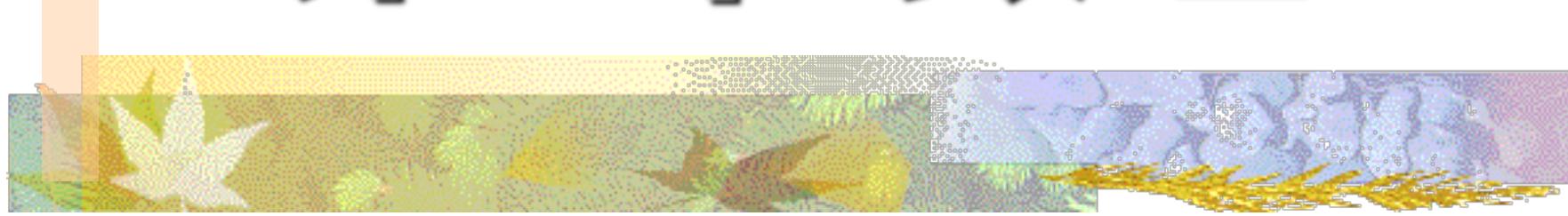




规格严格 功夫到家



第8章 数组



哈尔滨工业大学

计算机科学与技术学院

苏小红

sxh@hit.edu.cn

本章学习内容

- ☞ 对数组名特殊含义的理解
- ☞ 数组类型，数组的定义和初始化
- ☞ 向函数传递一维数组和二维数组
- ☞ 排序、查找、求最大最小值等常用算法

为什么使用数组(Array)?

【例8.1】要读入5人的成绩，然后求平均成绩

- 需定义5个不同名整型变量，需要使用多个scanf()

```
int score1, score2, score3, score4, score5;  
scanf("%d", &score1);  
scanf("%d", &score2);  
.....
```

- 而用数组，可共用一个scanf()并利用循环语句读取

```
int score[5], i;  
for (i=0; i<5; i++)  
{  
    scanf("%d", &score[i]);  
}
```



8.1 一维数组的定义和初始化

一维数组的定义

存储类型 数据类型 数组名 [整数1] [整数2] ... [整数n];

`int a[5];`

基类型

下标从0开始

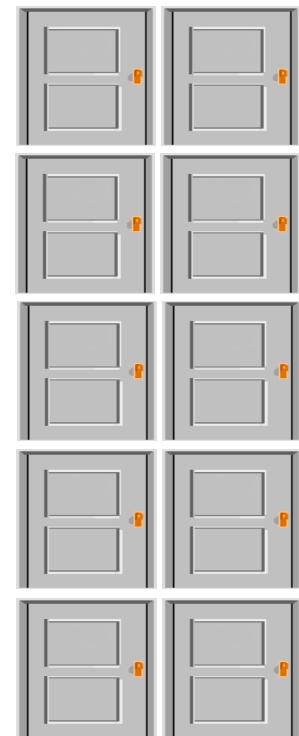
`a[0]`

`a[1]`

`a[2]`

`a[3]`

`a[4]`



- 定义一个有5个int型元素的数组
 - 系统在内存分配连续的5个int空间给此数组
- 直接对a的访问，就是访问此数组的首地址

8.1 一维数组的定义和初始化

一维数组的定义

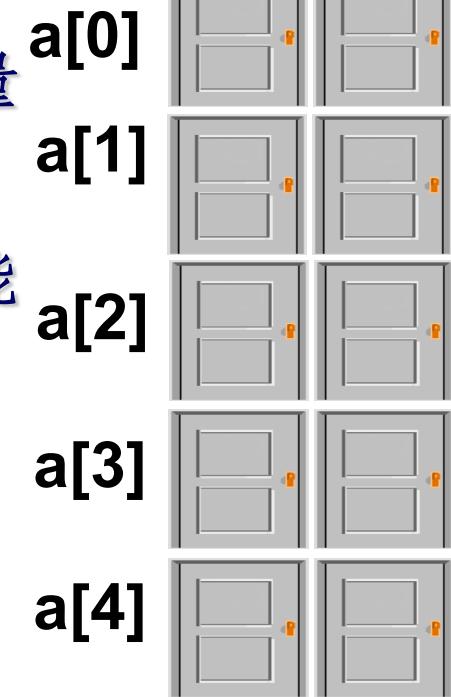
存储类型 数据类型 数组名 [整数1] [整数2] [整数n];

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

- 数组大小必须是值为正的常量，不能为变量
 - 一旦定义，不能改变大小
- 数组大小最好用宏来定义，以适应未来可能的变化

```
#define SIZE 5
```

```
int a[SIZE];
```



8.1 一维数组的定义和初始化

- 数组定义后的初值仍然是随机数
- 一般需要我们来初始化

```
int a[5] = { 12, 34, 56, 78, 9 };  
int a[5] = { 0 };  
int a[] = { 11, 22, 33, 44, 55 };
```

8.1 一维数组的定义和初始化

■ 数组的引用

数组名 [下标]

- 数组下标(index)都是从0开始
- 使用 **a[0]、a[1]、a[2]、a[3]、a[4]** 这样的形式访问每个元素
- 下标既可是常量，也可是整型表达式，允许快速随机访问，如 **a[i]**
 - 可以像使用普通变量一样使用它们

如何使两个数组的值相等?

```
main()
{
    int a[5] = {1,2,3,4,5}, b[5];
    b = a;
}
```



原因:
数组名表示数组的首地址,
其值不可改变!



解决方法

- **方法1:逐个元素赋值**

```
b[0]=a[0];
b[1]=a[1];
b[2]=a[2];
b[3]=a[3];
b[4]=a[4];
```

- **方法2:通过循环赋值**

```
int i;
for (i=0;i<5;i++)
{
    b[i] = a[i];
}
```

8.1 一维数组的定义和初始化

【例8.2】编程实现显示用户输入的月份（不包括闰年的月份）拥有的天数

```
1 #include <stdio.h>
2 #define MONTHS 12
3 int main()
4 {
5     int days[MONTHS] = {31,28,31,30,31,30,31,31,30,31,30,31};
6     int month;
7     do{
8         printf("Input a month:");
9         scanf("%d", &month);
10    }while(month < 1 || month > 12); /* 处理不合法数据的输入 */
11    printf("The number of days is %d\n", days[month-1]);
12    return 0;
13 }
```

