# 第5章

# 数　组

CHAPTER 5

## 本章导读

学习了C语言的整型、实型、字符型这些基本数据类型和程序流程控制结构后，很多问题都可以描述和解决了。但对于大规模的数据，尤其是相互间具有一定联系的数据，怎么才能高效表示和组织呢？C语言的数组类型为这些数据的组织提供了一种有效的形式。

本章介绍在C语言中怎样定义和使用数组，包括一维数组、二维数组、字符数组、结构体数组，并且采用结构体数组实现“学生成绩管理系统”案例的数据输入、浏览、成绩统计和成绩排序功能。

## 学习目标

- 掌握一维数组的定义和使用。
- 掌握二维数组的定义和使用。
- 掌握字符数组的定义和使用。
- 掌握字符串的处理方法，熟悉字符串的常用处理函数。
- 掌握结构体类型的定义，结构体变量的定义和使用。
- 掌握结构体数组的定义和使用。
- 熟悉数组基本操作算法，能够正确使用数组解决一般应用问题。

# 5.1　一维数组

## 5.1.1　一维数组的定义和引用

### 1. 一维数组的定义

在 C 语言中使用数组必须先进行定义。一维数组的定义格式为：

```
类型声明符 数组名[数组大小];
```

其中：

(1) 类型声明符是任一种基本数据类型或构造数据类型，即 int、float、double、char 等这些基本数据类型，也可以是 5.4 节介绍的结构体类型。从这里可以看出，数组是创建在其他数据类型基础之上的，因此数组是构造类型。

(2) 数组名是用户定义的数组标识符，其命名规则遵循标识符命名规则。对于数组元素来说，具有一个共同的名字，即数组名。

(3) 方括号中的数组大小表示数组元素的个数，也称为数组的长度。例如：

```
int a[5];                    //定义整型数组 a,有 5 个元素
float b[10],b[3 * 5];        //定义两个实型数组 b 和 c,分别有 10 和 15 个元素
char ch[20];                 //定义字符型数组 ch,有 20 个元素
```

对于数组定义，应注意如下几点。

(1) 数组的类型实际上是指数组元素的取值类型。对于同一个数组，其所有元素的数据类型都是相同的。

(2) 在 C89 标准中，数组的大小必须在编译时确定，这意味着数组的长度必须是一个常量表达式。但是，从 C99 标准开始，引入了变长数组，允许使用变量来指定数组的长度。

(3) 数组名不能与同一作用域范围内的其他变量名相同。例如：

```
int main()
{
    int a;
    int a[5];                //错误,数组名与变量名相同
    …
    return 0;
}
```

(4) 数组元素的下标从 0 开始，并且所有元素在内存中连续存储。因此，数组 a 的 5 个元素依次为 a[0]、a[1]、a[2]、a[3]、a[4]。这些数组元素在内存中的存储形式如图 5.1 所示。

数组名 a 是数组存储区的首地址，即存储数组第一个元素的地址，也就是 a 等价于 &a[0]，因此数组名是一个地址常量，不能对数组名进行赋值和进行运算。


图 5.1　一维数组在内存中的存储形式

### 2. 一维数组元素的引用

数组定义后,可以在程序中引用数组中的元素,如给数组元素赋值,从键盘输入数据存储到数组元素中,输出数组元素的值等。一维数组元素的引用形式如下:

```
数组名[下标]
```

其中:

(1) 下标可以是整型常量或整型常量表达式,如 a[3]、a[3+2]。

(2) 下标也可以是整型变量或整型变量表达式,如 a[i]、a[i+j]、a[i++]。

(3) 下标如果是表达式,首先计算表达式,计算的最终结果为下标值。

(4) 下标值从 0 开始,而不是从 1 开始。

(5) 下标值不能越限,例如,对于语句"int a[5];"定义的数组,引用时的下标不能超过或等于 5,也就是说,"a[5]=10;"是错误的。

(6) 每个数组元素相当于一个普通变量,因此,访问数组元素的方法与普通变量相同。

**注意:**

对于整型或实型数组,只能逐个引用数组元素,不能一次引用整个数组。

例如,对于语句"int a[5];"定义的数组,要从键盘输入数组 a 的值,采用如下语句是错误的:

```
scanf("%d",&a);
```

正确的方法是通过循环语句,逐个输入数组元素的值,程序代码如下:

```
for(i=0;i<5:i++)
    scanf"%d",&a[i]);
```

**【例 5.1】** 从键盘输入 10 个整数,求其中的最大数并输出。

**【问题分析】**

(1) 定义一个一维数组,数组长度为 10,用于存储从键盘输入的 10 个整数。

(2) 求一组数的最大数:首先假定第一个数最大,把该数存储到变量 max 中,然后依次处理后面 9 个数,如果当前正在处理的这个数比 max 还大,则更改 max 的值为后面这个数。

将数组与 for 循环相结合，可以轻松完成此任务。

解决该问题的算法流程图如图 5.2 所示。

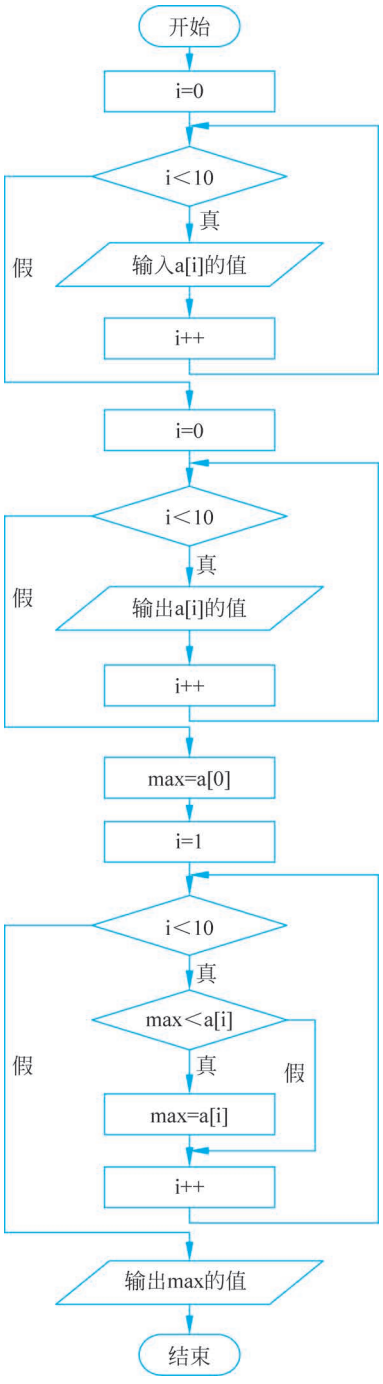


图 5.2　例 5.1 的算法流程图

【程序代码】

```
#include <stdio.h>
int main()
{
    int a[10];                                  //定义整型数组 a
    int i,max;                                  //定义循环控制变量 i 和存储最大数的变量 max
    printf("Please enter ten integers:\n");     //输出屏幕提示语
    for(i=0;i<10;i++)                           //循环 10 次
        scanf("%d",&a[i]);                      //从键盘接收数据,并存储到数组元素 a[i]中
    for(i=0;i<10;i++)                           //循环 10 次
        printf("%d  ",a[i]);                    //输出数组元素 a[i]的值
    max=a[0];                                   //给 max 变量赋值,假定第一个数最大
    for(i=1;i<10;i++)                           //循环 9 次
        if(max<a[i])                            //比较 max 与数组中当前正在处理的数组元素的
                                                //大小,将较大者赋给 max
            max=a[i];
    printf("\nThe max is %d\n",max);            //输出求得的最大数
    return 0;
}
```

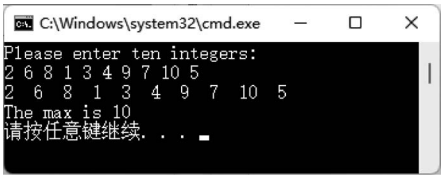


图 5.3　例 5.1 程序输出结果

【输出结果】

程序输出结果如图 5.3 所示。

**说明**：此例只是为了演示一维数组的定义与引用。在实际问题中，如果只是要求找出一组数中的最大或最小数，那么可以不用保存这组数据，而是采用边读边求最大或最小数的方式，这样的程序更简洁、高效。

## 5.1.2　一维数组的初始化

与一般变量的初始化一样，数组的初始化就是在定义数组的同时，给其数组元素赋初值。一维数组初始化赋值的一般形式如下：

```
类型声明符 数组名[数组大小]={数值 1,数值 2,……,数值 n};
```

其中赋值号右边大括号中的各数值即为各数组元素的初值，各数值之间用逗号分隔。例如：

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

相当于

```
a[0]=0;a[1]=1;a[2]=2;
```

数组初始化是在编译阶段进行的，这样可以减少运行时间，提高效率。C 语言对数组的初始化有如下几点规定。

(1) 数组在定义时如果不进行初始化,则数组中各元素的初值是随机的。

(2) 当大括号中的初值个数小于数组长度时,将只给前面的数组元素赋初值,后面的元素自动赋默认值,对于数值型(整型、实型)数组其默认值为 0。例如:

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

相当于给 a[0]赋初值 5,给 a[1]赋初值 6,后面 3 个元素自动赋 0 值。

(3) 只能给数组元素逐个赋值,不能给数组整体赋值。如下语句是正确的:

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

不能写为:

```
int a[5] = 1;
```

(4) 如果给出了全部数组元素的初值,则定义数组时可以省略数组大小。例如:

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

可写为:

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

(5) 如果大括号中的数值个数多于数组元素个数,将出现语法错误。

### 5.1.3　一维数组应用举例

【例 5.2】　任意给定 n 个数,按由小到大对其排序,并输出排序结果。

【问题分析】

这是一组数的排序问题,排序方法有多种,这里采用冒泡排序法。

冒泡排序法的思路:将相邻两个数比较,把较小的数调到前面(或把较大的数调到后面)。若有 5 个数,分别是 7、6、10、4、2,依次将其存储到数组 a 中。冒泡排序法的处理过程如下。

(1) 第一趟(如图 5.4 所示),经过 4 次比较。

第 1 次:将第 1 个数 a[0]和第 2 个数 a[1]进行比较,把较小的数调到前面。也就是说,若后面的数较小,就将两数交换,否则不交换。这里是把 7 和 6 对调位置,结果如图 5.4(b)所示。

第 2 次:将第 2 个数 a[1]和第 3 个数 a[2]进行比较,把较小的调到前面。这里是 7 和 10 比较,这次比较不用对调这两个元素的位置,结果如图 5.4(c)所示。

第 3 次:将第 3 个数 a[2]和第 4 个数 a[3]进行比较,把较小的调到前面。这里是 10 和 4 比较后对调位置,结果如图 5.4(d)所示。

第 4 次:将第 4 个数 a[3]和第 5 个数 a[4]进行比较,把较小的调到前面。这里是 10 和 2 比较后对调位置,结果如图 5.4(e)所示。

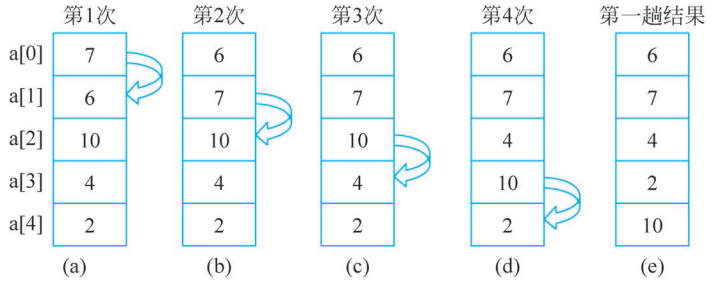


图 5.4　冒泡排序的第一趟处理过程

此时得到 6-7-4-2-10 的顺序,即最大的数 10 成为最下面的一个数。可见,较大的数向下“沉”,最大的数“沉底”,较小的数向上“浮”。

这 4 次处理过程都是类似的,都是“相邻两数比较,若后面的数较小,则两数交换,否则不交换”;所不同的是“比较的两个数,它们的位置不同”,先是 a[0]和 a[1]比较,再是 a[1]和 a[2]比较,然后是 a[2]和 a[3]比较,最后是 a[3]和 a[4]比较。我们会发现一个规律,每次比较完后,位置往后移了一位,所以可以用一个变量 i 控制,每次都是 a[i]和 a[i+1]比较,而每次比较完后 i+1,总共比较 4 次,这样就可以用一个循环来实现,即:

```
for(i = 0;i < 4;i++)
    if(a[i]> a[i + 1])
    {
        temp = a[i]; a[i] = a[i + 1]; a[i + 1] = temp;
    }
```

