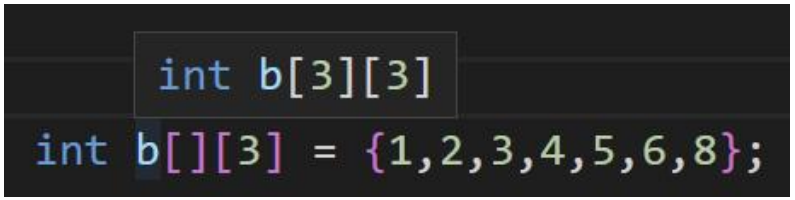


多维数组

- 初始化：类似嵌套一维数组初始化实现
  - int a[2][3] = {{1,2,3}, {4,5,6}};
  - int a[2][3] = {{1,2,3}};

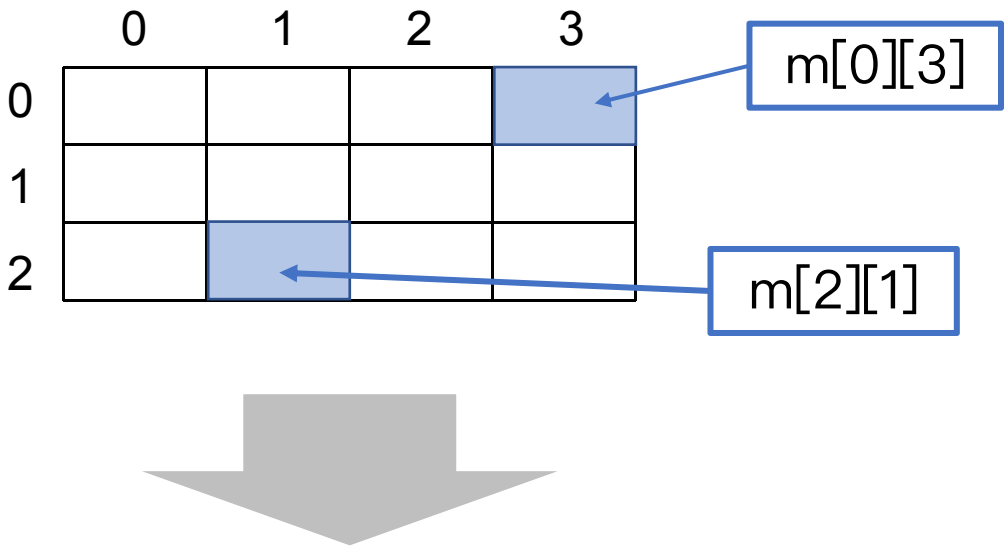
  - int a[2][3] = {1,2,3,4,5,6};
  - int a[ ][3] = {1,2,3,4,5,6};

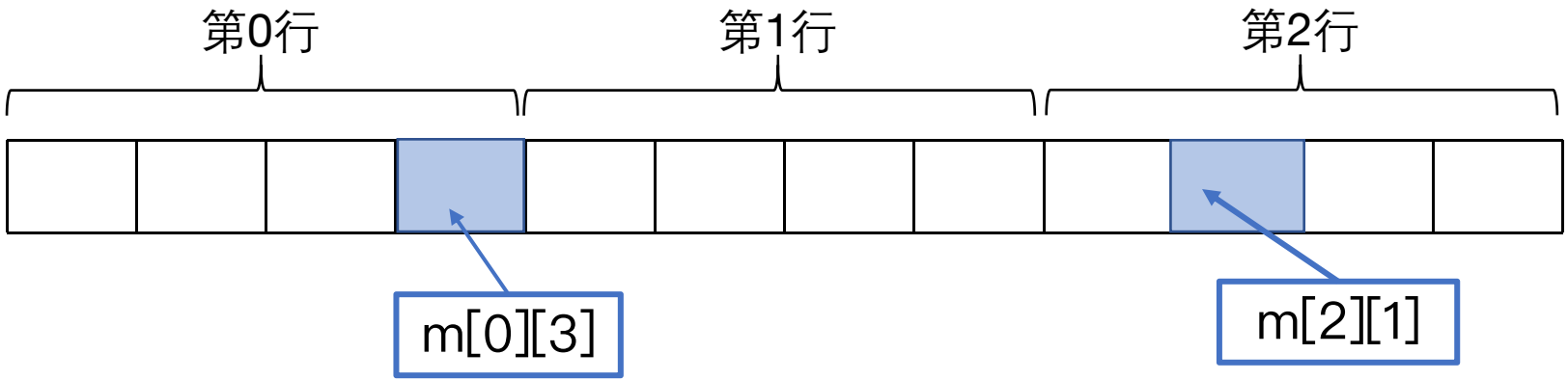
  - int a[2][3] = {[0][0] = 1, [2][1] = 5};



多维数组的存储

- 多维数组在内存中以行主存存储数组





多维数组

- multiarray.c
- multiarray2.c

- chararray.c

五子棋

- 需求：
  - 绘制棋盘
  - 两位棋手下棋
  - 判断是否棋局终止

- chess.c

矩阵相乘

- 数组可以用于存储一组有序或无序数据
  - 二维数组天然能够直接表示矩阵
  - 设有两个矩阵A，B，求其乘积结果
  - matrixMul.c

$$(AB)_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{ip} b_{pj}$$

如下所示：

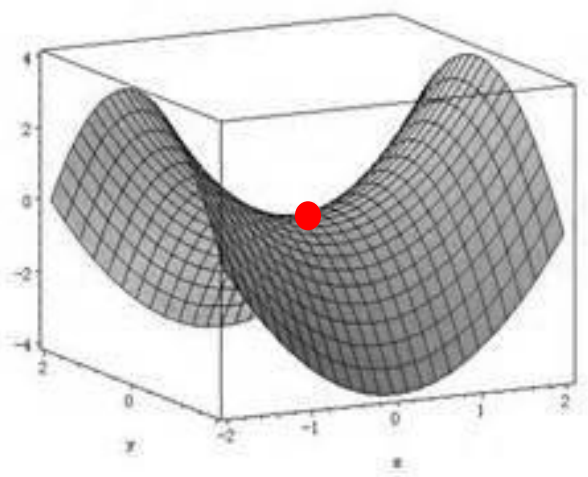
$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \end{bmatrix}$$

$$C = AB = \begin{bmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} + a_{1,3}b_{3,1}, & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} + a_{1,3}b_{3,2} \\ \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} + a_{2,3}b_{3,1}, & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} + a_{2,3}b_{3,2} \end{bmatrix}$$

数组也可以用来存储需要的标记

- 求矩阵的鞍点
  - 指矩阵中的一个位置，该位置上的元素是其所在行上的最大值且是一个极大值，所在列的最小值且是一个极小值
  - 一个矩阵可能有多个鞍点，也可能没有鞍点
  - anchor.c



3	7	7	9	3
3	6	3	6	3
4	5	4	5	4
3	6	3	6	3
3	8	3	8	3

常量数组

- 无论一维还是二维数组，可以声明时加上const使之成为“常量”

```
const char hex_chars[] =
            {'0', '', '2', '3', '4', '5', '6', '7', '8', '9',
                'A', 'B', 'C', 'D', 'E', 'F'};
```

变长数组VLA

- 在C99开始，数组长度也可以使用常量表达式
- 不推荐!

- 那怎么办？
  - 等待后续分解
  - 堆区 vs 栈区

End

- 关注解决问题能力
- 关注coding技术训练
- 请勿抄袭!