8.1 一维数组的定义和初始化

■ 下标越界是大忌！

- 编译程序不检查是否越界
- 下标越界，将访问数组以外的空间
- 那里的数据是未知的，不受我们掌控，可能带来严重后果

【例8.3】当下标值小于0或超过数组长度时会出现什么情况？

```
#include <stdio.h>
int main()
{
    int a = 1, c = 2, b[5] = {0}, i;
    printf("%p, %p, %p\n", b, &c, &a);
    for (i=0; i<=8; i++)
    {
        b[i] = i;
        printf("%d ", b[i]);
    }
    printf("\nc=%d, a=%d, i=%d\n", c, a, i);
    return 0;
}
```

| | | |
|------|-----|----|
| b[0] | 0 | 40 |
| b[1] | 1 | 44 |
| b[2] | 2 | 48 |
| b[3] | 3 | 4c |
| b[4] | 4 | 50 |
| c | 5 | 54 |
| a | 6 | 58 |
| i | 9 | 5c |
| b[8] | 8 | 60 |
| | | 64 |
| | | 68 |
| | ... | 68 |
| | | 6c |

运行程序或单步执行观察变量变化情况可以看到，变量c和a的值因数组越界而被悄悄破坏了



8.2 二维数组的定义和初始化

一维数组

- 用一个下标确定各元素在数组中的顺序
- 可用排列成一行的元素组来表示
 - 如 `int a[5];`

二维数组

- 用两个下标确定各元素在数组中的顺序
- 可用排列成i行，j列的元素组来表示
 - 如 `int b[2][3];`

n维数组

- 用n个下标来确定各元素在数组中的顺序
 - 如 `int c[3][2][4];`
- n≥3时，维数组无法在平面上表示其各元素的位置

| | | | | |
|-------------|-------------|-------------|-------------|-------------|
| a[0] | a[1] | a[2] | a[3] | a[4] |
|-------------|-------------|-------------|-------------|-------------|

2024/3/20

| | | |
|----------------|----------------|----------------|
| b[0][0] | b[0][1] | b[0][2] |
| b[1][0] | b[1][1] | b[1][2] |

二维数组的初始化



【例】以下程序的运行结果是什么？

```
int main()
{
    int a[][][3]={{1,2,3},{4,5},{6},{0}};
    printf("%d,%d,%d\n",a[1][1],a[2][1],a[3][1]);
    return 0;
}
```

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 0 |
| 6 | 0 | 0 |
| 0 | 0 | 0 |

结果：5， 0， 0

【例】若 `int a[][3]={1, 2, 3, 4, 5, 6, 7, }`，
则 `a` 数组的第一维大小是多少？

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 0 | 0 |

数组的数据类型和存储类型

- 根据数组的**数据类型**，为每一元素安排相同长度的存储单元
- 根据数组的**存储类型**，将其安排在内存的动态存储区、静态存储区或寄存器区
- 用**sizeof(a)**来获得数组a所占字节数

`short a[10]`

| | |
|-------------------|-----|
| <code>a[0]</code> | 2字节 |
| <code>a[1]</code> | 2字节 |
| ... | ... |
| <code>a[9]</code> | 2字节 |

`char name[8]`

| | |
|----------------------|-----|
| <code>name[0]</code> | 1字节 |
| <code>name[1]</code> | 1字节 |
| ... | ... |
| <code>name[7]</code> | 1字节 |

动态存储区

`static float x[8]`

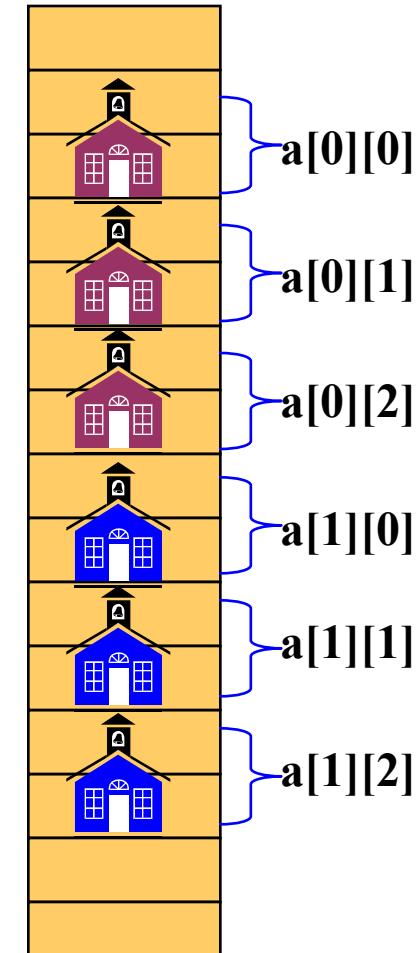
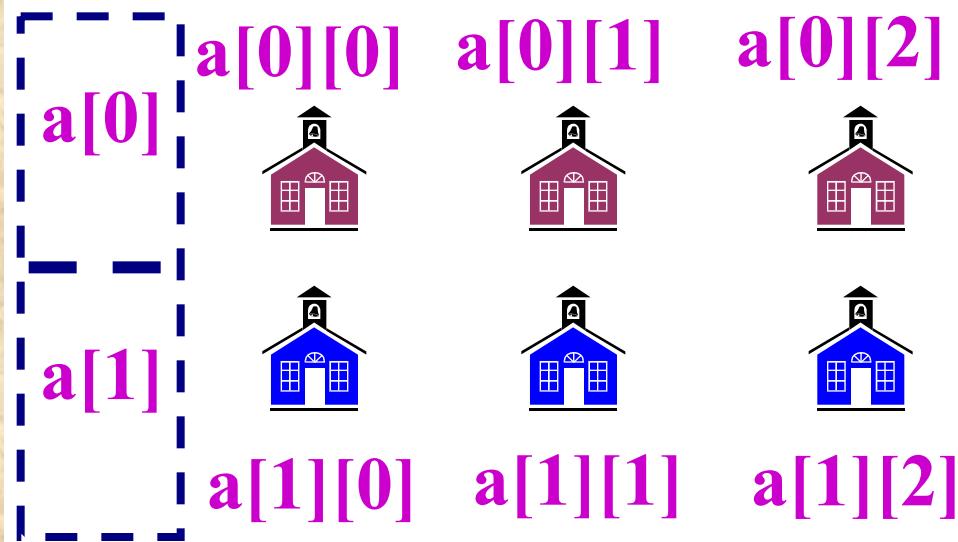
| | |
|-------------------|-----|
| <code>x[0]</code> | 4字节 |
| <code>x[1]</code> | 4字节 |
| ... | ... |
| <code>x[7]</code> | 4字节 |

