

数组

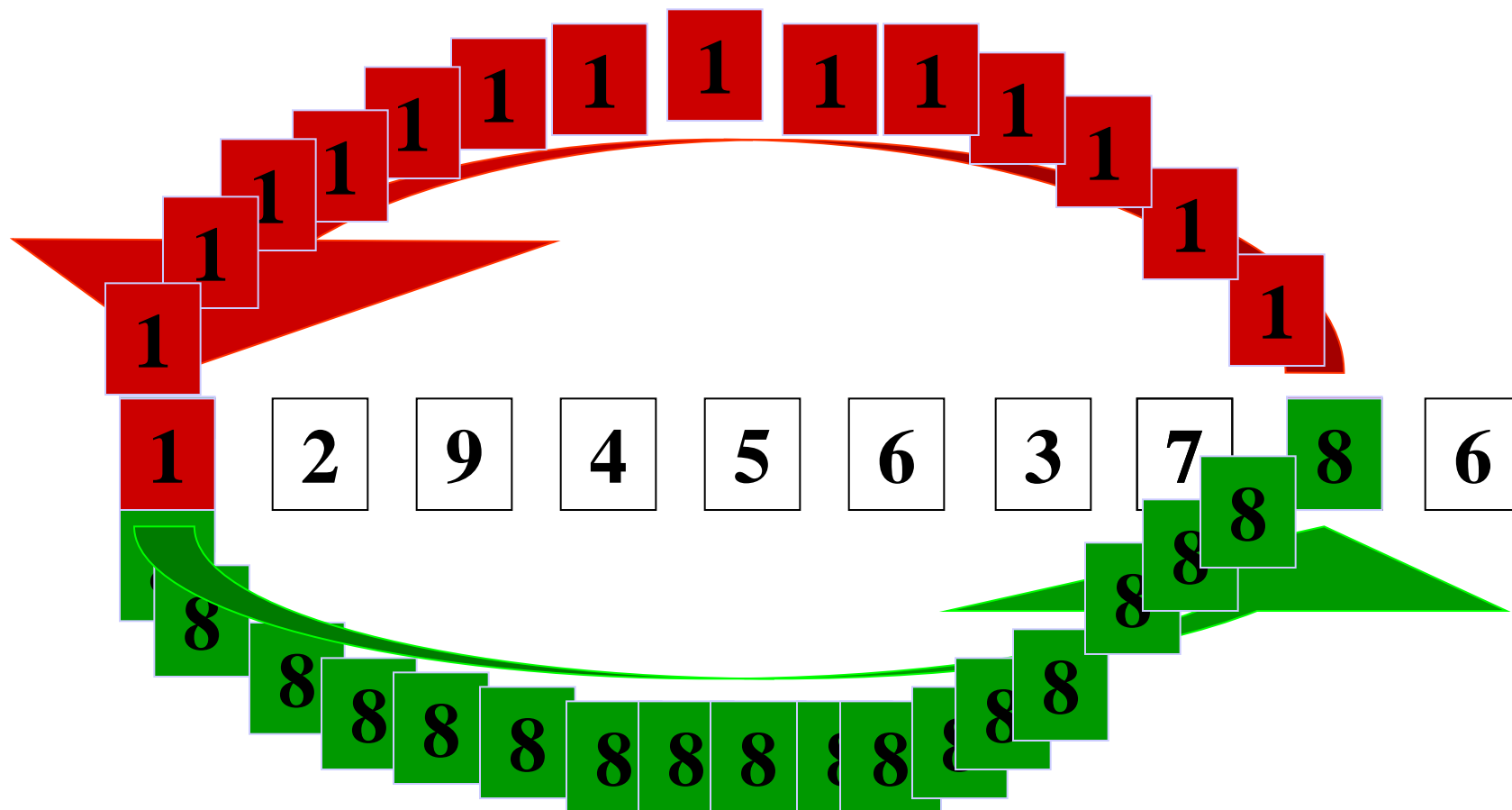
Wang Houfeng
EECS, Peking University
wanghf@pku.edu.cn

内容

➤ 数组基本概念

- 几个应用实例
- 二维/多维数组

问题引入——多个元素排序



如果有更多数据，如何存放才便于排序？

数组的特点

- 相同类型的数据集合；
- 所有数据元素都属于相同的数据类型（如整型）
- 用一个名称（**数组名**）标识整体（**集合名**）
- 元素：整体内的一个单元
- 用数组名和下标共同确定某个具体元素
 - ✓ 一个班学生的学习成绩（85，78，92，66，...）
 - ✓ 一行字符 abcdefgh
 - ✓ 一个矩阵(如， 9×9 乘法表)

这些数据的特点是：

- 1.具有相同的数据类型
- 2.使用过程中经常要保留原始数据

C/C++ 语言中的一维数组

- 数组的声明方式：

数据类型 数组名 [常量表达式];

[] :数组运算符
不能用()

合法标识符

表示元素个数: 从0开始编号

例 int a[6];

计算机分配连续内存
所占内存字节数=数组元素个数*
sizeof(元素数据类型)

数组名表示数组在内存的首地址,
是地址常量

a →

0	a[0]
1	a[1]
2	a[2]
3	a[3]
4	a[4]
5	a[5]

类型 数组名 [常量表达式];

- 类型可以是整型、浮点型、字符型，甚至用户定义的类型，如指针型、结构型和枚举型等
- 类型表示数组的每个元素的数据类型
- 常量表达式的值为整型，规定了元素的个数（也称数组长度）。数组的起始元素总是从0开始，到n-1（n表示元素个数），不能用圆括号。
- 数组名满足标识符的命名规则，是数组变量。

例：

char ch[30];

相当于30个字符变量

表示数组共30个元素：ch[0], ch[1], ..., ch[29]，每个元素为字符型，每个元素可以当做变量独立使用！

int grade[50]; 表示有50个元素，每个元素为整型变量

数组定义的常见问题

- 定义中**常量表达式**表示数组大小，不能含变量；
- 例如：

```
int n;
```

```
scanf("%d", &n); /*在程序中临时输入数组的大小*/
```

```
int a[n]; /* 数组大小不能由变量表示*/
```

- 数组说明中其他常见的错误：
 - ①float a[0]; /* 数组大小为0没有意义*/
 - ②int b(2); /* 不能使用圆括号*/
 - ③int k, a[k]; /* 不能用变量说明数组大小*/

数组的引用

所谓引用，就是使用。只能引用数组元素：

数组名 [下标]

下标可以是整型常量或整型表达式（表达式中可以有变量）

例如： $a[0]=a[5]+a[7]-a[2*3]$

每个元素可以独立使用，由下标确定具体元素

注意：

定义数组时用到的“数组名 [常量表达式]”和引用数组元素时用到的“数组名 [下标]”是有区别的。

例如： `int a[10]; /*定义大小为10的数组*/`
`t=a[6]; /*引用编号为 6 的元素*/`

数组引用的问题

- ❖ 必须**先声明数组名**，**然后才能引用数组元素**
- ❖ 注意不要越界（越界时系统不会报错）
- ❖ 只能引用数组元素，不能引用数组整体

```
例 int data[5];  
    data[5]=10;  
// × C语言对数组不作越界检查，程序员要控制
```

```
例    int a[10];  
        cout<<a; // (×) 不能整体操作  
必须 for(j=0;j<10;j++)  
        cout<<a[j]<<'\\t';           (✓)
```

一维数组的初始化

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

等价于：a[0]=1; a[1]=2; a[2]=3; a[3]=4; a[4]=5;

❖说明：

- 如果数组不初始化，其元素值为不确定值
- 若static数组元素不赋初值，系统会自动赋以0值

```
static int a[5];
```

等价于：a[0]=0; a[1]=0; a[2]=0; a[3]=0; a[4]=0;

- 如何只给部分数组元素赋初值，其余元素默认初始为0.