(2) 第二趟(如图 5.5 所示),经过 3 次比较。

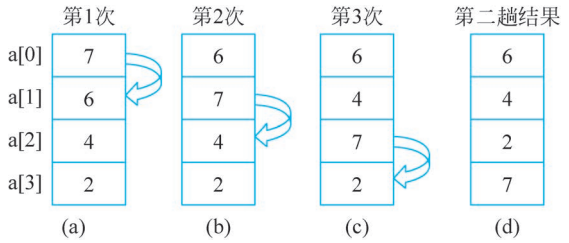


图 5.5　冒泡排序的第二趟处理过程

经过第一趟后最大数 10 已经沉到底了,第二趟就对余下的 4 个数(6、7、4、2)按上述的方法,经过 3 次比较,得到次大的数 7“沉底”,较小的数 4 和 2 向上“浮”,结果得到 6-4-2-7 的顺序。

这趟比较的代码与第一趟几乎一样,不一样的是比较的次数,也就是循环次数不一样,这趟循环 3 次。用循环语句实现如下:

```
for(i = 0;i < 3;i++)
    if(a[i]> a[i + 1])
    {
        temp = a[i]; a[i] = a[i + 1]; a[i + 1] = temp;
    }
```

(3) 第三趟(如图 5.6 所示),经过 2 次比较。

对余下的 3 个数(6、4、2)按上述方法,经过 2 次比较,得到第三大数 6“沉底”,较小的数 4 和 2 上“浮”。

这趟比较代码同上类似,只是循环次数变成了 2。

(4) 第四趟(如图 5.7 所示),经过 1 次比较。

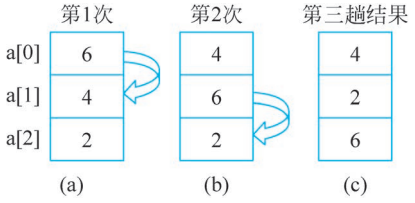


图 5.6　冒泡排序的第三趟处理过程

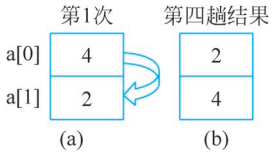


图 5.7　冒泡排序的第四趟处理过程

对余下的两个数(4 和 2)按上述方法,经过 1 次比较,得到第四大数 4“沉底”,最小的数 2 上“浮”。

最后得到 5 个数的排序结果:2-4-6-7-10(从小到大)。

从上面的 4 趟处理过程中可以看出,每一趟都很类似,都是“最大的数位置下沉,较小数向上浮起一个位置”;所不同的是“每趟比较的次数不同”,第一趟 4 次,第二趟 3 次,第三趟 2 次,第四趟 1 次,所以引入“趟次”循环变量 j,一共需要 4 趟,故 j 从 1 到 4,而每趟比较的次数都是 5−j。因此,可以用两个 for 循环来实现,“趟次”循环变量 j 作为外层循环控制变量,每趟里面的次数作为内循环,实现代码如下:

```
for(j = 1;j < = 4;j++)              //j 是趟次循环变量(外循环变量)
    for(i = 0;i < 5 - j;i++)        //i 是每趟中两两比较的次数变量(内循环变量)
        if(a[i]> a[i + 1])          //比较相邻两数大小,将较小的数放在前面
        {
            temp = a[i]; a[i] = a[i + 1]; a[i + 1] = temp;
        }
```

也就是说,“冒泡”排序最重要的是确定趟数和每趟比较的次数。我们来进行分析:其一,需要比较的趟数——5 个数需要冒 4 个泡,即要比较 4 趟,所以 n 个数要比较(n−1)趟。其二,每趟比较的次数——5 个数排序,第一趟比较 4 次,第二趟比较 3 次,第三趟比较 2 次,第四趟比较 1 次,得出规律“n 个数排序,第 j 趟要比较(n−j)次”。

综上所述,n 个数需要进行(n−1)趟比较,在第 j 趟的比较中要进行(n−j)次两两比较,任意 n 个数进行排序的程序如下。

**【程序代码】**

```
#include < stdio.h >
#define N 10                                  //定义符号常量,对几个数排序,N 的值就是几
int main()
{
    int a[N];                                 //定义数组
    int i,j,temp;                             //定义变量
```

```
    printf("Please enter ten integers:\n");    //输出提示语
    for(i = 0;i < N;i++)                       //从键盘接收 N 个数据存储到数组 a 中
        scanf(" % d",&a[i]);
    printf("\n");                              //输出换行符
    for(j = 1;j < N;j++)                       //j 是趟次循环变量(外循环变量)
        for(i = 0;i < N - j;i++)               //i 是每趟中两两比较的次数变量(内循环变量)
           if(a[i]> a[i + 1])                  //比较相邻两数大小,将较小的数放在前面
           {    temp  = a[i]; a[i] = a[i + 1]; a[i + 1] = temp;    }
    printf("The sorted numbers:\n");           //输出提示语
    for(i = 0;i < N;i++)                       //将排序好的数组输出
         printf(" % d  ",a[i]);
    printf("\n");
    return 0;
}
```

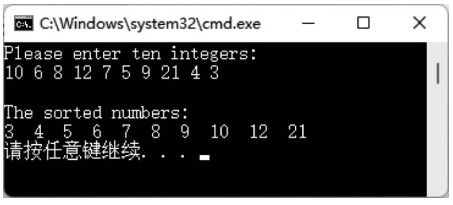


图 5.8　例 5.2 程序输出结果

**【输出结果】**

程序输出结果如图 5.8 所示。

**思考**：上述冒泡排序算法如何优化？

实际上，针对某些待排序的数据序列，有可能当排序进行到某一趟时，全部数据就已经有序了，但上述算法仍然会继续后面的排序操作，这显然是多余的。如何对上述算法进行改进呢？

## 5.2　二维数组

### 5.2.1　二维数组的定义和引用

前面介绍的数组只有一个下标，称为一维数组，其数组元素也称为单下标变量。在实际问题中有很多量是二维的或多维的，比如最常见的矩阵就是二维的，因此 C 语言允许构造二维或多维数组。多维数组元素有多个下标，以标识它在数组中的位置，所以也称为多下标变量。本节只介绍二维数组，多维数组可由二维数组类推而得到。

#### 1. 二维数组的定义

二维数组定义的一般形式如下：

```
类型声明符 数组名[行数][列数];
```

**说明**：

(1) 类型声明符、数组名的声明同一维数组的声明。

(2) 行数和列数为整型常量或整型常量表达式。

(3) 数组元素个数为：行数×列数。

(4) 数组元素的下标值从 0 开始。

例如：

```
int x[2][3];
```

x 是二维数组名，这个二维数组共有 6 个元素，它们是 x[0][0]、x[0][1]、x[0][2]、x[1][0]、x[1][1]、x[1][2]，且其全部元素数值均为整型。

### 2. 二维数组的存储

二维数组在概念上是二维的，比如说矩阵，但存储器单元是按一维线性排列的。在一维存储器中存储二维数组可有两种方式：一种是按行排列，即存储完一行之后顺次存储第二行。另一种是按列排列，即存储完一列之后再顺次存储第二列。在 C 语言中，二维数组是按行排列的。例如：

```
int x[2][3];
```

先存储第一行，即 x[0][0]、x[0][1]、x[0][2]，再存储第二行，即 x[1][0]、x[1][1]、x[1][2]，如图 5.9 所示。

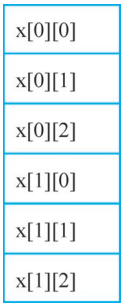


图 5.9　二维数组的存储

### 3. 二维数组的引用

二维数组元素的引用形式如下：

```
数组名[行下标][列下标]
```

**说明：**

(1) 行下标和列下标可以是常量(大于或等于 0)、常量表达式、变量或变量表达式。

(2) 数组中要特别注意下标越界。因为 C 语言编译系统不检查数组下标越界问题，所以程序设计者应特别注意。

(3) 二维数组的每个元素相当于一个普通变量，其使用方式与普通变量相同。

例如：

```
int a[3][4];
a[0][1] = 3;                    //直接给数组元素 a[0][1]赋值
scanf("%d",&a[0][1]);           //从键盘输入数据赋给数组元素 a[0][1]
printf("%d",a[0][1]);           //输出数组元素 a[0][1]的值
```

(4) 同一维数组一样，不能对一个二维数组的整体进行引用，只能对具体的数组元素进行引用。

## 5.2.2　二维数组的初始化

二维数组的初始化方法有分行赋初值和顺序赋初值两种。

### 1. 分行赋初值

例如：

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

在初始化后,数组 a 为:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

分行赋初值比较直观,每个花括号对应一行。也可以只对二维数组的部分元素赋初值。例如:

```
int b[3][3] = {{1,2,3},{},{7,8}};
```

上述语句只对数组 b 第 1 行的全部元素和第 3 行的前 2 个元素赋了初值,其他元素的初值为默认值(0)。在初始化后,数组 b 为:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 7 & 8 & 0 \end{bmatrix}$$

### 2. 顺序赋初值

把初始化值括在一对大括号内,系统将按数组元素在内存中的排列顺序依次对各元素赋初值,例如:

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

初始化结果是 x[0][0]=1,x[0][1]=2,x[0][2]=3,x[1][0]=4,x[1][1]=5,x[1][2]=6。

在定义二维数组时,如果进行了初始化,则可以省略第一维的长度,系统会自动根据初值的个数推算出第一维的大小,但第二维的长度不能省略。例如:

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

该数组每行有 3 列,这样可以推算出它一定有 3 行,所以该语句等价于:

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

再如:

```
int b[][3] = {{1,2,3},{},{7,8}};
```

该数组分 3 行赋初值,这样它一定有 3 行,所以该语句等价于:

```
int b[3][3] = {{1,2,3},{},{7,8}};
```

### 5.2.3　二维数组应用举例

【例 5.3】　编写一个程序实现 3×4 的矩阵的转置。矩阵转置是把矩阵的行和列互换，例如：

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

转置后变成 4×3 的矩阵：

$$\begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}$$

【问题分析】

(1) 定义两个二维数组 a 和 b，二维数组 a 为 3 行 4 列，用于存储转置前的 3×4 的矩阵，二维数组 b 为 4 行 3 列，用于存储转置后的 4×3 的矩阵。

(2) 使用双重 for 循环，将数组 a 中的元素 a[i][j]赋值给 b[j][i]，即可完成矩阵转置。

【程序代码】

```
#include <stdio.h>
int main()
{
    int a[3][4],b[4][3];                    //定义二维数组 a[3][4]和 b[4][3]
    int i,j;                                //定义循环控制变量
    printf("请输入 3 行 4 列的矩阵 a:\n");      //输出提示语
    for(i=0;i<3;i++)
        for(j=0;j<4;j++)
        {   scanf("%d",&a[i][j]);
            b[j][i]=a[i][j];                //矩阵转置
        }
    printf("转置后的矩阵 b 为:\n");             //输出提示
    for(i=0;i<4;i++)                        //输出转置后的矩阵 b
    {
        for(j=0;j<3;j++)
            printf("%5d",b[i][j]);
        printf("\n");                       //每输出 3 个元素后输出一个换行符
    }
    return 0;
}
```

【输出结果】

程序输出结果如图 5.10 所示。

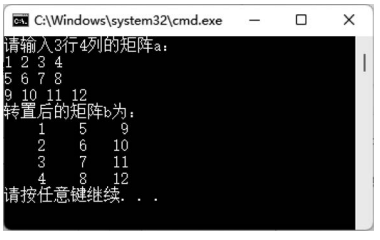


图 5.10　例 5.3 程序输出结果

【例 5.4】　某公司 2024 年上半年产品销售统计表如表 5.1 所示，求每种产品的月平均销售量和所有产品的总月平均销售量。

表 5.1　产品销售统计表

| 月份 | 产品 A | 产品 B | 产品 C | 产品 D | 产品 E |
|---|---|---|---|---|---|
| 1 | 30 | 21 | 50 | 35 | 42 |
| 2 | 35 | 15 | 60 | 40 | 40 |
| 3 | 32 | 18 | 56 | 37 | 50 |
| 4 | 40 | 25 | 48 | 42 | 48 |
| 5 | 36 | 23 | 52 | 33 | 46 |
| 6 | 41 | 19 | 55 | 39 | 52 |

【问题分析】

(1) 定义一个二维数组 a[5][6]存储该公司 5 种产品 6 个月的月销售量。再定义一个一维数组 aver[5]存储所求的 5 种产品的月平均销售量,定义变量 total 存储各产品月平均销售量累加和,定义变量 average 存储所有产品 6 个月的总月平均销售量。