静态存储区

二维数组的存储结构

存放顺序：按行存放

先顺序存放第0行元素，再存放第1行元素
short int a[2][3];



需知道数组每行列数才能从起始地址开始正确读出数组元素

二维数组实例

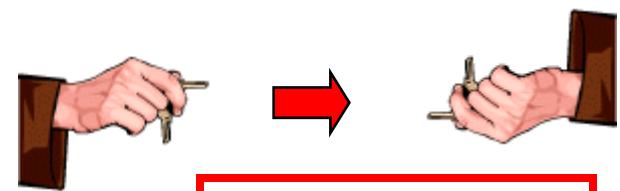
【例8.4】从键盘输入某年某月（包括闰年），编程输出该年的该月拥有的天数

```
1 #include <stdio.h>
2 #define MONTHS 12
3 int main()
4 {
5     int days[2][MONTHS] = {{31,28,31,30,31,30,31,31,30,31,30,31},
6                           {31,29,31,30,31,30,31,31,30,31,30,31}};
7     int year, month;
8     do{
9         printf("Input year,month:");
10        scanf("%d,%d", &year, &month);
11    } while(month < 1 || month > 12); /* 处理不合法数据的输入 */
12    if (((year%4 == 0) && (year%100 != 0)) || (year%400 == 0))/*闰年*/
13        printf("The number of days is %d\n", days[1][month-1]);
14    else /*非闰年*/
15        printf("The number of days is %d\n", days[0][month-1]);
16    return 0;
17 }
```

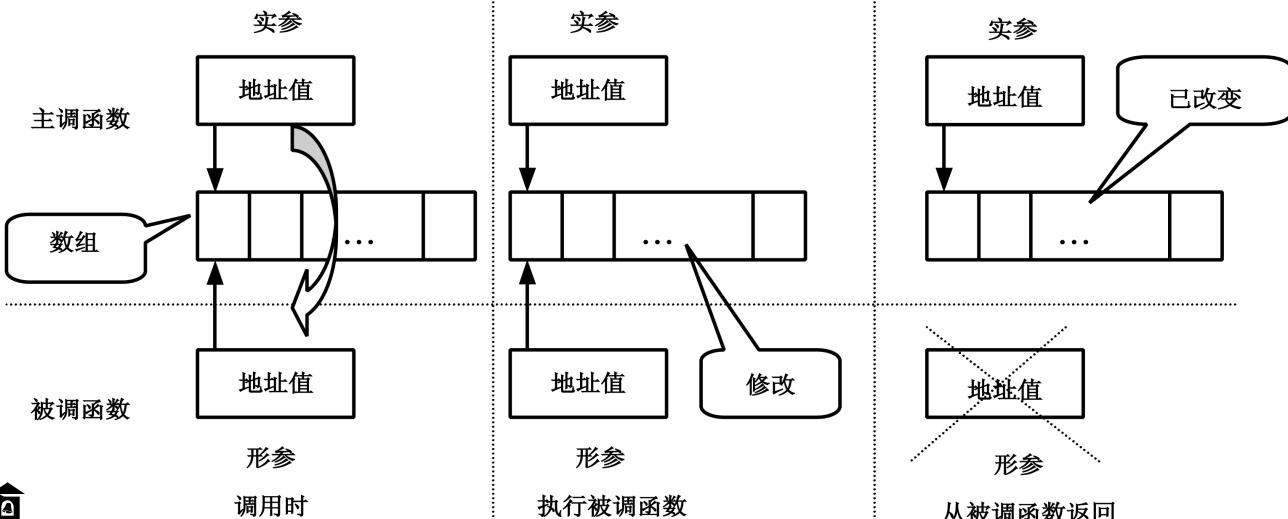
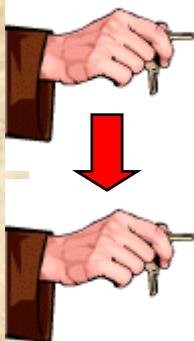
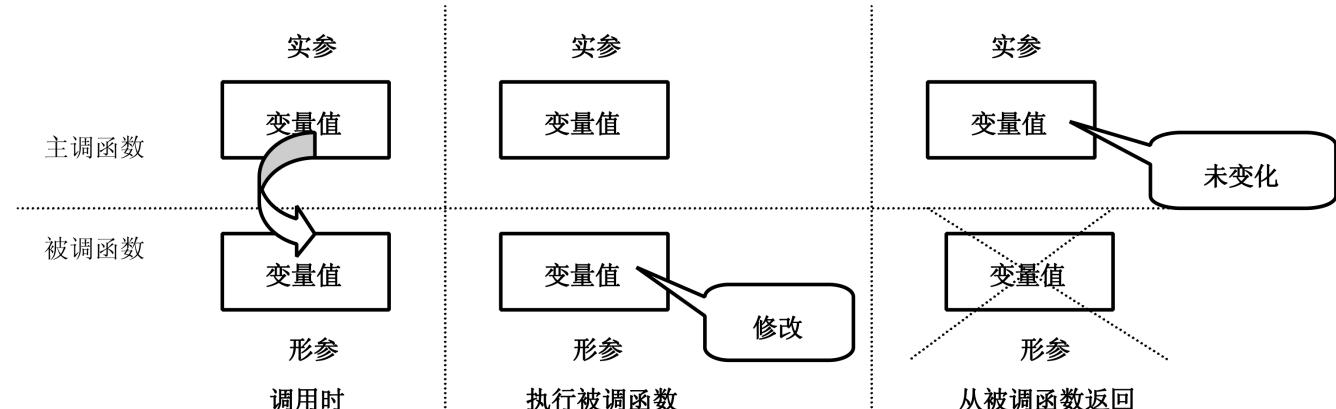
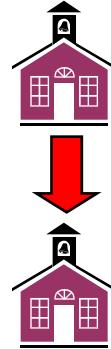
8.3 向函数传递一维数组

■ 传递整个数组给另一个函数，可将数组的首地址作为参数传过去

- 用数组名作函数参数
- 只复制一个地址自然比复制全部数据效率高
- 由于首地址相同，故实参数组与形参数组占用同一段内存
- 在该函数内，不仅可以读这个数组的元素，还可以修改它们