如 int a[5]={6,2,3};

等价于：a[0]=6; a[1]=2; a[2]=3; 但 a[3] 和 a[4] 为0;

注意：int a[3]={6,2,3,5,1}; (×) 个数不能超限

- 当全部数组元素赋初值时，可不指定数组长度

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

编译系统根据初值个数确定数组维数

内容

- 数组基本概念
- 几个应用实例
 - 二维/多维数组

实例分析，指出功能：

步骤:

1. **输入**: for循环输入10个整数
2. **处理**:
 - (a) 先令 $\text{max}=\text{min}=\text{x}[0]$ (为什么?)
 - (b) 依次用 $\text{x}[i]$ 和 max, min 比较(循环)
若 $\text{max}<\text{x}[i]$, 令 $\text{max}=\text{x}[i]$
若 $\text{min}>\text{x}[i]$, 令 $\text{min}=\text{x}[i]$
可以优化吗?
3. **输出**: max 和 min

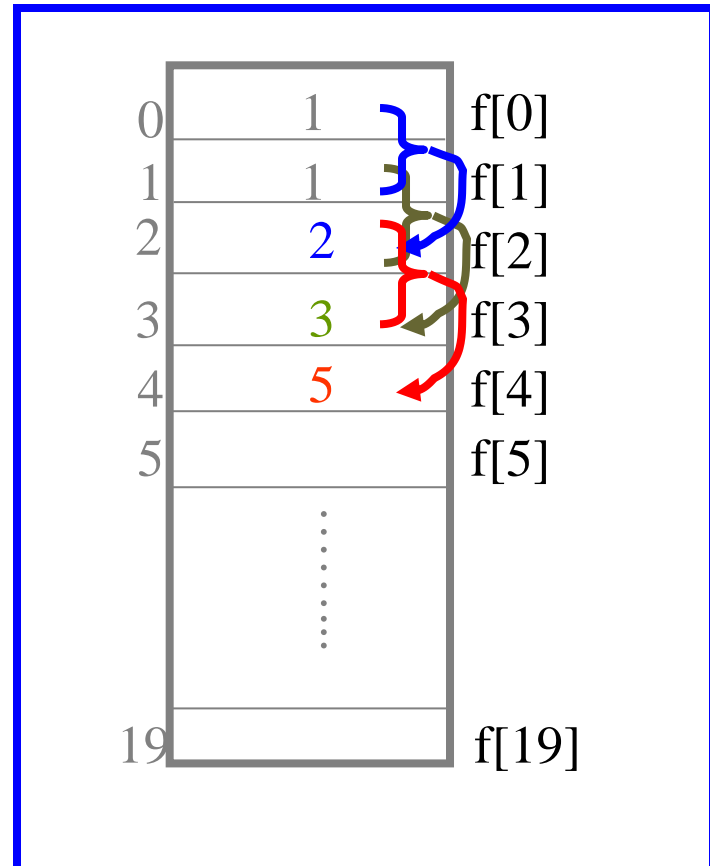
```
#include <iostream>
using namespace std;
#define SIZE 10
int main()
{   int x[SIZE], i, max, min;
    cout<<"Enter 10 integers:"<<endl;
    for(i=0; i<SIZE; i++)
    {   cout<<"\n"<<i+1<<" = ";
        cin>>x[i];
    }
    max=min=x[0];
    for(i=1; i<SIZE; i++)
    {   if(max<x[i]) max=x[i];
        if(min>x[i]) min=x[i];
    }
    cout<<"Maximum value is "<<max<<endl;
    cout<<"Minimum value is "<<min<<endl;
    return 0;
}
```

用数组求Fibonacci数列前20个数

定义=

$$\begin{aligned} F_1 &= 1 & (n = 1) \\ F_2 &= 1 & (n = 2) \\ F_n &= F_{n-1} + F_{n-2} & (n \geq 3) \end{aligned}$$

```
#include <iostream>
#include<iomanip>
using namespace std;
int main()
{   int i;
    int f[20]={ 1,1 };
    for(i=2;i<20;i++)
        f[i]=f[i-2]+f[i-1];
    for(i=0;i<20;i++)
    {   if(i%5==0) cout<<endl;
        cout<<setw(8)<<f[i];
    }
    return 0;
}
```



统计数字个数

- 输入20个0~9之间的整数，请统计每个数在输入数列中出现的次数
- 问题：统计的次数如何表示？
 - 数组下标对应0~9 的10个数，每个元素记录次数

4	5	7	4	3	2	4	5	7	7	7	4	3	2	2	4	6	7	8	4
2	输入了3次																		
3	输入了2次																		
4	输入了6次																		
5	输入了2次																		
6	输入了1次																		
7	输入了5次																		
8	输入了1次																		
a[0]		a[1]		a[2]		a[3]		a[4]		a[5]		a[6]		a[7]		a[8]		a[9]	
0		0		3		2		6		2		1		5		1		0	

统计数字个数

```
#include <iostream>
using namespace std;
int main()
{
```

```
    int num; count[10] = {0};
    for (int i = 1; i <= 1000; i++)
    {
        cin >> num;
        switch (num)
```

```
        {
            case 0: count[0]++; break;
            case 1: count[1]++; break;
            case 2: count[2]++; break;
            case 3: count[3]++; break;
            case 4: count[4]++; break;
            case 5: count[5]++; break;
            case 6: count[6]++; break;
            case 7: count[7]++; break;
            case 8: count[8]++; break;
            case 9: count[9]++; break;
        }
```

```
    }
    for (int i = 0; i < 10; i++)
    {
        if (count[i] != 0)
            cout << i << "输入了" << count[i] << "次" << endl;
    }
    return 0;
}
```

完全对应

程序结构可以更简洁

统计数字个数

```
#include <iostream>
using namespace std;
int main()
{
    int num, count[10] = {0}; //数组的每个元素初始化为0
    for (int i = 0; i < 20; i++)
    {
        cin >> num;
        if (num > -1) &&(num<10)    count[num]++;
        //若输入值在0~9之间，则对应元素增1（计数）
    }
    for (int i = 0; i < 10; i++)
    {
        if (count[i] != 0)
            cout << i << "输入了" << count[i] << "次" << endl;
    }
    return 0;
}
```


统计问题扩展（思考）

- **扩展1:** 如果输入一串符号（包括数字），统计出现的每个数字（0~9）次数
- **扩展2:** 如果输入一串符号（包括字母），统计出现的每个字母（不区分大小写）次数。