(2) 使用双重 for 循环,外层循环次数为 5 次,内层循环次数为 6 次。内循环实现一种产品 6 个月的销量的累加。外循环每循环一次,求出一种产品 6 个月的月平均销售量并输出,同时将其累加到变量 total 中。

(3) 求所有产品 6 个月的总月平均销售量,变量 total 的值除以 5 即可得到所有产品的总月平均销售量。

【程序代码】

```
#include <stdio.h>
int main()
{   //定义循环控制变量 i 和 j,存储累加和的变量 sum
    int i,j,sum = 0;
    /* 定义各产品月平均销售量数组 aver[5]、各产品月平均销售量累加和变量 total、
       总月平均销售量 average */
    float aver[5],total = 0.0,average;
    //定义二维数组,并初始化
    int a[5][6] = {{30,35,32,40,36,41},{21,15,18,25,23,19},
                   {50,60,56,48,52,55},{35,40,37,42,33,39},
                   {42,40,50,48,46,52}};
    for(i = 0;i < 5;i++)                    //外循环
    {
        for(j = 0;j < 6;j++)                //内循环累加各产品 6 个月的月销售量
            sum = sum + a[i][j];
        aver[i] = sum/6.0;                  //计算各产品的月平均销售量
        printf("产品 %c 的月平均销售量:%.2f\n",65 + i,aver[i]);  //输出月平均销售量
        total += aver[i];                   //累加各产品的月平均销售量
        sum = 0;                            //sum 清零,为累加下一产品的月销售量做准备
    }
    average = total/5;                      //计算所有产品 6 个月的总月平均销售量
    printf("所有产品 6 个月的总月平均销售额:%.2f\n",average);
    return 0;
}
```

【输出结果】

程序输出结果如图 5.11 所示。

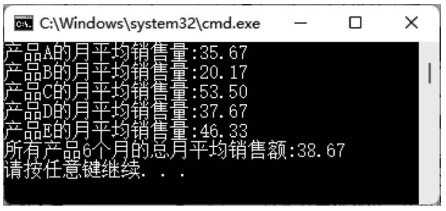


图 5.11　例 5.4 程序输出结果

## 5.3　字符数组和字符串

前面介绍的都是数值型数组，即数组元素都是数值。还有一种数组，其每个元素都是一个字符，也就是说，数组元素的数据类型都是 char 型的，除此之外，它与前面讲的数组没有区别。这种用来存储字符量的数组称为字符数组。

字符串应用广泛，但 C 语言中没有专门的字符串类型，字符串是存储在字符数组中的。

### 5.3.1　字符数组的定义和初始化

字符型数组的定义形式如下：

```
char  数组名[数组长度];
```

例如：

```
char a[5];
```

字符数组也可以是二维或多维数组。例如，char a[3][4]即为二维字符数组。

同样地，字符数组也允许在定义时进行初始化赋值。字符数组初始化的过程与数值型数组初始化的过程完全一样。例如：

```
char a[4] = {'G','o','o','d'};
```

赋值后各元素的值为 c[0]='G'、c[1]='o'、c[2]='o'、c[3]='d'。

字符型数组与数值型数组在初始化中的区别如下。

初始化时，如果大括号中初值的个数小于数组长度，则只给字符数组前面的元素赋值，剩下元素的初值为空字符(即'\0')，而前面讲过数值型数组初始化为 0。例如：

```
char b[9] = {'G', 'o', 'o', 'd'};
```

这样初始化后，字符数组 b 在内存中的存储形式如图 5.12 所示。

| b[0] | b[1] | b[2] | b[3] | b[4] | b[5] | b[6] | b[7] | b[8] |
|---|---|---|---|---|---|---|---|---|
| G | o | o | d | \0 | \0 | \0 | \0 | \0 |

图 5.12　字符数组 b 在内存中的存储形式

【例 5.5】　编写程序,使用字符数组存储下面一问一答的问候语,并输出。

```
How are you?
Fine! Thank you, and you?
```

【问题分析】

根据题目要求,可以使用两个一维字符数组存储这两句问候语。两个字符数组长度的确定,依据的是其所存储的问候语中字符的个数。数组与 for 循环相结合,输出字符数组中的各个字符,即可实现输出问候语。

【程序代码】

```
#include <stdio.h>
int main()
{
    //定义并初始化字符数组 greetings1、greetings2
    char greetings1[12] = {'H','o','w',' ','a','r','e',' ','y','o','u','?'};
    char greetings2[25] = {'F','i','n','e','!',' ','T','h','a','n','k',' ',
                           'y','o','u',',',' ','a','n','d',' ','y','o','u','?'};
    int i;                                  //定义循环控制变量 i
    for(i = 0;i < 12;i++)                   //输出字符数组 greetings1 中每个元素的值
        printf("%c", greetings1[i]);        //格式化输出语句中,输出字符用 %c
    printf("\n");
    for(i = 0;i < 25;i++)                   //输出字符数组 greetings2 中每个元素的值
        printf("%c", greetings2[i]);
    printf("\n");
    return 0;
}
```

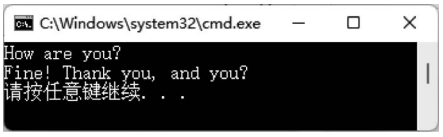


图 5.13　例 5.5 程序输出结果

【输出结果】

程序输出结果如图 5.13 所示。

**说明:**

也可以使用一个二维字符数组存储这两句问候语,该数组定义如下:

```
char greetings[2][25] = {{'H', 'o', 'w', ' ', 'a', 'r', 'e', ' ', 'y', 'o', 'u',
                          '?'}, {'F', 'i', 'n', 'e', '!', ' ','T', 'h', 'a',
                          'n', 'k',' ', 'y', 'o', 'u', ',', ' ', 'a', 'n', 'd',
                          ' ', 'y', 'o', 'u', '?'}};
```

然后使用双重 for 循环,输出两句问候语。

## 5.3.2　字符串

在 C 语言中没有专门的字符串类型,相应地也就没有字符串变量,C 语言使用字符数组来存储字符串。在第 2 章介绍字符串常量时,已说明系统会在字符串常量的末尾自动加上字符串结束标志('\0'),以便对字符串进行处理时能够判断字符串是否已经结束。因此,当把一个字符串存入一个数组时,系统也会把结束符'\0'存入数组。例如:

```
char str[6] = "China";
```

赋值结果，字符串数组 str 有 6 个元素，最后一个元素为'\0'。

存储字符串的字符数组的初始化有两种方法。

(1) 用字符常量初始化数组，用字符常量给字符数组赋初值要用大括号把字符常量括起来。例如：

```
char str[6] = {'C', 'h', 'i', 'n', 'a', '\0'};
```

数组 str 被初始化为"China"，其中最后一个元素的赋值'\0'可以省略。

**说明：**

① 如果提供赋值的字符个数少于数组元素的个数，则多余数组元素自动赋为'\0'。例如：

```
char str[10] = "China";
```

字符数组 str 从第 6 个元素开始之后全部赋为'\0'。

② 如果提供赋值的字符个数多于数组元素的个数，则出现语法错误。例如：

```
char str1[4] = {'C','h','i','n','a','\0'};        //错误
char str2[4] = "China";                           //错误
```

③ 初始化时，若字符个数与数组长度相同，则字符末尾不加'\0'，此时字符数组不能作为字符串处理，只能作字符逐个处理。初始化时是否加'\0'，要看是否作字符串处理。

(2) 用字符串常量初始化数组。例如：

```
char str[6] = "China"; 或 char str[6] = {"China"};
```

**说明：**

① 不论用字符常量初始化字符数组，还是用字符串常量初始化数组，若字符个数少于数组长度，系统都会自动在末尾字符后加'\0'。

② 用字符串常量初始化时，字符数组的长度可以省略。例如：

```
char str[ ] = "1a2b3c";
```

由于字符串常量的末尾由系统自动加上了字符串结束标志'\0'，因此，数组 str 的长度为 7，上面的语句等价于：

```
char str[7] = "1a2b3c";
```

显然，采用字符串常量进行初始化更加直观、方便，更符合人们的习惯。

**【例 5.6】** 编程将一个已知字符串存入一维字符数组中，并输出。

```
#include <stdio.h>
int main()
{
    //定义字符数组 str,并将字符串"Hello\nworld"存入该数组中
    char str[] = "Hello\nworld";
    printf("%s\n",str);          //输出字符串,字符串整体输出用%s 格式符
    return 0;
}
```

【输出结果】

程序输出结果如图 5.14 所示。


图 5.14　例 5.6 程序输出结果

## 5.3.3　字符串的输入和输出

### 1. scanf()和 printf()函数

scanf()和 printf()函数可以输入输出任何类型的数据。若要输入输出字符串,格式符为%s。

(1) 字符串输入。从键盘读取一个字符串:

```
scanf("%s",字符数组名);
```

**注意**:当从键盘输入完要输入的字符串时,字符数组自动包含一个结束标志'\0'。

scanf()函数的格式要求操作数是地址。假如 c 是字符数组名,"scanf("%s",&c);"的写法是不正确的。因为,字符数组名是字符串第一个字符的地址,是地址常量,对其操作不要再加地址运算符号。

默认情况下空格和回车键以及 Tab 键是作为字符串输入的结束符,所以 scanf()函数调用时的格式符%s,不能实现字符串中包含有空格的字符串的输入,需要使用格式符%[ ]。

```
char b[20];
scanf("%[^\n]",b);     //以换行符作为字符串输入的结束
```

(2) 字符串输出。向显示器输出一个字符串:

```
printf("%s",字符数组名);
```

**注意**:输出字符串字符时,遇到'\0'则结束。

【例 5.7】　编程实现在一个字符串中统计各元音字母(即 A、E、I、O、U)的个数。

注意,字母不分大小写。例如,输入 THIs is a boot,则输出应为 1 0 2 2 0。

【问题分析】

(1) 定义字符数组 s,用于存储一个字符串。定义计数数组 a[5],a[0]～a[4]依次存储元音字母 A、E、I、O、U 的个数。

(2) 从键盘输入一个字符串到字符数组 s 中。然后,从数组的第一个元素开始,利用循环语句依次检测数组元素的值是否为字符串结束标志('\0'),循环体依次判断该数组元素的值是否为元音字母 a、A、e、E、i、I、o、O、u、U,进行相应的处理,将计数数组 a 对应的数组元素值加 1,继续下一次循环,直到遇到字符串结束标志结束循环。

(3) 输出计数数组 a 中各元素的值,显示统计结果。

【程序代码】

```
#include <stdio.h>
int main()
{   char s[80];
    int a[5];
    int i,j;
    printf("Please enter a string:\n");
    scanf("%[^\n]",s);
    for(i=0;i<5;i++)  a[i]=0;
    for(i=0;s[i]!='\0';i++)
    {   j=-1;
        switch(s[i])
        {   case'a':
            case'A': j=0; break;
            case'e':
            case'E': j=1; break;
            case'i':
            case'I': j=2; break;
            case'o':
            case'O': j=3; break;
            case'u':
            case'U': j=4; break;
        }
        if(j>=0)  a[j]++;
    }
    for(i=0;i<5;i++)  printf("%4d",a[i]);
    printf("\n");
    return 0;
}
```

【输出结果】

程序输出结果如图 5.15 所示。

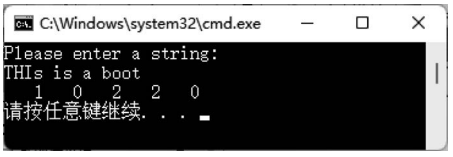


图 5.15　例 5.7 程序输出结果

### 2. gets()和 puts()函数

gets()和 puts()函数是专门用于字符串输入输出的库函数。当使用这两个库函数时需要包含头文件 string.h。

(1) 字符串输入。从键盘读取一个字符串:

```
gets(字符数组名)
```

gets()函数用于从终端输入一个字符串到字符数组,并且字符串中可以包含空格。通过 gets()函数输入字符串时,系统也会在字符串的末尾自动添加字符串结束标志'\0'。

(2) 字符串输出。向显示器输出一个字符串:

```
puts(字符数组名)
```

puts()函数用于将一个字符串(即以'\0'结束的字符序列)输出到终端,输出完字符串后换行。例如:

```
char a[10];              //定义长度为 10 的字符数组 a
gets(a);                 //用 gets()函数对字符数组 a 进行赋值
puts(a);                 //用 puts()函数输出字符数组 a 中的内容
```