简单变量和数组作函数参数的区别



【例8.5】计算平均分

计数控制的循环

```
1 #include <stdio.h>
2 #define N 40
3 int Average(int score[], int n);
4 void ReadScore(int score[], int n);
5 int main()
6 {
7     int score[N], aver, n;
8     printf("Input n:");
9     scanf("%d", &n);
10    ReadScore(score, n);
11    aver = Average(score, n);
12    printf("Average score is %d\n", aver);
13    return 0;
14 }
```

【例8.5】计算平均分

计数控制的循环

```
15  /* 函数功能：计算n个学生成绩的平均分 */
16  int Average(int score[], int n)
17  {
18      int i, sum = 0;
19      for (i=0; i<n; i++)
20      {
21          sum += score[i];
22      }
23      return sum / n;
24  }
25  /* 函数功能：输入n个学生的某门课成绩 */
26  void ReadScore(int score[], int n)
27  {
28      int i;
29      printf("Input score:");
30      for (i=0; i<n; i++)
31      {
32          scanf("%d", &score[i]);
33      }
34 }
```

```
return n>0 ? sum/n : -1;  
更安全
```

【例8.6】计算平均分 当输入负值时，表示输入结束

■ 标记控制的循环——负值作为输入结束标记

```
1 #include <stdio.h>
2 #define N 40
3 int Average(int score[], int n);
4 int ReadScore(int score[]);
5 int main()
6 {
7     int score[N], aver, n;
8     n = ReadScore(score);
9     printf("Total students are %d\n", n);
10    aver = Average(score, n);
11    printf("Average score is %d\n", aver);
12    return 0;
13 }
```

【例8.6】计算平均分 当输入负值时，表示输入结束

■ 标记控制的循环——负值作为输入结束标记

```
25 int ReadScore(int score[])
26 {
27     int i = -1;
28     do {
29         i++;
30         printf("Input score:");
31         scanf("%d", &score[i]);
32     }while (score[i] >= 0);
33     return i;
34 }
```

【例8.7】计算最高分

```
#include <stdio.h>
#define N 40
int ReadScore(int score[]);
int FindMax(int score[], int n);
int main()
{
    int score[N], max, n;

    n = ReadScore(score);

    printf("Total students are %d\n", n);
    max = FindMax(score, n);

    printf("The highest score is %d\n", max);

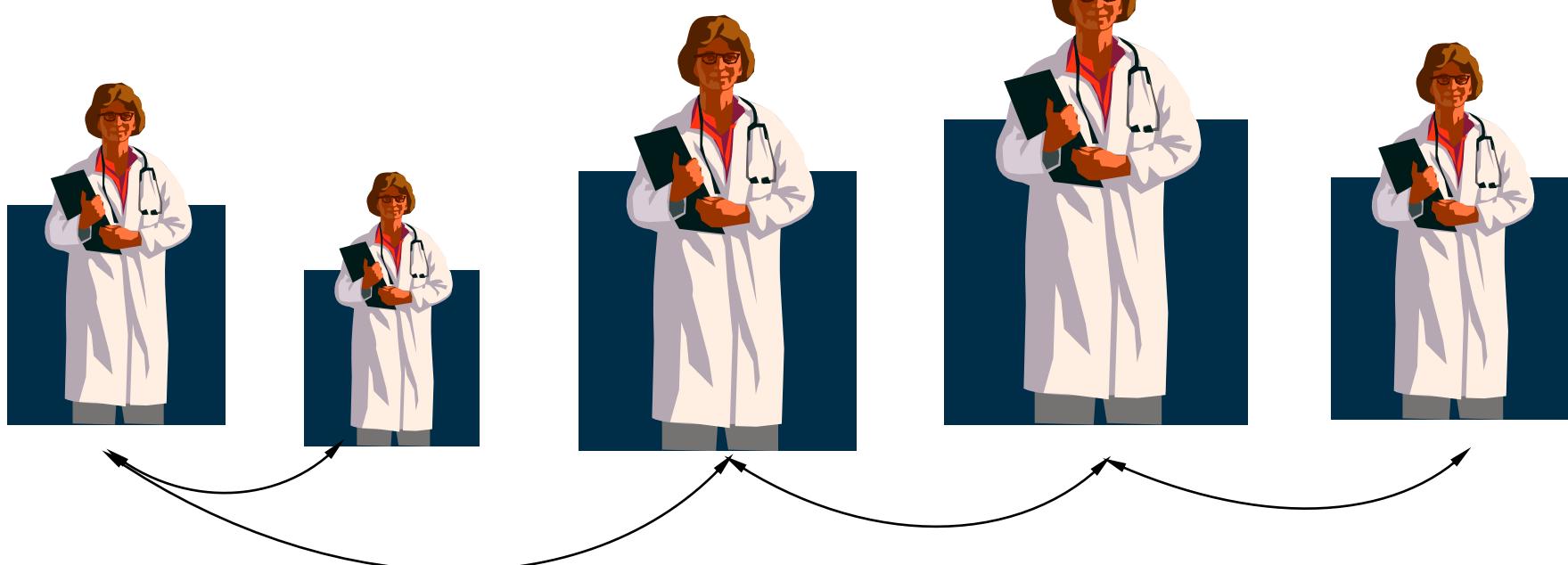
    return 0;
}
```

计算最大值算法

$\max(i=0)$

$\max(i=2)$

$\max(i=3)$



【例8.7】计算最高分

- 假设其中的一个学生成绩为最高

```
maxScore = score[0];
```

- 对所有学生成绩进行比较，即

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

```
{
```

```
    若 score[i] > maxScore
```

```
        则修改 maxScore 值为 score[i]
```

```
}
```