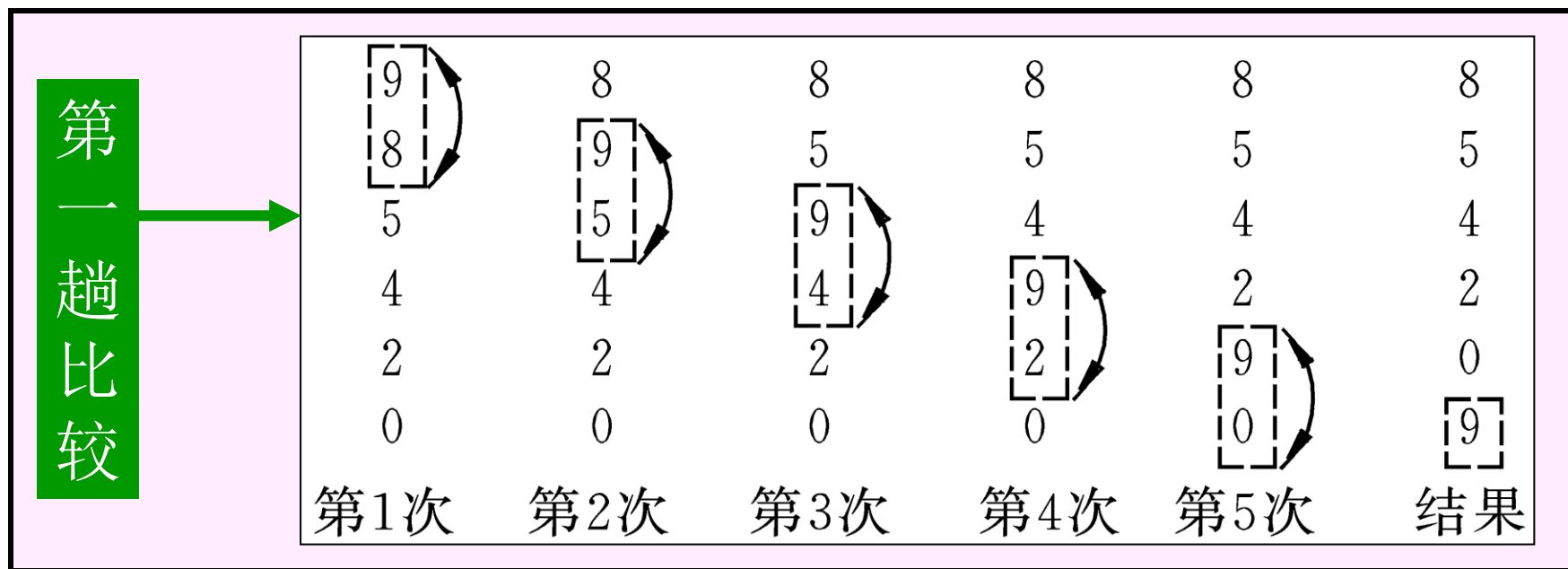
例 输入为:

1234012359078abcd3#defa0893

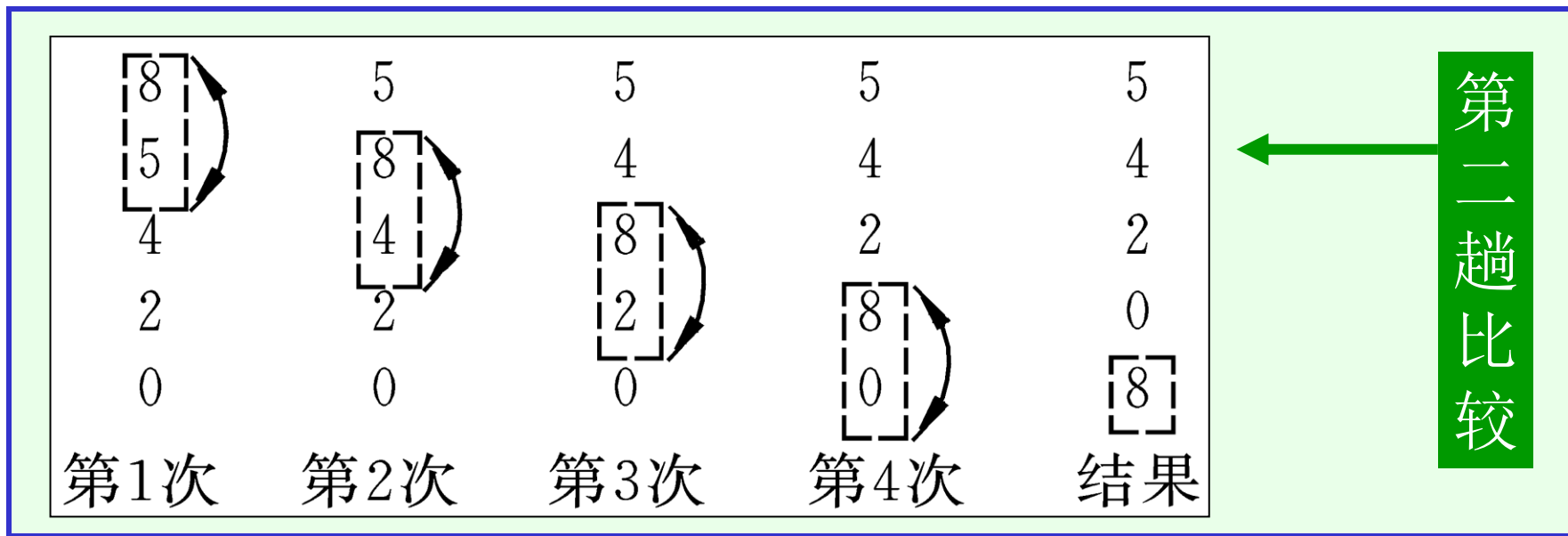
冒泡排序

—— 一个复杂的应用

对 n 个数进行冒泡排序



- 经过第一趟(共 $(6-1)=5$ 次比较与交换)后, 最大的数9已“**沉底**”。然后进行对余下的前面5个数第二趟比较。



规律:

如果有 n 个数，则要进行 $n-1$ 趟比较。在第1趟比较中要进行 $n-1$ 次两两比较，在第 j 趟比较中要进行 $n-j$ 次两两比较。

49	38	38	38	38	13	13	13
38	49	49	49	13	27	27	27
65	65	65	13	27	30	30	
97	76	13	27	30	38		
76	13	27	30	49			
13	27	30	65				
27	30	76					
30	97						
初始关键字	第一趟	第二趟	第三趟	第四趟	第五趟	第六趟	第七趟