**注意**:gets()和 puts()函数是对整个字符串的输入输出函数,对于这里定义的字符数组 a,当输入字符数少于 10 个时,输入输出是一样的字符串。但是当输入字符数大于或等于 10 个时就会造成溢出,从而导致运行出错。

**【例 5.8】** 编程实现凯撒加密,将待加密文本中的每个字符替换为其后面第 k 个字符。

**【问题分析】**

定义两个字符数组,一个字符数组 S 用来存储待加密文本字符串,一个字符数组 C 用来存储加密后的文本字符串。根据加密规则,密码字符的计算公式如下。

若是大写字母,计算公式为:

```
S[i] - 'A' + k) % 26 + 'A';
```

若是小写字母,计算公式为:

```
S[i] - 'a' + k) % 26 + 'a';
```

**【程序代码】**

```
#include <stdio.h>
#include <string.h>
#define MAX 100                              //待加密文本最大长度
int main()
{
    char S[MAX];                             //定义字符数组 S,存储待加密文本
    char C[MAX];                             //定义字符数组 C,存储加密后文本
    int i,k = 3;                             //定义循环控制变量 i,密码规则变量 k
    printf("Enter passage\n");               //输出提示语
    gets(S);                                 //从键盘读取待加密文本,存入字符数组 S
    i = 0;
    while(S[i]!= '\0')                       //依次处理待加密文本中的每个字符
    {
        if((S[i]>= 'A')&&(S[i]<= 'Z'))       //若为大写字母
            C[i] = (S[i] - 'A' + k) % 26 + 'A';  //密文字符
```

```
            else if((S[i]>='a')&&(S[i]<='z'))   //若为小写字母
                C[i]=(S[i]-'a'+k)%26+'a';       //密文字符
            else                                //非字母字符
                C[i]=S[i];                      //保持不变
            i++;
        }
        C[i]='\0';                              //在密文字符串的末尾添加字符串结束标志'\0'
        printf("Password\n%s\n",C);             //输出加密后的文本
        return 0;
    }
```

【输出结果】

程序输出结果如图 5.16 所示。

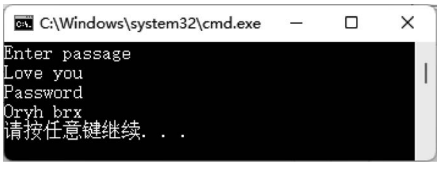


图 5.16　例 5.8 程序输出结果

## 5.3.4　字符串处理函数

C 语言具有丰富的字符串处理函数，常用的有 gets()与 puts()函数、strlen()函数、strcat()函数、strcmp()函数、strcpy()与 strncpyy()函数、strlwr()与 strupr()函数。当使用这些库函数时需要包含头文件 string.h。

### 1. strlen()函数——测字符串长度函数

函数调用形式如下：

```
strlen(字符数组名)
```

函数功能：测字符串的实际长度(不含字符串结束标志'\0')，并作为函数值返回。例如：

```
char s[]="C language";                  //初始化字符串 s
k=strlen(s);                            //调用 strlen()函数求字符串 s 的长度
printf("The length of the string is %d\n",k);   //输出字符串的长度值
```

### 2. strcat()函数——字符串连接函数

函数调用形式如下：

```
strcat(字符数组名 1,字符数组名 2)
```

函数功能：把字符数组 2 中的字符串连接到字符数组 1 中的字符串的后面，并删去字符串 1 后的串标志“\0”。本函数返回值是字符数组 1 的首地址。例如：

```
//定义字符数组 str1 和 str2,并初始化
char str1[30] = "My name is ",str2[ ] = "Xiao ming";
strcat(str1,str2);          //调用字符串连接函数 strcat()将 str2 接到 str1 的后面
puts(str1);                 //输出 str1
```

**注意**：字符数组 str1 应定义得足够长，否则不能全部装入被连接的字符串。

### 3. strcmp()函数——字符串比较函数

函数调用形式如下：

```
strcmp(字符数组名 1, 字符数组名 2)
```

函数功能：将两个字符数组中的字符串从左至右逐个相比较，即比较字符的 ASCII 码值大小，并由函数返回值返回比较结果。

- 字符串 1＝字符串 2，返回值＝0；
- 字符串 1＞字符串 2，返回值＞0；
- 字符串 1＜字符串 2，返回值＜0。

本函数也可用于比较两个字符串常量，或比较字符数组和字符串常量。例如：

```
char str1[15],str2[15];              //定义字符数组 str1 和 str2
printf("Please enter string1:\n");   //输出提示语
gets(str1);                          //输入字符串 1
printf("Please enter string2:\n");   //输出提示语
gets(str2);                          //输入字符串 2
k = strcmp(str1,str2);               //调用函数 strcmp()比较字符串 1 和字符串 2 的大小
if(k == 0) printf("str1 = str2\n");  //如果返回值为 0,那么字符串 1 = 字符串 2
if(k > 0) printf("str1 > str2\n");   //如果返回值> 0,那么字符串 1 >字符串 2
if(k < 0) printf("str1 < str2\n");   //如果返回值< 0,那么字符串 1 <字符串 2
```

### 4. strcpy()和 strncpy()函数——字符串复制函数

(1) strcpy()函数，函数调用形式如下：

```
strcpy(字符数组名 1,字符数组名 2)
```

函数功能：把字符数组 2 中的字符串复制到字符数组 1 中。字符串结束标志“\0”也一同复制。字符数组 2 也可以是一个字符串常量，这时相当于把一个字符串赋给一个字符数组。

(2) strncpy()函数，函数调用形式如下：

```
strncpy(字符数组名 1,字符数组名 2, n)
```

函数功能：把字符数组 2 中的前 n 个字符复制到字符数组 1 中，取代字符数组 1 中原有的前 n 个字符。例如：

```
char str1[20],str2[] = "These are apples.", str3[] = "Those are bananas.";
strcpy(str1,str2);              //调用字符串复制函数 strcpy(),将字符串 2 复制到字符串 1 中
puts(str1);                     //输出字符串 1
/* 调用字符串复制函数 strncpy(),将字符串 3 中的前 4 个字符复制到字符串 2 中,
    取代字符串 2 中原有的前 4 个字符。 */
strncpy(str2,str3,4);
puts(str2);                     //输出字符串 2
```

### 5. strlwr()和 strupr()函数——字符串字母大小写转换函数

(1) strlwr()函数是小写字母转换函数,函数调用形式如下:

```
strlwr(字符数组名)
```

函数功能:将字符串中的大写字母转换成小写字母。

(2) strupr()函数是大写字母转换函数,函数调用形式如下:

```
strupr(字符数组名)
```

函数功能:将字符串中的小写字母转换成大写字母。例如:

```
char s[] = "how ARE You?";          //定义字符数组 s,并初始化
strlwr(s);                          //将 s 中的字母转换成小写字母
puts(s);                            //输出 s
strupr(s);                          //将 s 中的字母转换成大写字母
puts(s);                            //输出 s
```

## 5.3.5　字符数组应用举例

【例 5.9】　约瑟夫环问题。设有编号为 1,2,…,n 的 n(n>0)个人围成一个圈,从第 1 个人开始报数,报到 m 时停止报数,报 m 的人出圈,再从他的下一个人起重新报数,报到 m 时停止报数,报 m 的人出圈……,直到剩余 1 个人出圈。当任意给定 n 和 m 后,求 n 个人出圈的次序。程序要求按出圈次序打印每个出圈人的姓名。

【问题分析】

(1) 定义一个二维字符数组存储所有人员的姓名即 name[N][LEN],一旦人员出圈,则将对应数组姓名字符串置为空串。

(2) 如何在数组中找到出圈人员?这里设置一个下标变量和两个计数器变量。

- index:下标变量。
- counter:报数计数器变量。
- outCounter:出圈人数统计计数器变量。

(3) 下标 index 是 0～(N－1)不断累加循环,当 index=N 时,重置为 0,继续循环,直到只剩下最后一个人出圈。这就构成了整个程序的主循环。循环的结束条件是出圈总人数达到 N 即结束。

(4) 主循环每循环一次,index++,然后判断字符数组 name 当前 index 下标所在元素

是否为空,即 name[index]是否为空串,如果为空串,表示此人已经出圈,直接进入下一次循环。只有 name[index]不为空串,表示此人还没有出圈,此时报数计数器+1,然后判断该报数计数器是否与之前设定的 m 值相等,如果不相等,则不是要出圈人员,进入下一次循环;如果与 m 相等,则找到一个出圈人员,此时出圈人数统计计数器+1,设置 name[index]为空串,表示该人员已出圈,counter 清零,重新开始计数,继续循环。

【程序代码】

```
#include <stdio.h>
#include <string.h>
#define MAX 20                          //总人数上限
#define LEN 10                          //人员姓名长度上限
int main()
{
    char name[MAX][LEN+1];
    int order[MAX];                     //记录出圈顺序数组
    int N;                              //总人数
    int index=0;                        //姓名字符数组下标,当 index 为 N 时,重置 index=0
    int counter=0;                      //报数计数器,当 counter 为 m 时,该人出圈,counter 重置 0
    int outCounter = 0;                 //出圈人数统计计数器,当 outCounter 为 N 时,循环结束
    int m;                              //表示数到第 m 时该人出圈
    int i;                              //循环控制变量 i
    printf("请输入总人数:");
    scanf("%d",&N);
    printf("请输入数到第几个人出圈:");
    scanf("%d",&m);
    printf("请依次输入%d个人的姓名(姓名长度不超过10个字符):\n",N);
    for(i=0;i<N;i++){     scanf("%s",name[i]);     }
    while(outCounter<N)                         //只要出圈人数<总人数,继续循环
    {
        if(strcmp(name[index],"\0")!=0)         //该人员未出圈
        {
            counter++;                          //报数计数器+1
            if(counter == m)                    //报数到第 m 时,该人出圈
            {
                ++outCounter;                   //出圈人数统计计数器+1
                //该人出圈,用'\0'标记
                printf("第%2d个出圈者:%s\n",outCounter,name[index]);
                strcpy(name[index],"\0");
                order[index]=outCounter;        //记录出圈顺序
                counter=0;                      //counter 重置 0
            }
        }
        index++;
        if(index==N) index=0;
    }
    printf("出圈顺序:\n");
    for(i=0;i<N;i++)                            //输出出圈顺序
        printf("%d ",order[i]);
    printf("\n");
    return 0;
}
```

【输出结果】

程序输出结果如图 5.17 所示。

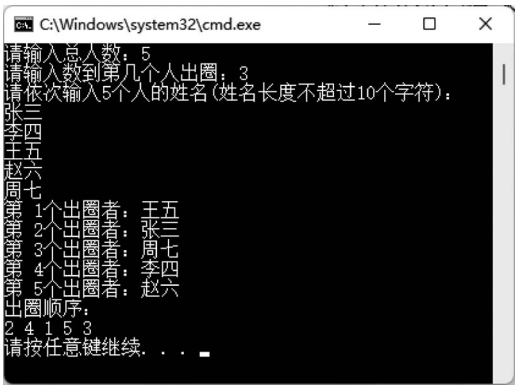


图 5.17　例 5.9 程序输出结果

**思考：**

复杂约瑟夫环问题：1～n 个人围成一圈，每个人手中有个号码，读入一个数 m，从第一个人开始报数，报到 m 的人出圈；下一个人接着从 1 报数，报到出圈人手里的号码的人出圈，依次进行。求 n 个人出圈的次序。

## 5.4　结构体数组

### 5.4.1　结构体类型的定义

结构体类型是一种构造数据类型，由程序设计者根据实际需要自己定义，主要用于把若干相关的类型不同(也可以相同)的数据组合成一个有机的整体，方便处理和传递。

比如要管理一个班级的学生成绩：每个学生信息包含学号、姓名、性别、成绩。这些项都有内在联系，它们是一个整体，都表示同一个学生的信息。如果将它们定义为互相独立的变量的话，就无法体现它们的内在联系，而且处理也不方便。能否用一个变量来保存一个学生的学号、姓名、性别、成绩的所有信息呢？答案是能，这就是结构体变量。在定义结构体变量之前，先要定义结构体类型，定义形式如下：

```
struct [结构体名]
{
    数据类型 成员 1 的名字;
    数据类型 成员 2 的名字;
    数据类型 成员 3 的名字;
    ……
};
```

struct 是定义结构体类型的关键字。结构体名是一个合法标识符，由程序设计者自己指定，与变量名的命名规则一样，遵循见名知义的原则，称为结构体类型名。大括号中的结构体成员表，称为结构体。结构体成员表包含若干成员。注意大括号后面的分号不能少，因