- 打印最高分 **maxScore**

【例8.7】计算最高分

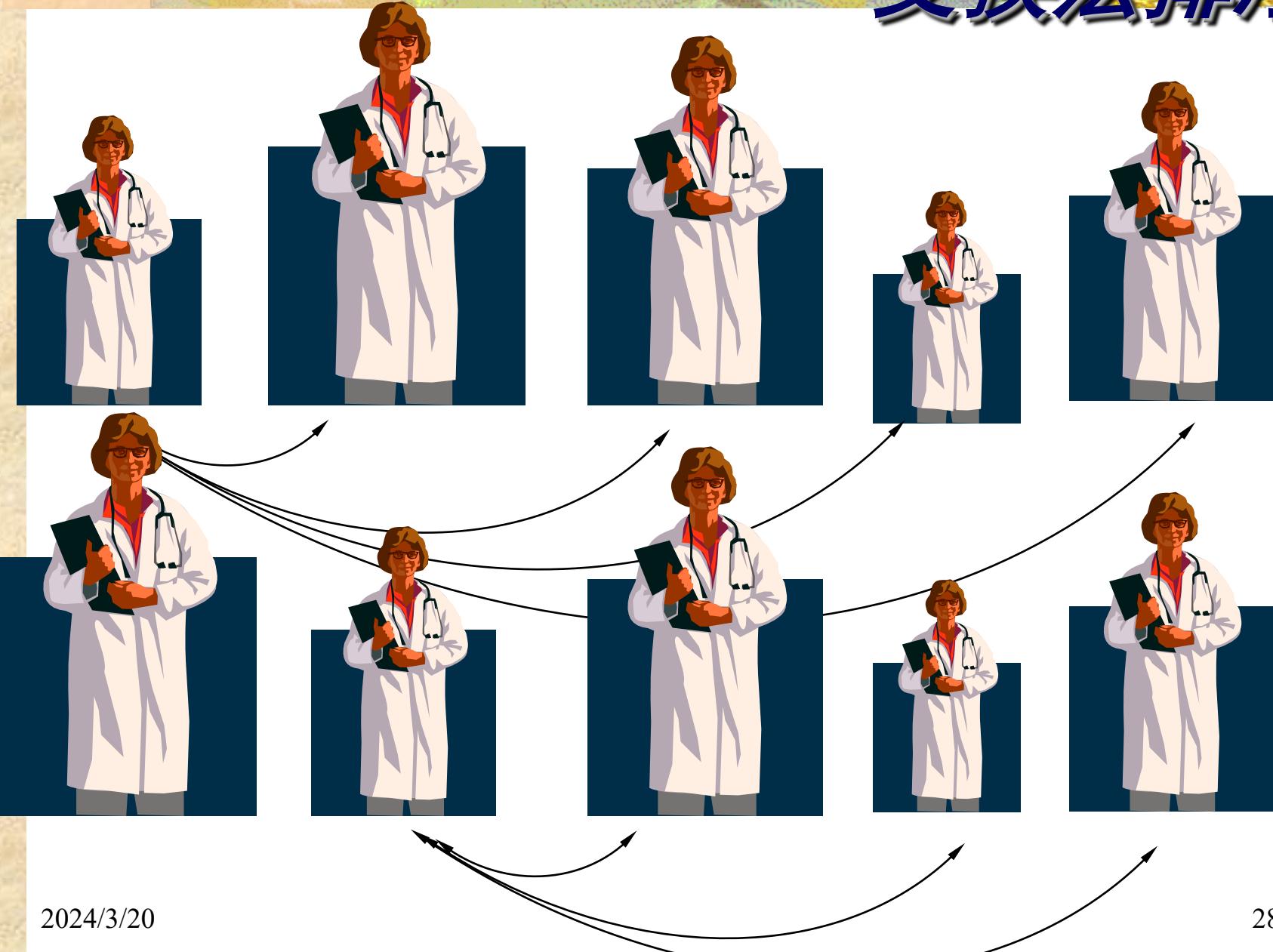
```
25  /* 函数功能：计算最高分 */
26  int FindMax(int score[], int n)
27  {
28      int max, i;
29      max = score[0];
30      for (i=1; i<n; i++)
31      {
32          if (score[i] > max)
33          {
34              max = score[i];
35          }
36      }
37      return max;
38 }
```