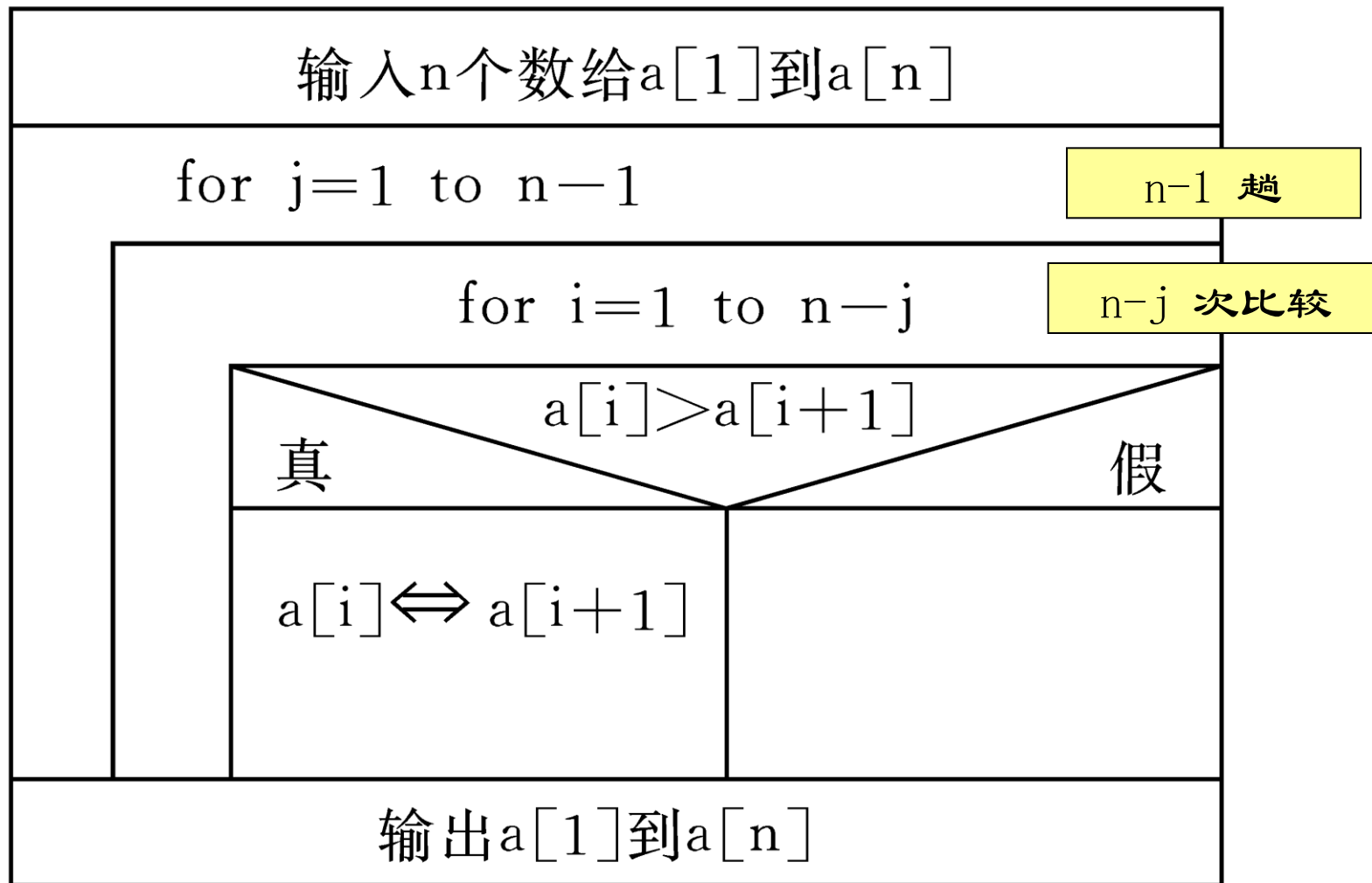
n=8

8个元素的更详细排序过程

排序过程：

- (1) 比较第一个数与第二个数，若为逆序 $a[0]>a[1]$ ，则交换；然后比较第二个数与第三个数；依次类推，直至第 $n-1$ 个数和第 n 个数比较为止——**第一趟冒泡排序**，结果**最大**的数被置换到最后一个元素位置上
- (2) 对前 $n-1$ 个数进行**第二趟冒泡排序**，结果使**次大**的数被安置在第 $n-1$ 个元素位置
- (3) 重复上述过程，共经过 $n-1$ 趟冒泡排序后，排序结束

大家自己绘制程序流程图！



程序实现

```
#include <iostream>
using namespace std;
int main()
{   int a[11],i,j,t,n=10;
    cout<<"Input 10 numbers:"<<endl;
    for(i=1;i<11;i++)
        cin>>a[i];
    cout<<endl;
    for(j=1;j<n;j++)
        for(i=1;i<=n-j;i++)
            if(a[i]>a[i+1])
                {t=a[i]; a[i]=a[i+1]; a[i+1]=t;}
    cout<<"The sorted numbers:"<<endl;
    for(i=1;i<11;i++)
        cout<<setw(8)<<a[i];
    return 0;
}
```