为这是一条完整的语句。

例如,表示学生信息的结构体类型可定义如下:

```
struct student
{
    int num;                          //存储一名学生的学号
    char name[20];                    //存储一名学生的姓名
    char sex;                         //存储一名学生的性别
    float score;                      //存储一名学生的成绩
};
```

这里的 struct student 是根据实际需要定义的一种新的数据类型,它相当于一个模型,其中并无具体数据,系统也不会为之分配实际内存单元。其功能相当于 int、float 等,可以用 struct student 这种结构体数据类型来定义相应的结构体变量。

结构体类型定义应注意如下几点。

(1) 结构体类型定义描述了结构体的组织形式,在程序编译时并不会为它分配存储空间。只是规定了一种特定的数据结构类型及它所占用的存储空间的存储模式。

(2) 结构体成员可以是简单变量、数组、指针、结构体或共用体等,结构体成员的定义方式与变量和数组的定义方式相同,只是不能初始化。

(3) 结构体可以嵌套使用,即一个结构体变量也可以成为另一个结构体的成员。例如,有一组学生的信息包括学号、姓名、出生年、月、日,则可进行如下结构体类型的声明:

```
struct date
{
    int year;
    int month;
    int day;
};                  //定义了一个包含 year、month、day 三个成员的结构体数据类型 struct date
struct student
{
    int nmu;
    char name[20];
    struct date birthday;
};                  //结构体数据类型 struct student 中有一个成员是结构体类型 struct date
```

(4) 结构体类型定义可以位于自定义函数内部,也可以位于自定义函数外部。在自定义函数内部定义的结构体,只对该自定义函数内部可见;在自定义函数外部定义的结构体,对定义点到源文件结束之间的所有函数都是可见的。一般将结构体类型定义放在源文件的开始部位。

(5) 结构体成员名可以与程序中其他变量同名,系统会自动识别它们,两者不会混淆。

## 5.4.2　结构体变量的定义和引用

### 1. 结构体变量的定义

为了能在程序中使用结构体类型的数据,应当定义结构体类型的变量,并在其中存储具

体的数据。结构体变量定义一般采用如下 3 种形式。

(1) 先声明结构体类型再定义结构体变量。

5.4.1 节已经定义了一个 struct student 的结构体数据类型,可以用该数据类型来定义变量,例如:

```
struct student S1;
```

与整型变量 a 的定义形式做对比:

```
int a;
```

这里的结构体变量的定义中,struct student 是结构体数据类型,功能相当于 int,即声明变量的数据类型; S1 是结构体变量名,功能相当于 a。

应当注意,将一个变量定义为标准类型(基本数据类型)与定义为结构体类型的不同之处在于:

① 定义结构体变量不仅要求指定变量为结构体类型,而且要求指定为某一特定的结构体类型,因为可以定义出许多种具体的结构体类型,如 struct student(后面的 student 就是该结构体类型的特定名字,用来与其他结构体数据类型区分开)。而在定义整型变量时,只需指定为 int 型即可。

② 定义基本数据类型变量,都可以用系统提供的相关数据类型直接定义,但定义结构体数据类型变量,必须先声明结构体数据类型再进行变量的定义。

在定义了结构体变量后,系统会为结构体变量分配内存单元。

这种定义方法的特点是:用声明的结构体类型 struct student 定义了一次结构体变量之后,在此之后的任何位置还可用 struct student 类型来定义其他结构体变量。

若程序规模比较大,可将对结构体类型的声明集中放到一个文件中(以.h 为后缀的头文件),便于修改和使用。若其他源文件需要用到此结构体类型,则可用 #include 命令将该头文件包含到本文件中。

(2) 在声明结构体类型的同时定义结构体变量。其定义形式如下:

```
struct  结构体名
{
    数据类型 成员 1 的名字;
    数据类型 成员 2 的名字;
    数据类型 成员 3 的名字;
    …
}结构体变量名表;
```

例如:

```
struct student
{
    int num;
    char name[20];
    char sex;
    float score;
}S1;
```

该例中结构体数据类型名为 struct student,用它定义了一个结构体变量 S1。

(3) 直接定义结构体变量,不出现结构体名。其定义形式如下:

```
struct
{
    数据类型 成员 1 的名字;
    数据类型 成员 2 的名字;
    数据类型 成员 3 的名字;
    …
}结构体变量名表;
```

例如:

```
struct
{
    int num;
    char name[20];
    char sex;
    float score;
}S1;
```

这种定义方法的特点是:不能用定义的结构体数据类型来另行定义其他结构体变量,要想定义新的结构体变量,就必须将 struct{}这部分重写。

结构体变量的定义应注意如下几点。

(1) 注意结构体数据类型的定义和结构体变量的定义的区别。结构体数据类型的定义描述了结构体的类型的模式,不分配内存;而结构体变量定义则是按照结构体声明中规定的结构体类型(或内存模式),在编译时为结构体变量分配内存单元。该变量和其他变量一样可以进行赋值、存取或运算等操作,但结构体数据类型是无法实现这些操作的。

(2) 结构体变量的定义一定要在结构体数据类型定义之后或与结构体数据类型定义同时进行。若结构体数据类型没有定义,不能用它来定义结构体变量。

(3) 结构体变量中的成员可以单独使用,其作用和地位与一般变量相同。

(4) 系统为结构体变量分配内存的大小理论上是各成员内存量总和,但有时与 sizeof()函数计算出来的值不是完全一样,这个实际的内存量,不仅与所定义的结构体类型有关,还与计算机系统及编译系统有关。通常系统为结构体变量分配内存的大小,会大于或等于所有成员所占内存字节数的总和。关于这一点,有兴趣的读者可以查阅相关资料。

### 2. 用 typedef 定义数据类型

在使用结构体类型定义结构体变量时,如果频繁使用 struct student 会比较烦琐。可以利用 typedef 为结构体数据类型定义一个别名,方便使用,如下面的语句:

```
struct student
{
    int num;
    char name[20];
    char sex;
    float score;
```

```
};
typedef struct student STU;
```

或者

```
typedef struct student
{
    int num;
    char name[20];
    char sex;
    float score;
}STU;
```

以上两种定义方式是等价的，都是为 struct student 这种结构体数据类型定义了一个新的名字 STU，利用 STU 定义结构体变量与利用 struct student 定义结构体变量是一样的。即，下面两条语句是等价的，两者都能用于定义结构体变量。显然，前者定义变量的形式更简洁，代码可维护性也更强。

```
STU S1,S2;
struct student S1,S2;
```

当然，利用 typedef 也可以为系统固有的数据类型定义一个别名。例如：

```
typedef int INTEGER;              //为 int 数据类型定义了一个新名字 INTEGER
typedef unsigned int UINT ;       //为 unsigned int 数据类型定义了一个新名字 UINT
```

则若程序中出现

```
INTEGER a;
UINT b;
```

即表示定义了一个 int 型的变量 a，一个 unsigned int 型的变量 b。

**注意**：typedef 只是为一种已存在的类型定义一个新的名字而已，并不是定义一种新的数据类型。

### 3. 结构体变量的引用

定义了结构体变量后，可以引用该变量。但需注意如下几点。

(1) 不能将一个结构体变量作为一个整体进行输入输出操作，只能对每个具体的成员进行输入输出操作。

如有已定义的结构体变量 S1，不能按如下方式引用：

```
printf("%d%c%d%d%d",S1);
```

访问结构体变量的成员,需使用“成员运算符”(也称“圆点运算符”)。其访问形式如下:

```
结构体变量名.成员名
```

例如,可用下面的语句为结构体变量 S1 的 score 成员进行赋值。

```
S1.score = 90;
```

S1.score 为结构体成员,与其他类型变量的使用方法是一样的。

**注意**:结构体变量不能进行整体输入输出,但允许具有相同结构体类型的结构体变量间赋值。

例如:

```
STU S1,S2;
S1.num = 101; S1.score = 99; …
S2 = S1;          //S2 得到了 S1 中每个成员的值
```

(2) 如果成员本身又属一个结构体类型,则要用若干个成员运算符,一级一级地找到最低一级的成员。例如:

```
struct date
{
    int year;
    int month;
};
struct student
{
    int num;
    char name[20];
    struct date birth;
}S1,S2;
```

若要引用结构体变量 S1 的 birth 成员的 year 成员,则需如此引用:S1.birth.year。

### 5.4.3 结构体变量的初始化

与其他数据类型的变量一样,可以对结构体变量进行初始化,即在定义结构体变量的同时,对其成员指定初始值。

结构体变量初始化的形式如下:

```
struct  结构体类型名  结构体变量名 = { 初始数据 };
```

对结构体变量初始化应注意如下几点。

(1) 初始化数据与数据之间用逗号分隔开。

(2) 初始化数据的个数要与被赋值的结构体成员的个数相等。

（3）初始化数据的类型要与相应的结构体成员的数据类型一致。

（4）不能直接在结构体成员表中对成员赋初值。

例如：

```
struct student
{
    int num;
    char name[20];
    char sex;
    float score;
};
struct student S1 = {1001,"zhao",'M',85.0};
```

或者

```
struct student
{
    int num;
    char name[20];
    char sex;
    float score;
}S1 = {1001,"zhao",'M',85.0};
```

对已经初始化的结构体变量，可以用 printf()函数将其数据输出：

```
printf("学号 %d 的学生的成绩是 %d。\n",S1.num,S1.score);
```

**注意**：下面对结构体变量初始化的方法是错误的，因为不能直接在结构体成员表中对成员赋初值。

```
struct student
{
    int num = 1001;
    char name[20] = "zhao";
    char sex = 'M';
    float score = 85.0;
}S1;
```

## 5.4.4　结构体数组的定义

存储一个学生的信息可以用一个结构体变量，但如果要存储一个班的学生信息，则需要用结构体数组。结构体数组必须先定义，后引用。其定义形式与定义结构体变量的方法差不多，只需声明其为数组即可。例如，下面定义了表示学生信息的结构体数据类型：

```
struct student
{
    int num;
```

```
    char name[20];
    char sex;
    float score;
};
```

若该班有 30 名学生,就可以用结构体数组表示,即:

```
struct student s[30];
```

其中 s[0]、s[1]、…、s[29]分别存储一个学生的信息。

也可以在定义结构体类型的同时定义一个结构体数组,例如:

```
struct student
{
    int num;
    char name[20];
    char sex;
    float score;
}studs[30];
```

或者

```
struct
{
    int num;
    char name[20];
    char sex;
    float score;
}studs[30];
```

### 5.4.5　结构体数组的初始化

结构体数组也可在定义的同时进行赋值,即对其进行初始化。下面对结构体数组 studs[30]的前 3 个元素进行初始化:

```
struct student studs[30] = {{1001, "Wang Lin", 'M',89.5},
                            {1002, "Li Gang", 'M',79.6},
                            {003, "Liu Yan", 'F',90.7}};
```

当所提供的数据个数小于数组的大小时,系统将只对数组的前几个元素赋初值,其他元素自动赋默认值。

当赋初值的数据个数与所定义的数组元素的个数相等时,数组元素的个数可以省略,此时系统会自动根据所提供的数据个数来确定数组的大小。例如:

```
struct student studs[ ] = {{1001,"Wang Lin",'M',89.5},
                           {1002,"Li Gang",'M',79.6},
                           {003,"Liu Yan",'F',90.7}};
```

### 5.4.6　结构体数组的引用

下面以一个例子来说明结构体数组的引用。

【例 5.10】　现有 5 名学生的信息如表 5.2 所示，每名学生信息包括学号(int num)、姓名(char name[20])、性别(char sex)、成绩(float score)。计算这 5 名学生的平均成绩并输出。

表 5.2　某班 5 位学生信息表

| 学　　号 | 姓　　名 | 性　　别 | 成　　绩 |
|---|---|---|---|
| 1001 | 赵敏 | F | 81.9 |
| 1002 | 张伟 | M | 83.7 |
| 1003 | 刘强 | M | 82.5 |
| 1004 | 王凯 | M | 90.4 |
| 1005 | 李娟 | F | 75.8 |