8.4 排序和查找

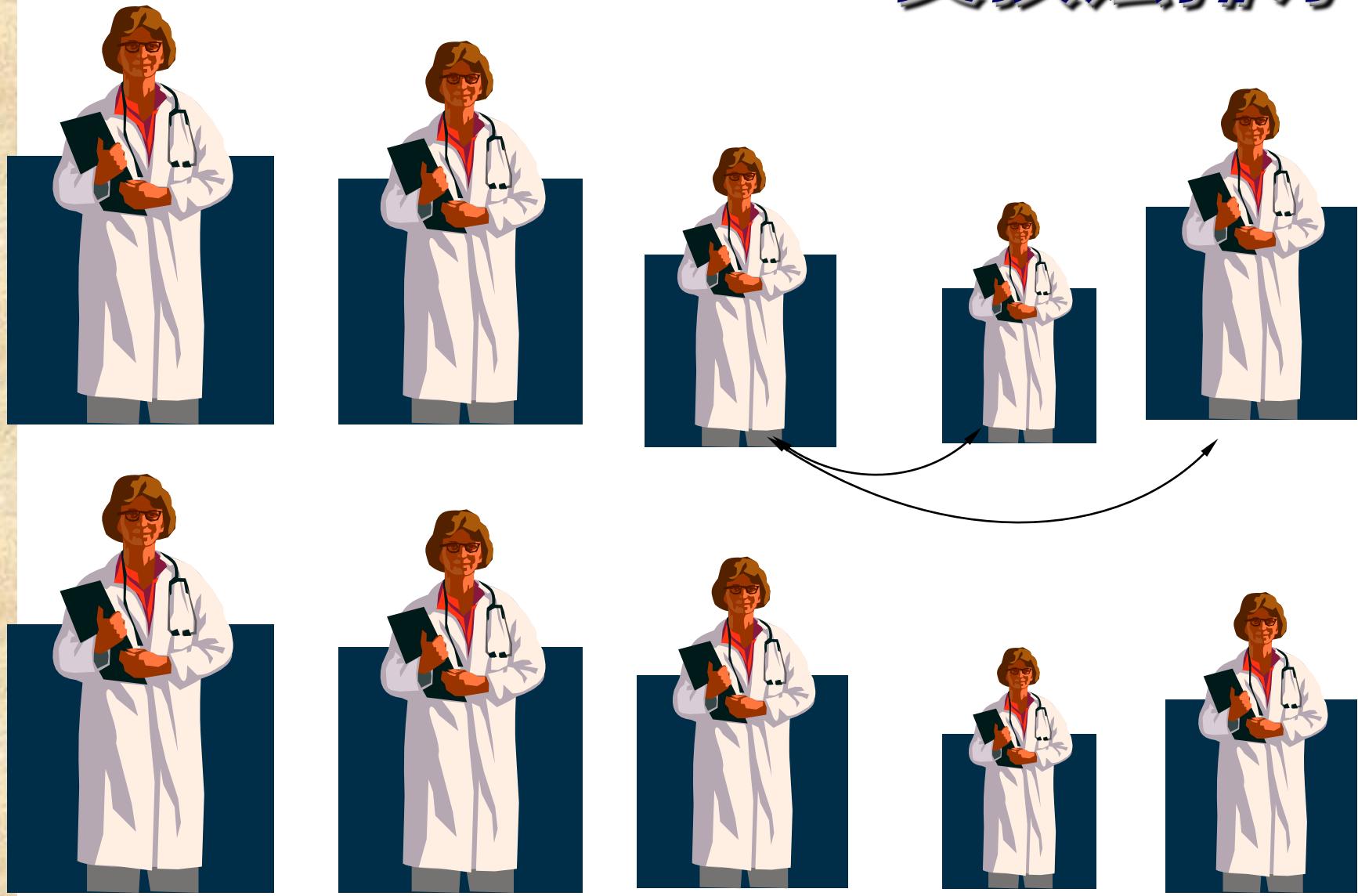
■ 排序 (Sorting) 算法

- 交换法排序
- 选择法排序

交換法排序



交換法排序



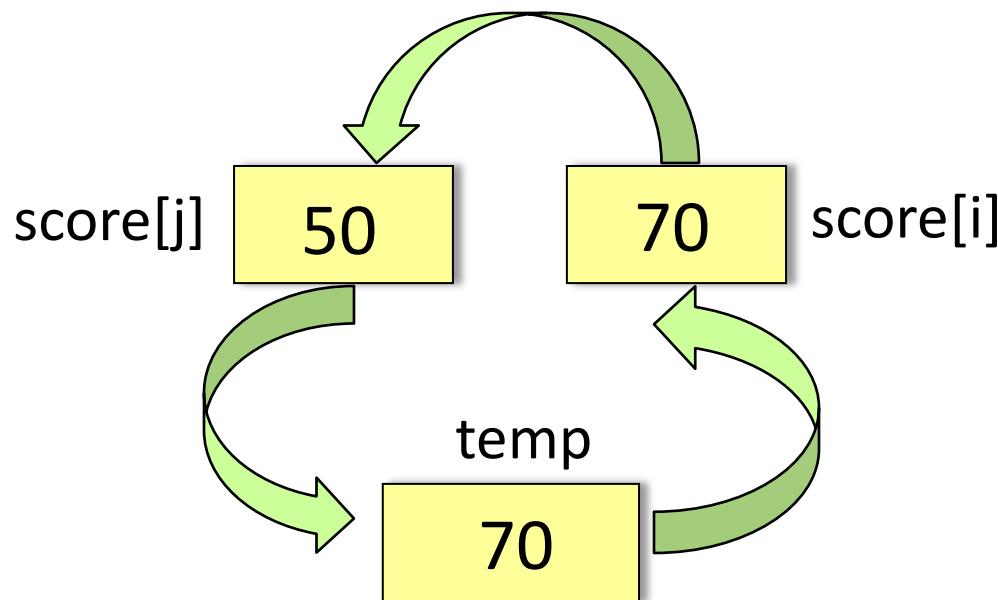
【例8.8】交换法从高到低排序

交换法排序

```
for (i=0; i<n-1; i++)
{
    for (j=i+1; j<n; j++)
        if (score[j] > score[i])
            "交换成绩score[j]和score[i]"
}
```

如何实现两数交换？

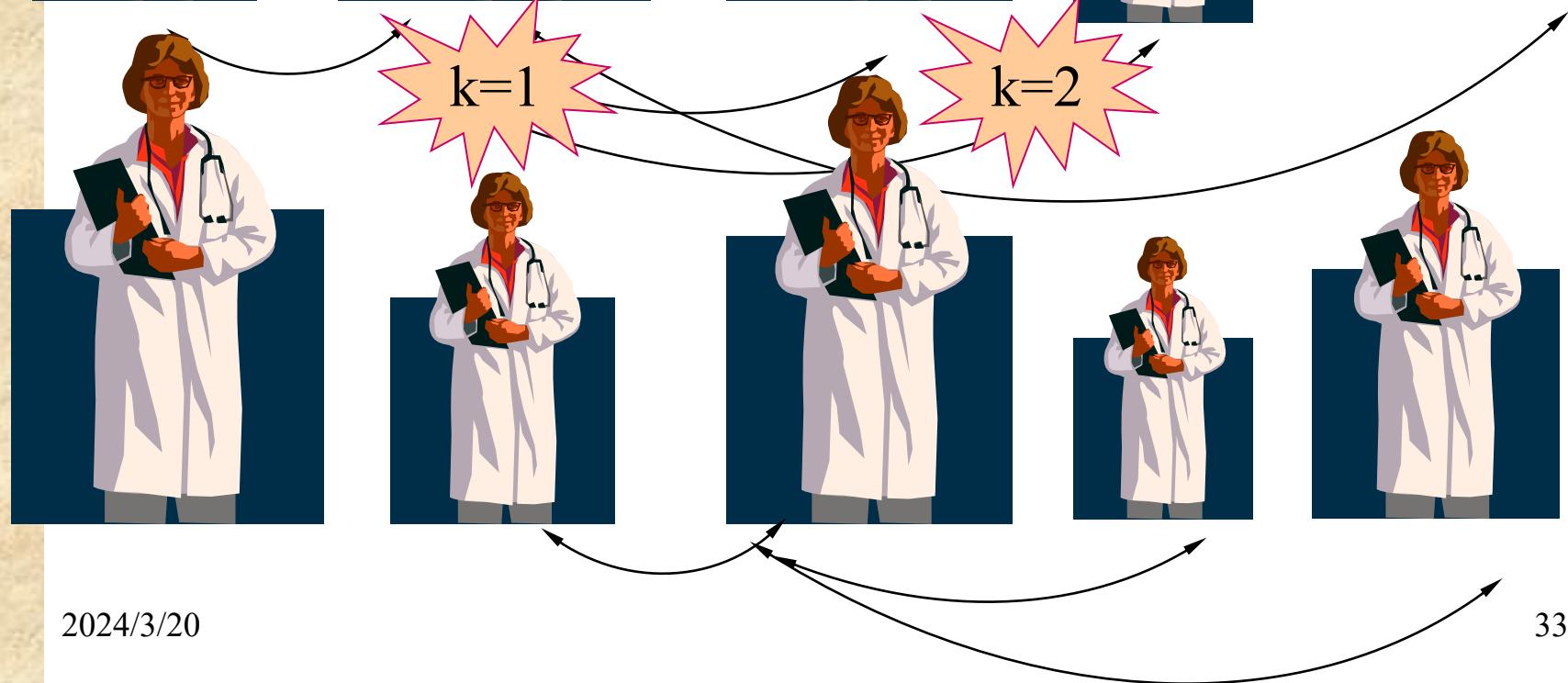
```
temp = score[j];  
score[j] = score[i];  
score[i] = temp;
```