二重循环结构：

- (1) 用一个变量 j 控制趟数； n 个元素共 $n-1$ 趟；
- (2) 在第 j 趟内，需要比较的 $(n-j)$ 次，另一变量 i 控制

问题：

变量的含义是什么？

❖ 为什么引入它们？

需要哪些运算？

❖ 元素间的比较运算

❖ 元素间的交换运算

选择排序

—— 另一个复杂的应用

例

初始: $i=1$

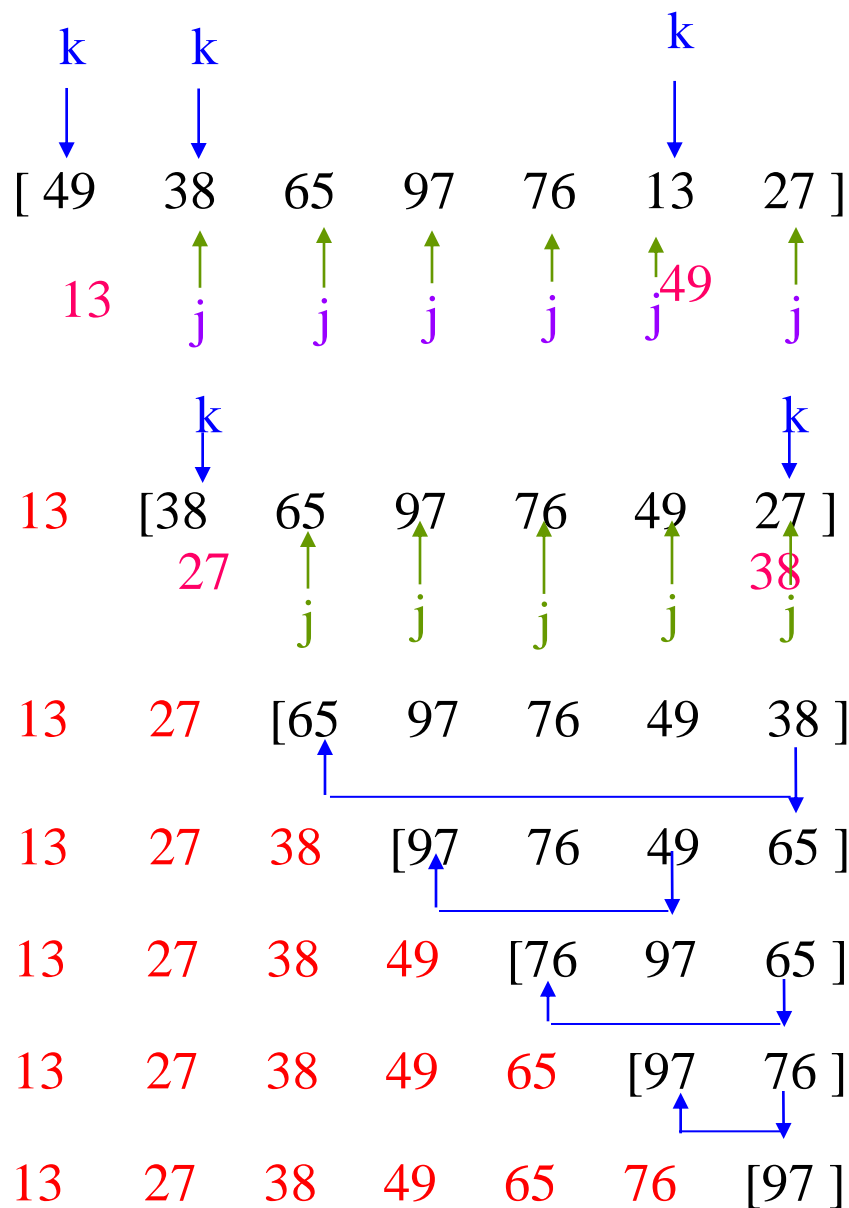
$i=2$

$i=3$

$i=4$

$i=5$

$i=6$



i 的含义?
 j 的含义?
 k 的含义?

```

#include <iostream>
using namespace std;
int main()
{   int a[11],i,j,k,x;
    cout<<"Input 10 numbers:"<<endl;
    for(i=1;i<11;i++)
        cin>>a[i];
    cout<<endl;
    for(i=1;i<10;i++)
    {   k=i;
        for(j=i+1;j<=10;j++)
            if(a[j]<a[k]) k=j;
        if(i!=k)
        {   x=a[i]; a[i]=a[k]; a[k]=x; }
    }
    cout<<"The sorted numbers: "<<endl;
    for(i=1;i<11;i++)
        cout<<a[i];
}

```

基本思想：

- (1) 首先通过 $n-1$ 次比较，从 n 个数中找出最小的，将它与第一个数交换——**第一趟选择排序**，结果**最小**的数被安置在第一个元素位置上
- (2) 再通过 $n-2$ 次比较，从剩余的 $n-1$ 个数中找出关键字**次小**的记录，将它与第二个数交换——**第二趟选择排序**
- (3) 重复上述过程，共经过 $n-1$ 趟排序后，排序结束

明确变量的含义与作用！

❖ 为什么引入它们？

需要哪些运算？

❖ 元素间的比较运算

❖ 元素间的交换运算

例子：用筛选法求1-1000的素数

1	2	3	4	5	6	7	8	9	...
---	---	---	---	---	---	---	---	---	-----

$i=2$ 为素数，输出，同时，筛选掉2的倍数

1	2	3	4	5	6	7	8	9	...
---	---	---	---	---	---	---	---	---	-----

$i=3$ 为素数（未被筛选），输出，同时，筛选掉3的倍数

1	2	3	4	5	6	7	8	9	...
---	---	---	---	---	---	---	---	---	-----

$i=???$ （当前的 i 为什么值？）

...

```
char prime[1000];
```

```
int i, j, m;
```

```
prime[1]=0;
```

```
prime[2]=1;
```

```
prime[1000]=0;
```

```
for(i=3; i<999;)
```

```
{
```

```
    prime[i++]=1;
```

```
    prime[i++]=0;
```

```
}
```

```
i=3;
```

```
m=sqr(1000);
```

```
while (i<m)
```

```
{
```

```
    for(j=i+1; j<1000;++j)
```

```
        if (prime[j]) prime[j] = j % i;
```

```
    i+=2;
```

```
}
```

大家试着写出程序！

内容

- 数组基本概念
- 几个应用实例
- 二维/多维数组

二维数组

格式：

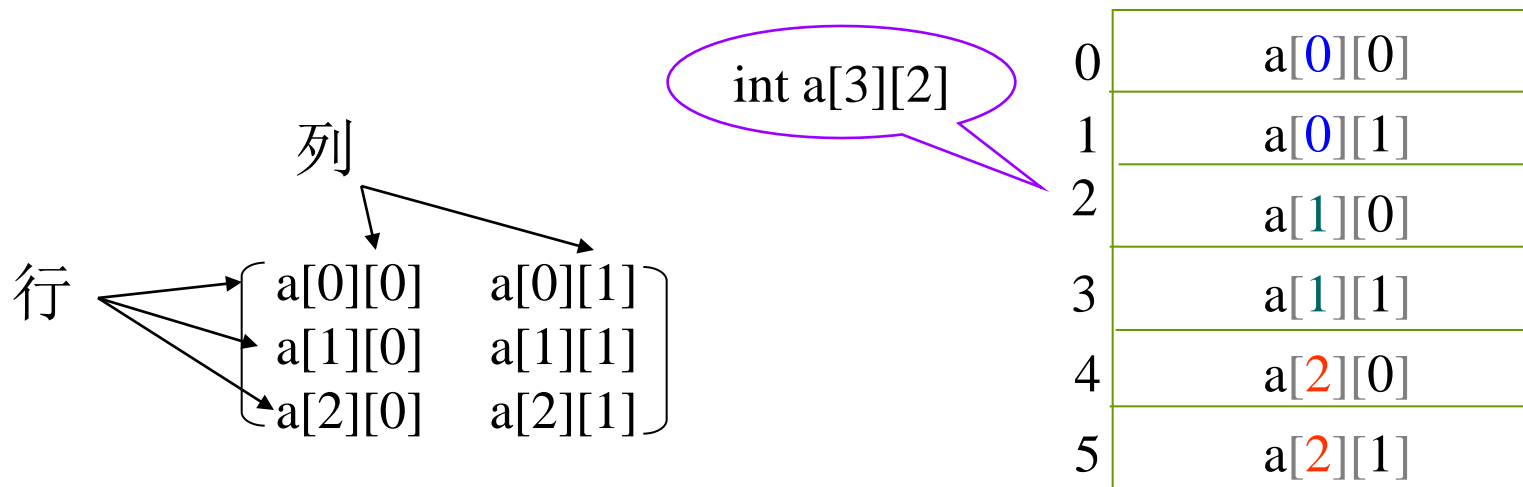
数据类型 数组名 [常量表达式] [常量表达式];

例：int a[3][2]; //二维数组

int c[2][3][4]; //三维数组

int a[3,4]; **X**

数组元素在计算机中的存放顺序（先行后列，顺序存放）



按行存储

例如：int a[3][4];

a[0][0] a[0][1] a[0][2] a[0][3]

a[1][0] a[1][1] a[1][2] a[1][3]

a[2][0] a[2][1] a[2][2] a[2][3]

1	2	3	4
5	6	7	8
9	10	11	12

a[0][0]

1

a[0][1]

2

a[0][2]

3

a[0][3]

4

a[1][0]

5

a[1][1]

6

a[1][2]

7

a[1][3]

8

a[2][0]

9

a[2][1]

10

a[2][2]