【问题分析】

(1) 定义结构体类型 student，结构体成员包括 num(学号)、name(姓名)、sex(性别)和 score(成绩)。

(2) 定义一个结构体数组 studs，同时用表 5.2 中 5 名学生的信息进行初始化。

(3) 计算平均成绩：遍历结构体数组，将每名学生的成绩相加，然后除以学生的数量。

(4) 输出平均成绩。

解决该问题的算法流程图如图 5.18 所示。

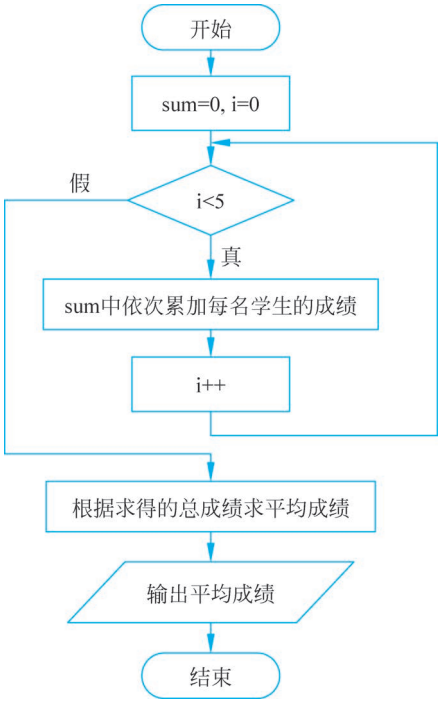


图 5.18　例 5.10 的算法流程图

【程序代码】

```
#include <stdio.h>
typedef struct student
{
    int num;                                          //存储一名学生的学号
    char name[20];                                    //存储一名学生的姓名
    char sex;                                         //存储一名学生的性别
    float score;                                      //存储一名学生的成绩
}STU;
int main()
{
    int i;
    float sum=0,ave;
    STU studs[]={{1001, "赵敏", 'F',81.9},{1002, "张伟", 'M',83.7},
                 {1003, "刘强", 'M',82.5},{1004, "王凯", 'M',90.4},
                 {1005, "李娟", 'F',75.8}};           //定义结构体数组,并进行初始化
    for(i=0;i<5;i++)
        sum=sum+studs[i].score;                       //计算 5 名学生的成绩总和
    ave=sum/5;                                        //求平均成绩
    //输出 5 名学生的平均成绩
    printf("5 名学生的平均成绩为:%5.1f\n",ave);
    return 0;
}
```

【输出结果】

程序输出结果如图 5.19 所示。


图 5.19　例 5.10 程序输出结果

## 5.5　“学生成绩管理系统”案例分析与实现

在“学生成绩管理系统”中,要处理若干名学生的信息,每名学生的信息包括学号、姓名、C 语言成绩、高数成绩、英语成绩、总成绩、平均成绩,这些信息不是一个个孤立的数据,而是相互联系的一个整体,那么在编程时如何体现出它们之间的联系呢? 这就需要用到结构体数组。

### 1. 学生信息结构体类型定义

表示学生信息的结构体类型定义如下:

```
typedef struct student                                //学生信息结构体 student
{
    char num[10];                                     //学号
    char name[15];                                    //姓名
    int cgrade;                                       //C 语言成绩
```

```
    int mgrade;              //高数成绩
    int egrade;              //英语成绩
    int total;               //总成绩
    float ave;               //平均成绩
}Student;
```

结构体类型 struct student 规定了学生信息的组织形式，为了简化程序，我们只取了三门课程的成绩。

### 2. 学生信息结构体数组的定义

存储一个学生的信息可以用一个结构体变量，但如果要存储一个班或者更多的学生信息，则需要用结构体数组。

学生信息结构体数组的定义如下：

```
#define MAXSIZE 30           //定义符号常量 MAXSIZE,用作学生信息结构体数组的长度
Student stud[MAXSIZE];       //定义学生信息结构体数组
```

把学生信息结构体数组的长度定义为符号常量是为了方便后期处理数组大小。如果后期需要更改结构体数组的大小，只需修改这一处即可。

### 3. 案例中部分功能模块的实现

下面利用循环语句和结构体数组来实现“学生成绩管理系统”的数据录入、显示、成绩统计和成绩排序 4 个功能模块。整个系统采用模块化程序设计，每个功能模块对应一个自定义函数，这样做最大的好处是：程序结构清晰、便于编程和维护。自定义函数的语法知识将在第 6 章进行详细介绍，读者现在只需要关注自定义函数内部的实现方法。

**【例 5.11】** 从键盘输入若干名学生的信息。请编写在学生人数未知情况下的数据输入程序。

**【问题分析】**

由于学生人数未知，所以循环不能由学生人数来控制，而应改为判断某一标志值：当输入的学号为 0 时表示输入结束，否则继续输入，输入的学生信息依次存储在学生结构体数组 stud 中。采用这种实现方式，用户使用时非常方便、灵活。

**【程序代码】**

```
int stu_num = 0;             //定义变量 stu_num,用于存储学生总人数,初始值为 0
void append(){
    int i = stu_num;
    while(1)
    {
        printf("请输入第 %d 个学生的信息\n",i + 1);
        printf("请输入学号:");
        scanf("%s",stud[i].num);
        if(strcmp(stud[i].num,"0") == 0) break;
        printf("请输入姓名:");
        scanf("%s",stud[i].name);
```

```
            printf("请输入 C 语言成绩:");
            scanf("%d",&stud[i].cgrade);
            printf("请输入数学成绩:");
            scanf("%d",&stud[i].mgrade);
            printf("请输入英语成绩:");
            scanf("%d",&stud[i].egrade);
            stud[i].total = stud[i].cgrade + stud[i].mgrade + stud[i].egrade;
            stud[i].ave = stud[i].total/3.0;
            i++;
        }
        stu_num = i;
        printf("总共已输入 %d 个学生的信息\n",stu_num);
    }
```

【输出结果】

程序输出结果如图 5.20 所示。


图 5.20　例 5.11 程序输出结果

**说明**：上述代码只实现了学生信息的简单录入功能，并没有对录入数据的合法性进行检查。例如，检查录入的学生学号是否重复？学生成绩是否在合法范围内？这些问题的解决放到第 6 章中介绍。

【例 5.12】　显示“学生成绩管理系统”中所有学生的信息。

【问题分析】

利用 for 循环遍历学生信息结构体数组，输出当前数组元素的各数据成员的值，循环结束，所有学生信息显示完毕。

【程序代码】

```
void display()
{
    int i;
    if (stu_num == 0)                    //学生人数等于 0
```

```
    {
        printf("\n\t 学生信息为空!");
    }
    else                                    //学生人数大于 0
    {
        printf("全部学生信息如下:\n");
        printf("学生学号\t 姓名\tC 语言\t 数学\t 英语\t 总成绩\t 平均成绩\n");
        for(i = 0;i < stu_num;i++)
        {
            printf(" % s\t\t % s\t % d\t % d\t % d\t % d\t % .1f\n",
                stud[i].num,stud[i].name,stud[i].cgrade,stud[i].mgrade,
                stud[i].egrade,stud[i].total,stud[i].ave);
            if((i + 1) % 10 == 0)   systom("pause");
        }
    }
    printf("共计 % d 人\n",stu_num);
}
```

【输出结果】

程序输出结果如图 5.21 所示。

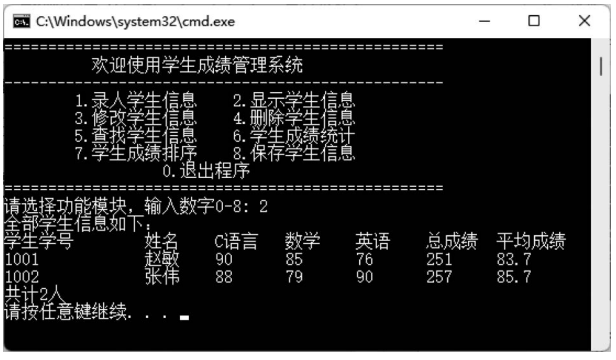


图 5.21　例 5.12 程序输出结果

【例 5.13】　统计“学生成绩管理系统”中指定课程的成绩不及格人数、最高分和平均分。

【问题分析】

(1) 以菜单的形式让用户选择对哪门课程的成绩进行统计,然后利用 switch 循环来根据用户的选择跳到需要的操作。

(2) 统计不及格人数、最高分：用 counter 变量来保存当前的不及格人数,其初值为 0；统计时检查每个学生的指定课程的成绩是否小于 60,如果是,counter 变量的值加 1。循环结束后,counter 值即为不及格人数。用 max 变量来保存当前的最高分,其初值为 0；统计时将每个学生的指定课程的成绩与 max 进行比较,如果高于 max,则将该学生的成绩赋给 max；当所有学生处理完后,max 中就保存了所有学生的最高分。

(3) 求所有学生的平均分,需要先求出总分,并保存在 sum 变量中。

【程序代码】

```
void total()
{
```

```
    int choice,i,counter = 0,max = 0,sum = 0;
    printf("请选择你想要统计成绩的课程\n");
    printf(" 0.退出 1.C 语言成绩 2.数学成绩 3.英语成绩\n");
    scanf(" %d", &choice);
    switch (choice){
        case 1: for(i = 0;i < stu_num;i++)          //统计 C 语言成绩不及格人数、最高分、总分
        {
            if(stud[i].cgrade < 60) counter++;
            if(stud[i].cgrade > max) max = stud[i].cgrade;
            sum += stud[i].cgrade;
        }
        break;
        case 2: for(i = 0;i < stu_num;i++)          //统计数学成绩不及格人数、最高分、总分
        {
            if(stud[i].mgrade < 60) counter++;
            if(stud[i].mgrade > max) max = stud[i].mgrade;
            sum += stud[i].mgrade;
        }
        break;
        case 3: for(i = 0;i < stu_num;i++)          //统计英语成绩不及格人数、最高分、总分
        {
            if(stud[i].egrade < 60) counter++;
            if(stud[i].egrade > max) max = stud[i].egrade;
            sum += stud[i].egrade;
        }
        break;
        case 0: printf("不统计返回主菜单\n");   break;
        default:printf("无效操作数!\n");
    }
    if(choice == 1||choice == 2||choice == 3)
    {
        printf("统计结果如下:\n");
        printf("不及格人数\t 最高分\t 平均分\n");
        printf(" %d\t\t %d\t %5.2f\n",counter,max,(float)sum/stu_num);
    }
}
```

【输出结果】

程序输出结果如图 5.22 所示。

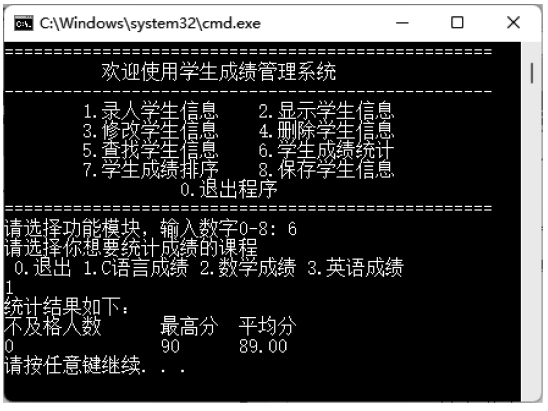


图 5.22　例 5.13 程序输出结果

【例 5.14】　对“学生成绩管理系统”案例中的所有学生信息，采用冒泡排序法按总成绩从高到低进行排序。

【程序代码】

```
void sort()
{   //冒泡排序
    int i,j;
    Student temp;
    for(i=1;i<stu_num;i++)
    {
        for(j=0;j<stu_num-i;j++){
            if(stud[j].total<stud[j+1].total)
            {   temp=stud[j];stud[j]=stud[j+1];stud[j+1]=temp;     }
        }
    }
    printf("排序结果如下:\n");
    printf("序号\t学生学号\t姓名\tC语言\t数学\t英语\t总成绩\t平均成绩\n");
    for(i=0;i<stu_num;i++)
    {
        printf("%d\t%s\t%s\t%d\t%d\t%d\t%d\t%.1f\n",i+1,
            stud[i].num,stud[i].name,stud[i].cgrade,stud[i].mgrade,
            stud[i].egrade,stud[i].total,stud[i].ave);
    }
}
```

【输出结果】

程序输出结果如图 5.23 所示。

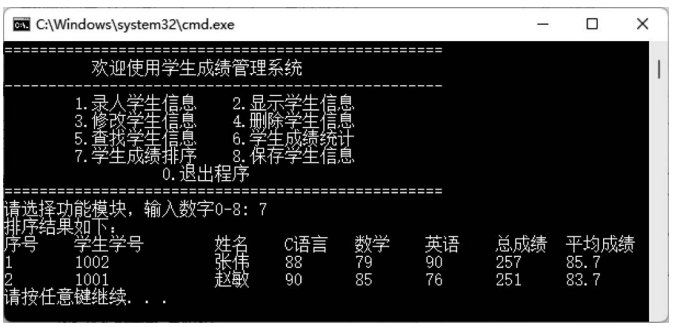


图 5.23　例 5.14 程序输出结果

本节主要介绍了简单的学生信息录入、显示全部学生信息、成绩统计、按总成绩排序四项功能的实现。“学生成绩管理系统”案例的更完善、更多功能的实现将在第 6 章中介绍。本节案例完整代码请参见本书配套教学资源。