【例8.8】交换法从高到低排序

```
void DataSort(int score[], int n) /*交换法排序*/
{
    int i, j, temp;
    for (i=0; i<n-1; i++)
    {
        for (j=i+1; j<n; j++)
        {
            if (score[j] > score[i]) /*从高到低*/
            {
                temp = score[j];
                score[j] = score[i];
                score[i] = temp;
            }
        }
    }
}
```

选择法排序



选择法排序



选择法排序

选择法排序

```
for (i=0; i<n-1; i++)
{
    k = i;
    for (j=i+1; j<n; j++)
    {
        if (score[j] > score[k])
            记录此轮比较中最高分的元素下标 k = j;
    }
}
```

若k中记录的最大数不在位置i，则

"交换成绩score[k]和score[i]",
"交换学号num[k]和num[i]";

```
void DataSort(int score[], long num[], int n) /*选择法*/
{
    int i, j, k, temp1;
    long temp2;
    for (i=0; i<n-1; i++)
    {
        k = i;
        for (j=i+1; j<n; j++)
        {
            if (score[j] > score[k])
            {
                k = j; /*记录最大数下标位置*/
            }
        }
        if (k != i) /*若最大数不在下标位置i*/
        {
            temp1 = score[k];
            score[k] = score[i];
            score[i] = temp1;
            temp2 = num[k];
            num[k] = num[i];
            num[i] = temp2;
        }
    }
}
```

【例8.8】成绩从高到低顺序

```
1 #include <stdio.h>
2 #define N 40
3 int ReadScore(int score[]);
4 void DataSort(int score[], int n);
5 void PrintScore(int score[], int n);
6 int main()
7 {
8     int score[N], n;
9     n = ReadScore(score);
10    printf("Total students are %d\n", n);
11    DataSort(score, n);
12    printf("Sorted scores:");
13    PrintScore(score, n);
14    return 0;
15 }
```

8.4 排序和查找

■ 查找 (Searching) 算法

- 顺序查找
- 折半查找

【例8.10】顺序查找学号

```
int LinSearch(long num[], long x, int n)
{
    int i;
    for (i=0; i<n; i++)
    {
        if (num[i] == x)
        {
            return i;
        }
    }
    return -1;
}
```

事先不必排序



哈，找到了！

【例8.11】折半查找学号

数组下标 0 1 2 3 4
第一次循环: 070310122 070310124 070310126 070310128 070310130

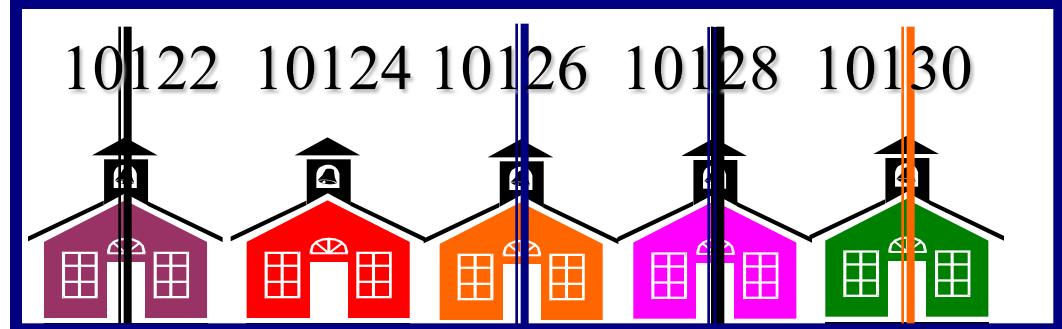
①查找值 x=070310128

low mid high x>num[mid], low=mid+1

第二次循环: 070310122 070310124 070310126 070310128 070310130

low(mid) high x=num[mid], 找到

按升序排序



2024/3/20

哈，找到了！

【例8.11】折半查找学号

第一次循环: 070310122 070310124 070310126 070310128 070310130 ②查找值 $x=070310127$

low mid high $x > \text{num}[\text{mid}]$, $\text{low} = \text{mid} + 1$

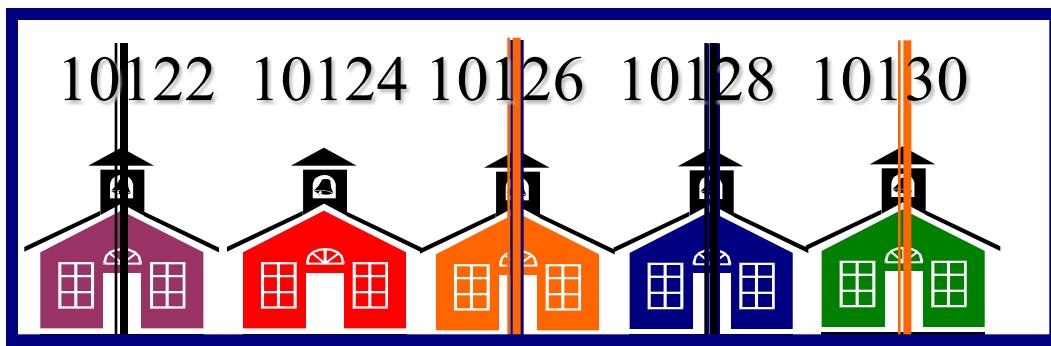
第二次循环: 070310122 070310124 070310126 070310128 070310130

 low(mid) high $x < \text{num}[\text{mid}]$, $\text{high} = \text{mid} - 1$

第三次循环: 070310122 070310124 070310126 070310128 070310130

 high low

不满足 $\text{low} \leq \text{high}$
循环结束, 未找到



```

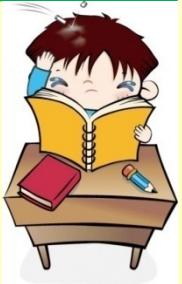
int BinSearch(long num[], long x, int n)
{
    int low, high, mid;
    low = 0;
    high = n - 1;
    while (low <= high)
    {
        mid = (high + low) / 2;
        if (x > num[mid])
        {
            low = mid + 1;
        }
        else if (x < num[mid])
        {
            high = mid - 1;
        }
        else
        {
            return mid;
        }
    }
    return -1;
}

```

若未按学号排序,
则如何修改程序?