11

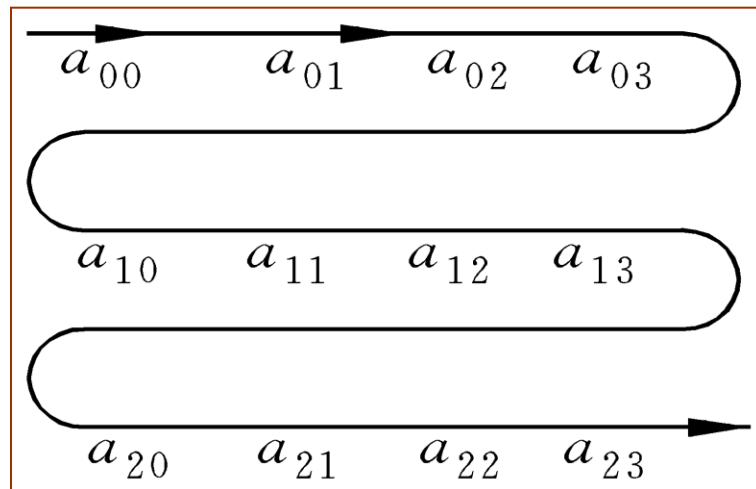
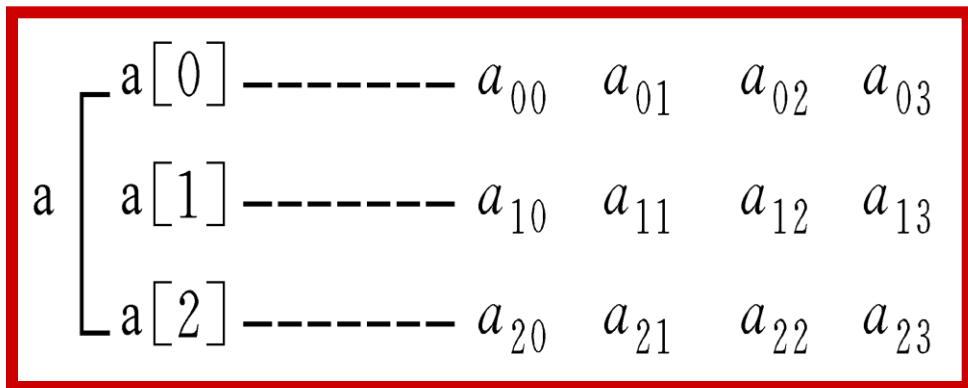
a[2][3]

12

二维数组：特殊的一维

注意： 二维数组可以看成是一种特殊的一维数组：即，它的元素是一个一维数组。

例如： 把a看作是一个一维数组，它有3个元素： $a[0]$ 、 $a[1]$ 、 $a[2]$ ，每个元素又是含4个元素的一维数组。



三维数组

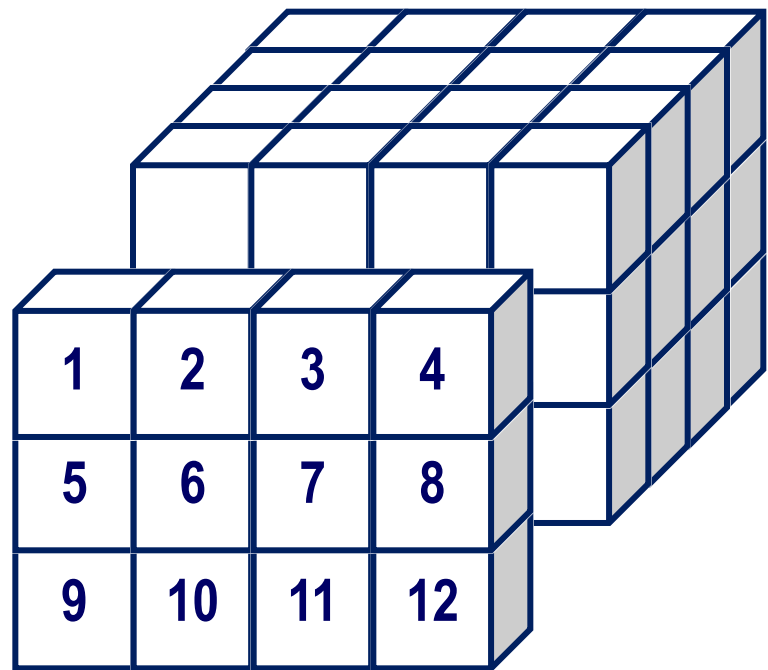
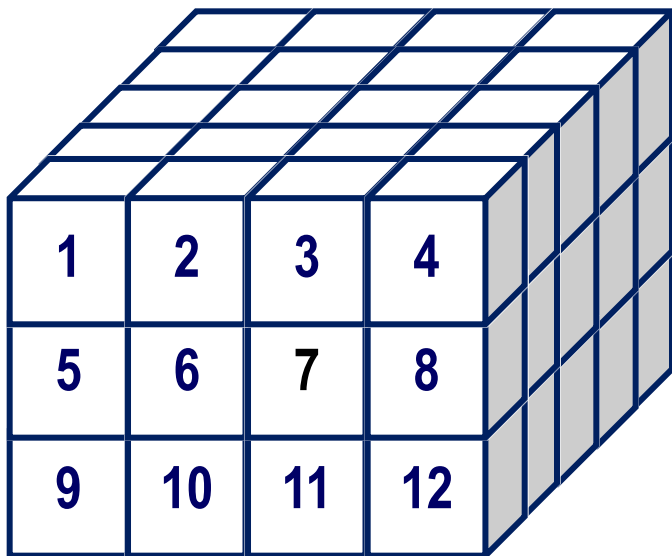
- 问题：有了二维数组的基础，多维数组如何定义呢？

定义三维数组：float a[2][3][4];