## 5.6　常见错误分析

### 1. 数组下标越界

例如：

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

由于数组 a 定义有 5 个元素,下标为 0～4,当 i 为 5 时,实际上 scanf()函数使用形式如下:

```
scanf("%d",&a[5]);
```

而数组 a 中根本就没有 a[5]这个元素,所以这次接收输入是错误的。C 语言本身对下标越界不做检查,因此发生程序运行错误。

**【运行报错信息】**

运行报错信息如图 5.24 所示。

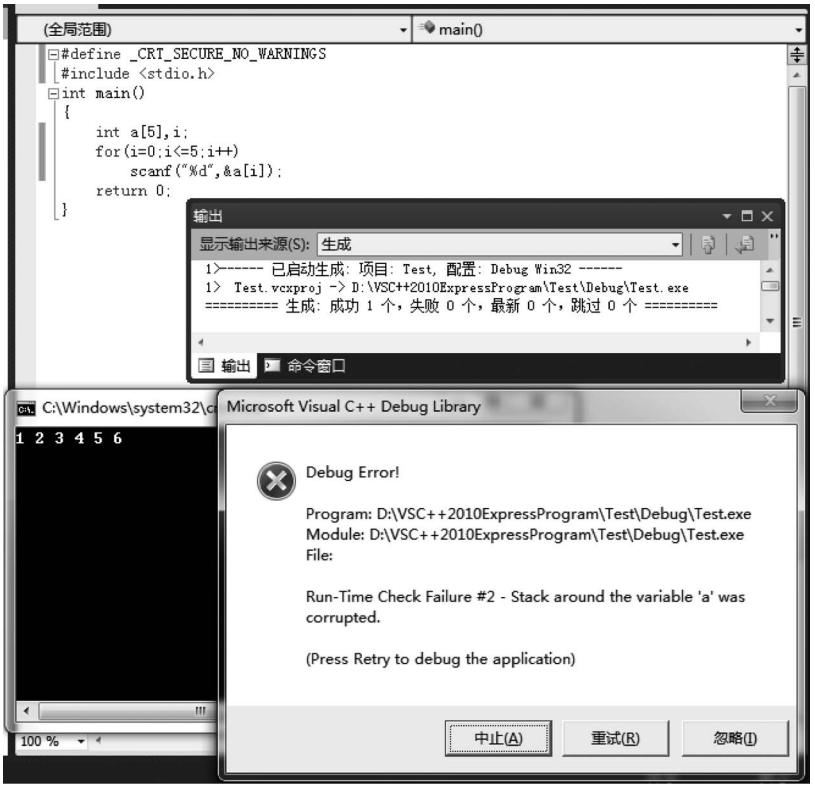


图 5.24　运行报错信息截图

### 2. 不能对数组整体进行读取操作

例如:

```
int a[5]={1,23,67,52};
printf("a=%d",a);
```

是错误的，C 语言不允许对数组作整体的操作，如果想把数组 c 的元素输出，需要用循环来实现，例如：

```
int a[5] = {1,23,67,52};
for(i = 0;i < 4;i++)
    printf(" % d",a[i]);
```

同样也不能用 scanf()函数一次接收一个数组的值，如“scanf("%d",&a);”是错误的，也得用循环来实现。但这两种情况在编译时都不会提示错误，使得程序结果是错误的。

### 3. 二维数组初始化时，第二维长度不能省略

例如：

```
int b[ ][ ] = {{1,1,1,1 },{2,2,2,2 },{3,3,3,3 }};
char c[3][ ] = {"good", "morning", "Wang! " };
```

是错误的。一维数组初始化时长度可以省略，二维数组初始化时第一维长度可以省略，但第二维长度不能省略。

**【编译报错信息】**

编译报错信息如图 5.25 所示。

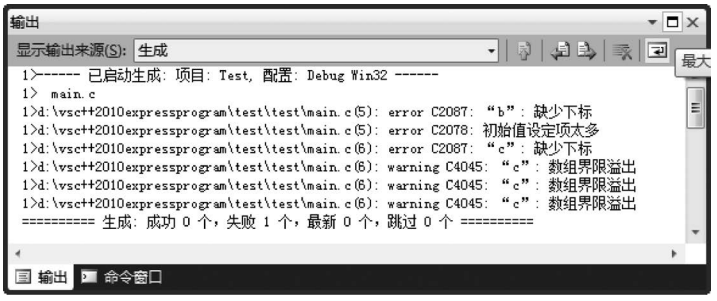


图 5.25　编译报错信息截图 1

**【错误分析】**

提示下标丢失。

### 4. 接收字符串时，使用了取址运算符

例如：

```
char str[10];
scanf(" % s",&str);
```

是错误的，由于数组名本身就代表地址，所以不应再加 & 符号，实际上，只要是用%s 控制字符，其对应字符数组名前就不加 & 符号了，正确的写法为“scanf("%s",str);”。这种错误在编译的时候同样不会提示错误。

### 5. 数组赋值只能是对每个元素赋值,不能整体赋值

例如:

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

或

```
char str[];
str = "hello";
str[6] = "hello";
```

都是错误的,其实这种错误跟上面的第二种错误是一样的,C 语言不支持对数组的整体操作,但使用者由于看到数组初始化的情形,就以为能够把字符串赋给一个数组,这种错误出现的频率很高,应加以重视。这种赋值只能在初始化时进行。

**【编译报错信息】**

编译报错信息如图 5.26 所示。


图 5.26　编译报错信息截图 2

**【错误分析】**

提示 data 大小未知和"{"附近有语法错误。

### 6. 初始化时是否加'\0'

字符数组初始化时,若字符个数与数组长度相同,则字符末尾不加'\0',此时字符数组不能作为字符串处理,只能对字符逐个处理。初始化时是否加'\0',要看是否作字符串处理。

例如:

```
char b[4] = {'G', 'o', 'o', 'd'};
```

只能对字符逐个处理,不能当字符串处理。

### 7. 结构体类型声明时,漏掉了大括号后面的分号

```
#include <stdio.h>
struct node
```

```
{
    int num;
    int score1;
    int score2;
}
struct node n1,n2;
int main()
{
    n1.num = 1;
    n2.num = 2;
    printf("两个学生的学号分别为:%d,%d\n",n1.num,n2.num);
    return 0;
}
```

【编译报错信息】

编译报错信息如图 5.27 所示。

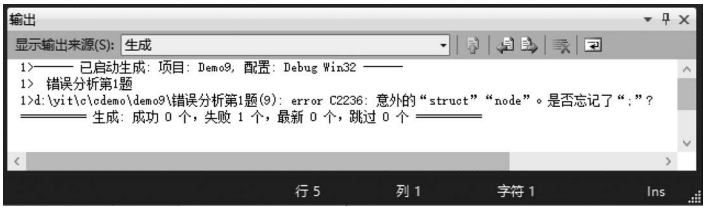


图 5.27　编译报错信息截图 3

【错误分析】

编译系统提示语句中意外的“struct”和“node”,提示后面是否忘记加“;”,程序中只要在 struct node 结构体数据类型声明的最后加上分号即可修改错误。

### 8. 混淆了结构体数据类型和结构体变量

可对结构体变量成员赋值,不能对结构体类型成员进行赋值。

错误一:

```
#include <stdio.h>
struct student
{
    int sID = 100;                          //学号
    char sSex = 'F';                        //性别
    int sMath = 90;                         //高数成绩
    int sEng = 80;                          //英语成绩
    int sC = 89;                            //C 语言程序设计成绩
}sx;
int main()
{
    printf("学号为%d的学生的英语成绩为%d",sx.sID,sx.sEng);
    return 0;
}
```

【编译报错信息】

编译报错信息如图 5.28 所示。

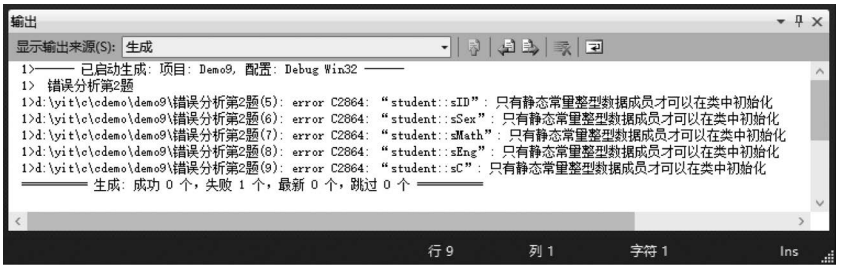


图 5.28　编译报错信息截图 4

错误二：

```
#include <stdio.h>
struct student
{
    int sID;                                    //学号
    char sSex;                                  //性别
    int sMath;                                  //高数成绩
    int sEng;                                   //英语成绩
    int sC;                                     //C 语言程序设计成绩
};
int main()
{
    student.sID = 100;
    student.sSex = 'F';
    student.sMath = 90;
    student.sEng = 80;
    student.sC = 89;
    printf("学号为 %d 的学生的英语成绩为 %d\n",student.sID,student.sEng);
    return 0;
}
```

【编译报错信息】

编译报错信息如图 5.29 所示。


图 5.29　编译报错信息截图 5

【错误分析】

上述两种赋值方法都是错误的，在 C 语言程序中，只能对结构体变量中的成员赋值，而不能对结构体数据类型中的成员赋值。

struct student 是用户自己定义的一种结构体数据类型，其用法相当于基本数据类型 int，struct student 仅是数据类型的名字，不是变量，不占存储单元。所以不能对数据类型的成员直接赋值，而应对定义的结构体变量相应成员赋值，例如：

```
#include <stdio.h>
struct student
{
    int sID;                                    //学号
    char sSex;                                  //性别
    int sMath;                                  //高数成绩
    int sEng;                                   //英语成绩
    int sC;                                     //C 语言程序设计成绩
};                                              //定义了结构体数据类型 struct student
int main()
{
    struct student stud;                        //定义了一个结构体类型的变量 stud
    stud.sID = 100;
    stud.sSex = 'F';
    stud.sMath = 90;
    stud.sEng = 80;
    stud.sC = 89;
    printf("学号为 %d 的学生的英语成绩为 %d\n", stud.sID, stud.sEng);
    return 0;
}
```

## 本章小结

数组是程序设计中最常用的构造类型。数组是一组相同类型数据的有序集合，它们都拥有同一个名字，在大批量数据处理和字符串操作时，广泛使用数组。数组按维数划分，可分为一维数组、二维数组和多维数组；数组按数据类型划分，可分为数值型数织、字符型数组、结构体数组和指针数组等。

数组中的每一个元素都属于同一种数据类型，不能把不同类型的数据放在同一个数组中。数组可以在定义时对其赋初值，称为数组的初始化。使用时，数组元素通过下标来引用，数组下标从 0 开始。

字符串应用广泛，但 C 语言中没有专门的字符串类型，字符串是存储在字符数组中的。字符数组并不要求它的最后一个字符为'\0'，但在使用字符数组存储字符串时，数组中最后一个字符一定要是'\0'，否则此时的字符数组不能作为字符串处理，调用字符串处理函数会发生程序运行错误。

结构体变量用来存储多个不同类型的数据，将它们组织成一个整体，定义时应根据实际需要先定义结构体类型，再用该结构体类型定义结构体变量和结构体数组。在实际开发中，真正的核心部分正是从这里开始的。

将数组和循环结合起来,可以有效处理大批量数据,大大提高工作效率,十分方便。

在本章中,一维数组的概念及其应用是基础,也是重点;对于多维数组,仅以二维数组作简单介绍;字符数组可以存储字符串,对字符串的概念、应用以及常用的字符串函数做了介绍;结构体是 C 语言中一种重要的数据类型,介绍了结构体类型的定义,结构体变量的定义和引用,结构体数组的定义与引用,这部分内容属于 C 语言的高级部分的内容;针对本章所学内容,应用结构体数组实现了“学生成绩管理系统”的四个功能模块:录入学生信息、显示学生信息、学生成绩统计、学生成绩排序。