找到时返回
下标位置

找不到时
返回-1



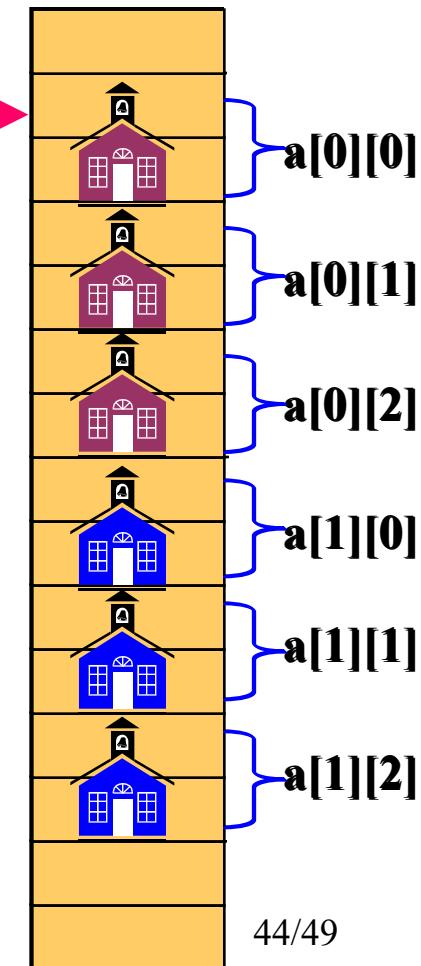
```
void DataSort(int score[], long num[], int n) /*选择法*/
{
    int i, j, k, temp1;
    long temp2;
    for (i=0; i<n-1; i++)
    {
        k = i;
        for (j=i+1; j<n; j++)
        {
            if (num[j] < num[k])
            {
                k = j; /*记录最大数下标位置*/
            }
        }
        if (k != i) /*若最大数不在下标位置i*/
        {
            temp1 = score[k];
            score[k] = score[i];
            score[i] = temp1;
            temp2 = num[k];
            num[k] = num[i];
            num[i] = temp2;
        }
    }
}
```

按学号由小
到大排序

8.5 向函数传递二维数组

```
short a[2][3];
```

实际传送的是数组第一个元素的地址



8.5 向函数传递二维数组



`short a[M][N];`

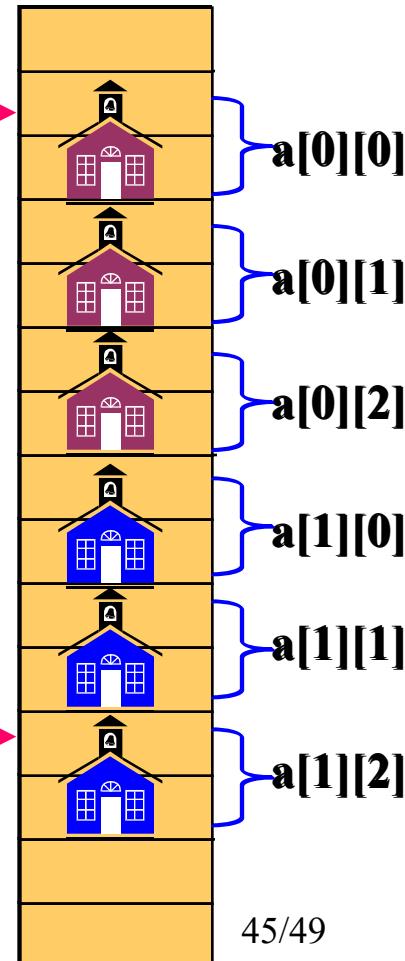
实际传送的是数组第一个元素的地址

- 在声明二维数组形参时，不能省略数组第二维的长度（列数），为什么？
- 想想数组在内存中是如何分布的？
- 元素`a[i][j]`在数组`a`中的位置是：

$$i * N + j$$

元素地址：首地址 + 偏移量

偏移 $1 * 3 + 2$



例8.12 计算每门课程的总分和平均分

```
void AverforCourse(int score[][][COURSE_N], int sum[],  
                    float aver[], int n)  
{  
    int i, j;  
  
    for (j=0; j<COURSE_N; j++)  
    {  
        sum[j] = 0;  
        for (i=0; i<n; i++)  
        {  
            sum[j] = sum[j] + score[i][j];  
        }  
        aver[j] = (float) sum[j] / n;  
    }  
}
```

可省略数组第一维的长度
不能省略第二维的长度



例8.12 计算每门学生的总分和平均分

```
void AverforStud(int score[][][COURSE_N], int sum[],
                  float aver[], int n)
{
    int i, j;

    for (i=0; i<n; i++)
    {
        sum[i] = 0;
        for (j=0; j<COURSE_N; j++)
        {
            sum[i] = sum[i] + score[i][j];
        }
        aver[i] = (float) sum[i] / COURSE_N;
    }
}
```



例8.12计算每门的总分和平均分

```
#include <stdio.h>

#define STUD_N 40           /* 最多学生人数 */
#define COURSE_N 3          /* 考试科目数 */

void ReadScore(int score[][COURSE_N], long num[], int n);
void AverforStud(int score[][COURSE_N], int sum[], float aver[], int n);
void AverforCourse(int score[][COURSE_N], int sum[], float aver[], int n);
void Print(int score[][COURSE_N], long num[], int sumS[], float averS[],
           int sumC[], float averC[], int n);

int main()
{
    int score[STUD_N][COURSE_N], sumS[STUD_N], sumC[STUD_N], n;
    long num[STUD_N];
    float averS[STUD_N], averC[STUD_N];
    printf("Input the total number of the students(n<=40):");
    scanf("%d", &n);           /* 输入参加考试的学生人数 */
    ReadScore(score, num, n);   /* 读入n个学生的学号和成绩 */
    AverforStud(score, sumS, averS, n); /* 计算每个学生的总分平均分 */
    AverforCourse(score, sumC, averC, n); /* 计算每门课程的总分平均分 */
    Print(score, num, sumS, averS, sumC, averC, n); /* 输出学生成绩 */
    return 0;
}
```



■ Questions and answers