- **注意：**多维数组元素在内存中的排列顺序：第一维的下标变化最慢，最右面的下标变化最快。

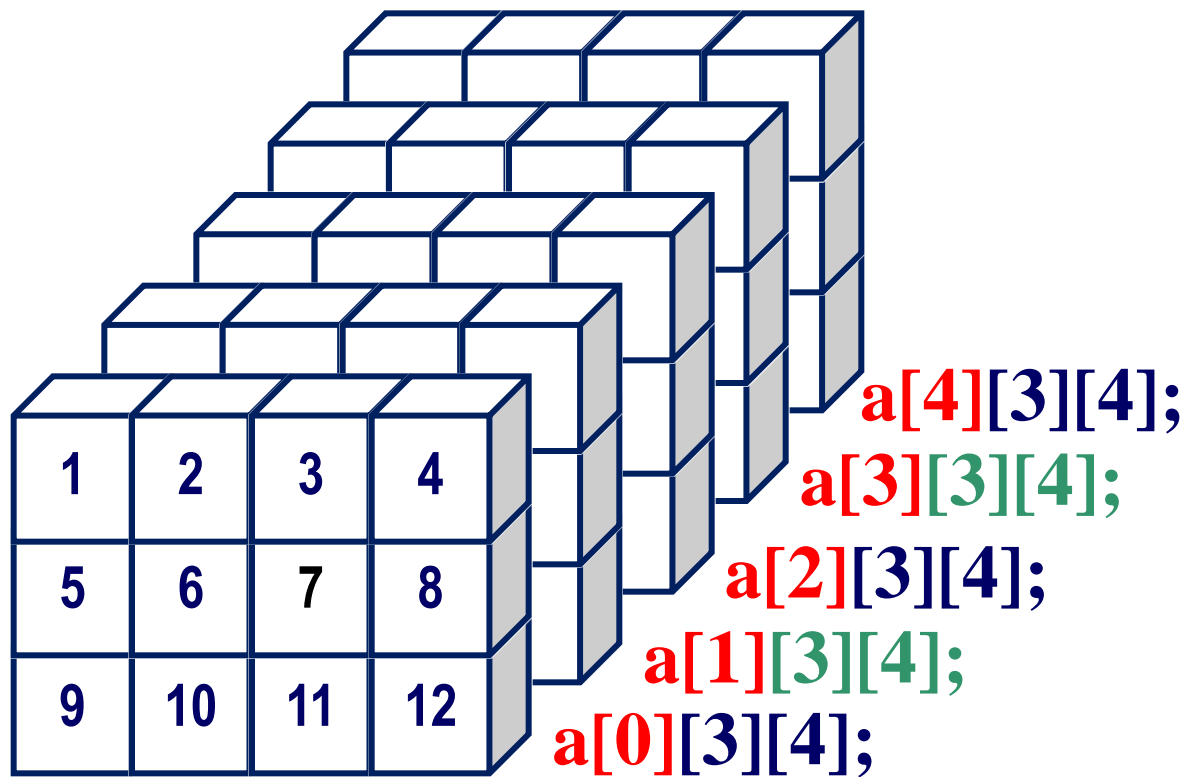
三维数组

```
int a[5][3][4];
```



a[3][4];

三维数组存储对应关系



...
$a[0][0][0]$
$a[0][0][1]$
$a[0][0][2]$
$a[0][0][3]$
$a[0][1][0]$
$a[0][1][1]$
$a[0][1][2]$
$a[0][1][3]$
$a[0][2][0]$
$a[0][2][1]$
$a[0][2][2]$
$a[0][2][3]$
$a[0][3][0]$
$a[0][3][1]$
$a[0][3][2]$
$a[0][3][3]$
$a[1][0][0]$
$a[1][0][1]$
.....
$a[1][3][3]$
$a[2][0][0]$
$a[2][0][1]$
.....
$a[2][3][3]$
$a[3][0][0]$
.....
$a[3][3][3]$
$a[4][0][0]$
.....
$a[4][3][3]$

举例

```
int main()
{
```

```
    int a[5][3][4] = { 0 };
```

```
    for (int i = 0; i < 5; i++)
```

```
    for (int j = 0; j < 3; j++)
```

```
    for (int k = 0; k < 4; k++)
```

```
        a[i][j][k] = 12 * i + 4 * j + k + 1;
```

```
    for (int i = 0; i < 5; i++)
```

```
    {
```

```
        for (int j = 0; j < 3; j++)
```

```
        {
```

```
            {
```

```
                for (int k = 0; k < 4; k++)
```

```
                cout << setw(3) << a[i][j][k];
```

```
            }
```

```
            cout << endl;
```

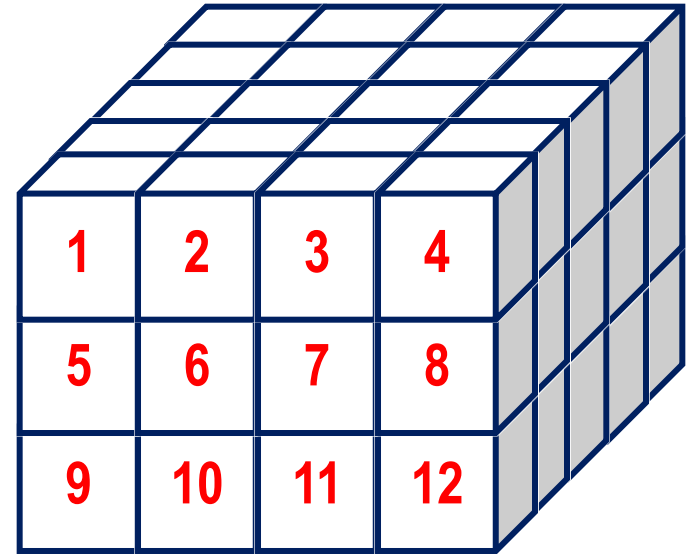
```
        }
```

```
        cout << endl;
```

```
    }
```

```
    return 0;
```

```
}
```



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28
29	30	31	32
33	34	35	36
37	38	39	40
41	42	43	44
45	46	47	48
49	50	51	52
53	54	55	56
57	58	59	60

二维数组初始化

- 四种初始化：

1. 分行赋初值。

例如：

```
int a[3][4]={ {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}};
```

2. 可以将所有数据写在一个花括号内，按数组排列的顺序对各元素赋初值。

例如：

```
int a[3][4]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

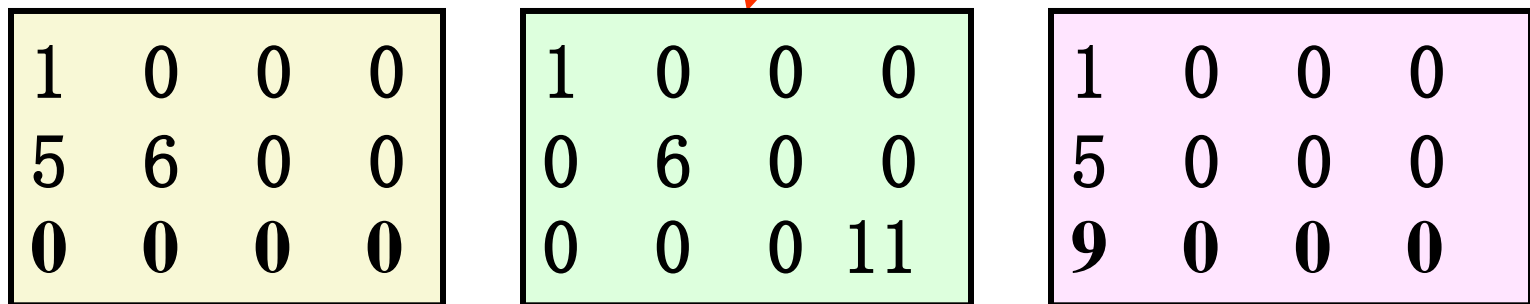
二维数组初始化(续)

3. 可以对部分元素赋初值。

例如： `int a[3][4]={ {1}, {5}, {9} };`

也可以对各行中的某一元素赋初值，如

`int a[3][4]={ {1}, {0, 6}, {0, 0, 0, 11} };`



1	0	0	0
5	6	0	0
0	0	0	0

1	0	0	0
0	6	0	0
0	0	0	11

1	0	0	0
5	0	0	0
9	0	0	0

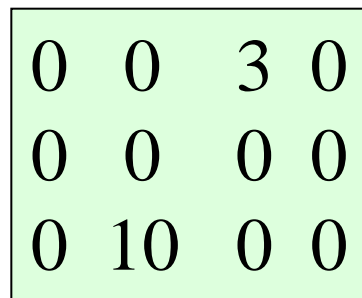
也可以只对某几行元素赋初值。如：

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

二维数组初始化(续)

4. 如果对全部元素都赋初值，则定义数组时对**第一维**的长度可以不指定，但第二维的长度不能省。

例如： `int a[3][4]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};` 它等价于： `int a[][4]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};`



0	0	3	0
0	0	0	0
0	10	0	0

在定义时也可以只对部分元素赋初值而省略第一维的长度，但**应分行**赋初值。

例如： `int a[][4]={{0, 0, 3}, {}, {0, 10}};`

引用

二维数组元素的表示形式为：

数组名[下标][下标]

例如： $a[2][3]$

下标可以是整型表达式，如 $a[2-1][2*2-1]$

不要写成 $a[2, 3]$ ， $a[2-1, 2*2-1]$ 形式 

数组元素可以出现在表达式中，也可以被赋值

例如： $b[1][2]=a[2][3]/2$

注意越界问题

在使用数组元素时，注意下标值应在已定义的数组大小的范围内。

常出现的错误有：