## 习题五

### 一、选择题

1. 下列数组定义合法的是(　　)。

A. char a[5]="string";　　B. int a[5]={0,1,2,3,4,5};

C. char s="string";　　D. int c[]={0,1,2,3,4,5};

2. 以下对一维数组 a 进行初始化不正确的是(　　)。

A. int a[10]=(0,0,0,0);　　B. int a[10]={};

C. int a[]={0};　　D. int a[10]={10 * 2};

3. 在定义语句“int a[5][4];”之后,对数组元素的引用正确的是(　　)。

A. a[2][4]　　B. a[5][0]　　C. a[0][0]　　D. a[0,0]

4. 若有语句“int a[4]={5,3,8,9};”,其中 a[3]的值为(　　)。

A. 5　　B. 3　　C. 8　　D. 9

5. 在数组中,数组名表示(　　)。

A. 数组第 1 个元素的首地址　　B. 数组第 2 个元素的首地址

C. 数组所有元素的首地址　　D. 数组最后 1 个元素的首地址

6. 若有定义语句“char s[12] = "string";”,则语句“printf("%d\n",strlen(s));”的输出是(　　)。

A. 6　　B. 7　　C. 11　　D. 12

7. 若有以下数组声明,则数值最小的和最大的元素下标分别是(　　)。

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

A. 1,12　　B. 0,11　　C. 1,11　　D. 0,12

8. 下面程序中有错误的一行是(　　)。

```
#include <stdio.h>
int main(){
    float array[5] = {0.0};                    //第 A 行
    int i;
    for(i = 0;i < 5;i++)
```

```
        scanf("%f",&array[i]);
    for(i=1;i<5;i++)
        array[0]=array[0]+array[i];          //第 B 行
    printf("%f\n",array[0]);                 //第 C 行
    return 0;
}
```

A. 第 A 行　　B. 第 B 行　　C. 第 C 行　　D. 没有

9. 下面对二维数组声明正确的是(　　)。

A. int a[][]={1,2,3,4,5,6};　　B. int a[2][]={1,2,3,4,5,6};

C. int a[][3]={1,2,3,4,5,6};　　D. int a[2,3]={1,2,3,4,5,6};

10. 数组定义语句为“int a[3][2]={1,2,3,4,5,6};”,值为 6 的数组元素是(　　)。

A. a[3][2]　　B. a[2][1]　　C. a[1][2]　　D. a[2][3]

11. 下列语句中正确的是(　　)。

A. char a[3][ ]={'abc', '1'};　　B. char a[ ][3] ={'abc', '1'};

C. char a[3][ ]={'a', "1"};　　D. char a[ ][3] ={"a", "1"};

12. 下面程序的输出结果是(　　)。

```
#include <stdio.h>
int main(){
    char ch[3][5]={"AAAA","BBB","CC"};
    printf("\"%s\"\n",ch[1]);
    return 0;
}
```

A. "AAAA"　　B. "BBB"　　C. "BBBCC"　　D. "CC"

13. 若有以下声明和语句,则输出结果是(　　)。

```
char str[]="\"c:\\abc.dat\"";
printf("%s",str);
```

A. 字符串中有非法字符　　B. \"c:\\abc.dat\"

C. "c:\abc.dat"　　D. "c:\\abc.dat"

14. 若有以下声明和语句,则输出结果是(　　)。

```
//strlen(s)为求字符串 s 的长度的函数
char sp[]="\t\v\\\0will\n";
printf("%d",strlen(sp));
```

A. 14　　B. 3

C. 9　　D. 字符串中有非法字符

15. 判断字符串 x 是否大于字符串 y,应当使用(　　)。

A. if(x>y)　　B. if(strcmp(x,y))

C. if(strcmp(y,x)>0)　　D. if(strcmp(x,y)>0)

16. 若有以下声明语句

```
typedef struct{
    int n;
    char ch[8];
}PER;
```

则下面叙述中正确的是(　　)。

A. PER 是结构体变量名　　B. PER 是结构体类型名

C. typedef struct 是结构体类型　　D. struct 是结构体类型名

17. 以下对结构体类型变量 td 的定义中,错误的是(　　)。

A.

```
typedef struct aa{
    int n;
    float m;
}AA;
AA td;
```

B.

```
struct aa{
    int n;
    float m;
} ;
struct aa td;
```

C.

```
struct {
    int n;
    float m;
}aa;
struct aa td;
```

D.

```
struct{
    int n;
    float m;
}td;
```

18. 以下关于 typedef 叙述不正确的是(　　)。

A. 用 typedef 可以定义各种类型名,但不能定义变量

B. 用 typedef 可以增加新的类型

C. 用 typedef 只是将已经存在的类型用一个新的名字来代表

D. 使用 typedef 便于程序的通用

19. 若有定义

```
struct complex{
    int real,unreal;
}data1 = {1,8},data2;
```

则以下赋值语句中错误的是(　　)。

A. data2=data1;　　B. data2=(2,6);

C. data2.real=data1.real;　　D. data2.real=data1.unreal;

20. 已知学生记录描述为:

```
struct date{
     int year;
     int month;
     int day;
};
```

```
struct student{
      int sID;                        //学生学号
      struct date birth;              //学生生日
};
struct student s;
```

设变量 s 所代表的学生生日是“1990 年 8 月 16 日”,下列对“生日”的正确赋值是(　　)。

A. year＝1990;month＝8;day＝16;

B. birth. year＝1990;birth. month＝8;birth. day＝16;

C. s. birth. year＝1990;s. birth. month＝8;s. birth. day＝16;

D. s. year＝1990;s. month＝8;s. day＝16;

21. 下面程序的输出结果是(　　)。

```
struct s{
     int x;
     int y;
};
int main()
{
     struct s c[2] = {1,3,2,7};
     printf(" % d\n",c[0]. y/c[0]. x * c[1]. x);
     return 0 ;
}
```

A. 6　　B. 1　　C. 3　　D. 0

22. 根据下面的定义,能打印出字母 M 的语句是(　　)。

```
struct p{
     char name[10];
     int age;
};
struct p stu[6] = {"Jone",23, "Paul",22, "Mary",20, "adam",21};
```

A. printf("%c\n",stu[3]. name);

B. printf("%c\n",stu[3]. name[1]);

C. printf("%c\n",stu[2]. name[1]);

D. printf("%c\n",stu[2]. name[0]);

## 二、填空题

1. 执行语句“static int b[5], a[ ][3] ＝{1,2,3,4,5,6};”后,b[4] ＝_____,a[1][2] ＝_____。

2. 若有定义语句“static int a[3][4] ＝{{1},{2},{3}};”,则 a[1][0]的值为_____,a[1][1]的值为_____,a[2][1]的值为_____。

3. 若有定义语句“char a[ ]＝"windows",b[ ]＝"2000";”,则语句“printf("%s",strcat(a,b));”的输出结果为_____。

4. 下面程序的功能是读入 20 个整数,统计负数个数,并计算负数之和,请在下画线处填入正确的内容,使程序运行后得出正确的结果。

```
#include <stdio.h>
int  main()
{
    int i,a[20],s,count;
    ____①____;
    for(i=0;i<20;i++)
        scanf("%d",&a[i]);
    for(____②____)
    {
        if(a[i]>=0)
            ____③____;
        s+=a[i];
        ____④____;
    }
    printf("s=%d\t count=%d\n",s,count);
    return 0;
}
```

5. 下列程序的功能是:把 a 数组中的最大值放在 a[0]中,最小值放在 a[1]中,再把 a 数组元素中的次大值放在 a[2]中,次小值放在 a[3]中,以此类推。例如,若 a 数组中的数据最初排列为 1、4、2、3、9、6、5、8、7,按规则移动后,数据排列为 9、1、8、2、7、3、6、4、5。

请在下画线处填入正确的内容,使程序运行后得出正确的结果。

```
#include <stdio.h>
#include <stdio.h>
#define N 9
int main()
{
    int a[N]={1,4,2,3,9,6,5,8,7};
    int i, j, max, min, px, pn, t;
    printf("\nThe original data  :\n");
    for(i=0; i<N; i++)  printf("%4d ",____①____);
    printf("\n");
    for(i=0; i<N-1; i+=____②____)
    {
        max = min = a[i];
        px = pn = i;
        for(j=____③____; j<N; j++)
        {
            if(max<a[j])
            {  max = a[j]; px = j;  }
            if(min>a[j])
            {  min = a[j]; pn = j;  }
        }
        if(px != i)
        {   t = a[i]; a[i] = max; a[px] = t;
            if(pn == i) pn= px;
        }
```

```
            if(pn != i + 1)
            {   t = a[i + 1]; a[i + 1] = min; a[pn] = t; }
        }
        printf("\nThe data after moving  :\n");
        for(i = 0; i < N; i++)  printf(" % 4d ", a[i]);
        printf("\n");
        return 0;
    }
```

6. 下列程序的功能是：将字符数组 s 中下标为奇数的字符取出，并按 ASCII 码值大小递增排序，将排序后的字符存入字符数组 p 中，形成一个新串。请在下画线处填入正确的内容，使程序运行后得出正确的结果。

```
    # include < stdio.h >
    int main()
    {
        char s[80] = "baawrskjghzlicda", p[80];
        int i, j, n, x, t;
        printf("\nThe original string is :  % s\n",s);
        n = 0;
        for(i = 0; s[i]!= '\0'; i++)  n++;
        for(i = 1; i < n - 2; i = i + 2) {
              ____①____;
            for(j = ____②____ + 2; j < n; j = j + 2)
                if(s[t]> s[j]) t = j;
                if(t!= i)
                {  x = s[i]; s[i] = s[t]; s[t] = x; }
        }
        for(i = 1,j = 0; i < n; i = i + 2, j++)p[j] = s[i];
        p[j] = ____③____;
        printf("\nThe result is :  % s\n",p);
        return 0;
    }
```

7. 从键盘输入一个小组的学生信息，包含姓名、学号和出生年份，最后输出所有出生年份为 2002 的学生信息。

```
    # include < stdio.h >
    # define N 5
    int main()
    {
        struct stu
    {
            char name[20];
            char num[20];
            int year;
        }____①____[N];
        int i;
        for(i = 0; i < N; i++)
        {
```

```
            printf("Enter no. %d:\n", i + 1);
            scanf("%s", class1[i].name);
            scanf("%s", class1[i].num);
            scanf("%d",      ②      );
        }
        for(i = 0; i < N; i++)
        {
            if(      ③      )
            {
                printf("%s(%s)\n", class1[i].name, class1[i].num);
            }
        }
        return 0;
    }
```

## 三、编程题

1. 编程实现把一个一维数组的元素按逆序重新放置。

2. 编程实现,输入某年某月某天,求这个日期是该年的第几天(提示:首先判断所输入的年份是否是闰年,因为平年的 2 月是 28 天,闰年的 2 月是 29 天。该年的第几天=该年该月之前的各月份天数和+输入的天数)。

3. 编程实现查找数组中是否存在与给定值相同的元素,若存在,输出该元素在数组中的序号,若不存在,输出未找到。

4. 使用随机数生成函数 rand()生成 10 个 100 以内的随机整数存入一维数组,按升序排序后输出。

5. 有一个非递减有序的数组,编程实现插入一个数,要求插入该数后的数组仍然非递减有序。

6. 编程实现求如下 5×5 矩阵周边元素的平方和并输出。

$$\begin{bmatrix} 0 & 1 & 2 & 7 & 9 \\ 1 & 11 & 21 & 5 & 5 \\ 2 & 21 & 6 & 11 & 1 \\ 9 & 7 & 9 & 10 & 2 \\ 5 & 4 & 1 & 1 & 1 \end{bmatrix}$$

7. 编程实现从键盘输入一个字符串,并判断是否形成回文(即正序和逆序是一样的)。

8. 编程实现在输入的字符串中的所有数字字符前加一个 $ 字符。例如,输入 A1B23CD45,则输出为 A$1B$2$3CD$4$5。

9. 从键盘输入 10 个候选人的姓名和得票数,编程实现如下功能。

(1) 统计总票数。

(2) 打印得票数最多的候选人的姓名和得票数。

(3) 给定姓名,查询该候选人的得票数。

(4) 按得票数从高到低的顺序,打印所有候选人的姓名和得票数。

10. 在一个一维数组中存储着 N 个长方形,编程实现找出其中面积最大的长方形。

11. 定义一个描述平面上的点的结构体 Point，再定义一个描述平面上的圆的结构体 Circle，圆的圆心为 Point 类型，半径为 float 类型。编写程序，求平面上的两个圆的关系是相交、相切还是相离。

12. 编写程序，从键盘输入职工人数及每位职工的信息，包括职工号、姓名和工资，输出所有职工的平均工资，以及工资低于 2000.00 元的职工信息。