```
int a[3][4];  /* 定义a为3×4的数组 */
```

⋮

```
a[3][4]=3; /* 边界应该是? */
```

二维数组元素的输入

- 从键盘输入二维数组元素的值

```
int i,j;  
int a[3][4];  
  
for(i = 0; i < 3; i++)  
    for(j = 0; j < 4; j++)  
        cin>>a[i][j];
```

例子： 将二维数组行列元素互换，存到另一个数组中

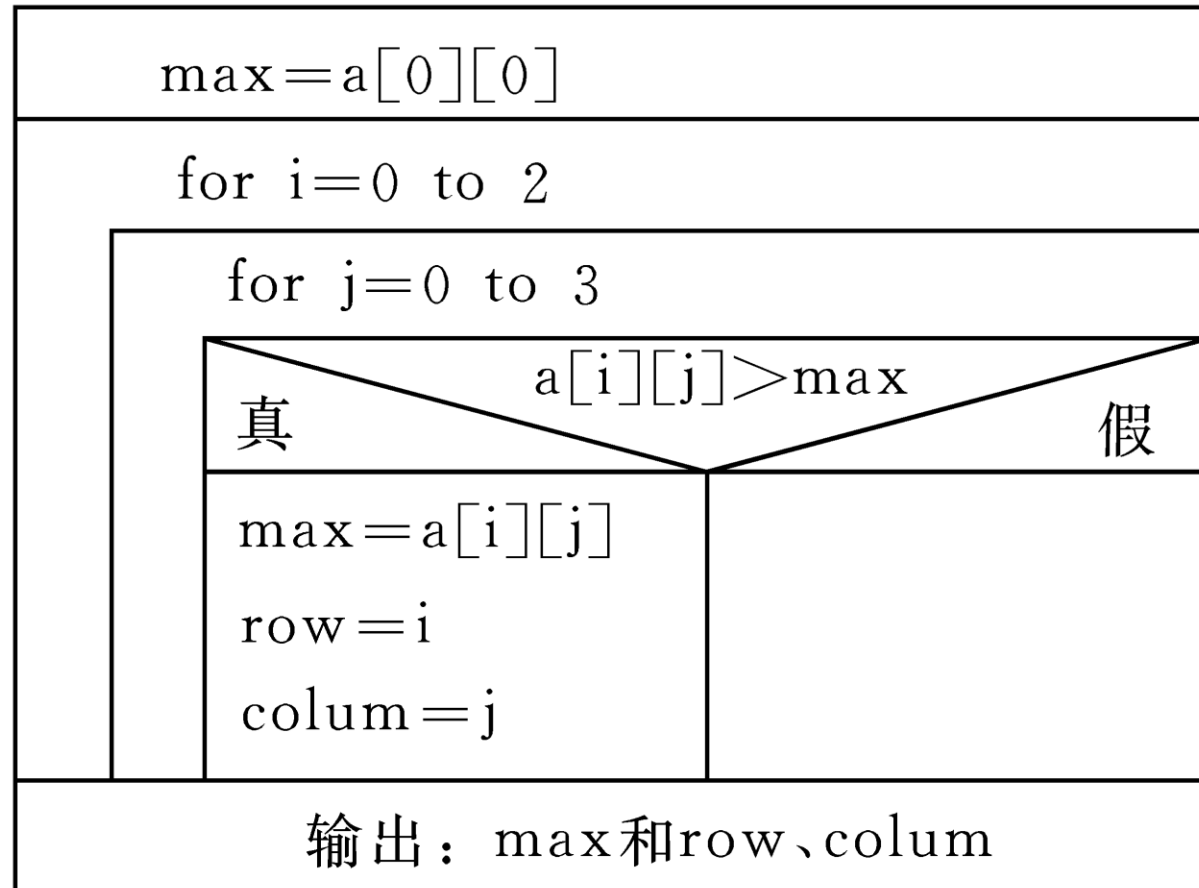
```
#include <iostream>
using namespace std;
int main()
{   int a[2][3]={ { 1,2,3},{4,5,6} };
    int b[3][2],i,j;
    cout<<"array a:"<<endl;
    for(i=0;i<=1;i++)
    {   for(j=0;j<=2;j++)
        {   cout<<a[i][j];
            b[j][i]=a[i][j];
        }
        cout<<endl;
    }
}
```

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad b = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
cout<<"array b:"<<endl;
    for(i=0;i<=2;i++)
    {   for(j=0;j<=1;j++)
        {   cout<<b[i][j];
        }
        cout<<endl;
    }
}
```

计算矩阵中最大元素的下标

假设3×4的矩阵:



```

#include <iostream>
using namespace std;
int main()
{
    int i, j, row=0, colum=0, max;
    int a[3][4]={ { 1, 2, 3, 4}, {9, 8, 7, 6}, {-10, 10, -5, 2} };
    max=a[0][0];
    for (i=0; i<3; i++)
        for (j=0; j<=3; j++)
            if (a[i][j] > max)
            {
                max=a[i][j]; /* 记录最大的值 */
                row=i;      /* 记录值最大的行号 */
                colum=j;    /* 记录值最大的列号 */
            }
    cout<<"max="<<max<<" ,row="<< row <<" ,colum="<< colum<<endl;
    return 0;
} /*程序结束*/

```

统计人数：二维数组应用

- 问题
 - 某学校有1000位老师，分布在20个不同的学院中，每个学院最多有12个系，请你编写一个程序，输入每位老师的所在院、系的编号（院编号1-20，系编号1-12），打印出各个系老师的数量。

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{   int teacher[21][13];
    int school, department;
    int i,j;

    for(i=0;i<1000;i++)
    {
        cin>>school>>department;
        teacher[school][department]++;
    }

    for (i=1;i<21;i++)
        for(j=1;j<13;j++)
            cout << setw(4) << teacher[i][j] ;
    cout << endl;
    return 0;
}
```

例子：9-9乘法表（三角形）

	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0	0	0	0
2	2	4	0	0	0	0	0	0	0
3	3	6	9	0	0	0	0	0	0
.....									
..									
9	9	18	27	36	45	54	63	72	81


```

int main()
{
    int nine_nine[10][10];
    int j,k
    for(j=1;j<10; ++j)
    {
        for(k=j; k<10; ++k)
            nine_nine[j][k]=0;
        nine_nine[0][j]=j;      //? ? ?
        nine_nine[j][0]=j;      //? ? ?
    }
    for(j=1; j<10; ++j)
        for(k=1; k <= j; ++k)
            nine_nine[j][k]=j*k;
    {打印输出? ? };
}

```

j, k 的作用是什么。

} 受哪一个
循环控制？