# 第 4 章　数组

## 本章导读

数组是 C 语言中一种非常重要的数据类型，属于构造类型。它是由若干个具有相同数据类型的变量按一定的存储顺序组成的，每一个变量称为一个数组元素。数组元素用数组名及下标来惟一确定。这就为我们处理大量相同类型的数据提供了方便。本章通过 C 程序实例分析着手，使读者能够掌握数组的定义及引用方法，并能够应用数组解决实际问题。本章学习的主要内容为：

- 一维数组的定义和引用
- 二维数组的定义和引用
- 字符数组与字符串的应用

## 4.1　数组应用的 C 程序实例

数组是若干具有相同数据类型且按一定存储顺序排列的一组变量。数组中的变量称为数组元素。每一个元素通过数组名和存储位置（下标）来确定。根据确定数组的一个元素所需要的下标数把数组分为一维数组、二维数组、三维数组等。二维以上的数组也称为多维数组。下面介绍两个 C 程序，说明 C 语言数组的相关知识。

【例 4.1】应用一维数组，实现从键盘输入 10 个整数，输出其中的最小数。

```
/*程序名为 l4_1.cpp*/
#include"stdio.h"
main()
{   int a[10],i,min;                /*定义一维整型数组 a 及整型变量 i 和 min，数组 a 有 10 个
                                    元素*/
    for(i=0;i<10;i++)               /*循环输入数组 a 的 10 个元素*/
   scanf("%d",&a[i]);
    min=a[0];                       /*设 a[0]元素为最小值 min 的初值*/
    for(i=1;i<10;i++)               /*逐个元素与 min 比较，找出最小值*/
    if(min>a[i])
        min=a[i];
    printf("MIN=%d\n",min);         /*输出找到的最小值 min*/
}
```

【例 4.2】应用二维数组，实现从键盘为 2×3 数组输入值，并输出数组所有元素之和。

```
/*程序名为 l4_2.cpp*/
#include"stdio.h"
```

```
main()
{   int a[2][3],i,j,sum=0;                  /*定义二维整型数组 a 及整型变量 i，j 和 sum，数
                                             组 a 有两行三列 6 个元素*/
    for(i=0;i<2;i++)                        /*按行向数组 a 输入 6 个元素，并将元素值送入 sum
                                             进行累加求和*/
      for(j=0;j<3;j++)
        {   scanf("%d",&a[i][j]);
            sum+=a[i][j];
        }
    printf("Sum=%d\n",sum);                 /*输出和值 sum*/
}
```

在【例 4.1】程序实例中，输入 10 个整型数 12，34，56，9，21，-12，34，0，-3，1 分别存放在一维数组 a 的 10 个元素中。故程序运行结果如图 4-1 所示。

在【例 4.2】程序实例中，输入 6 个整型数 23，12，45，-12，-34，35 分别存放在二维数组 a 两行三列的 6 个元素中。故程序运行结果如图 4-2 所示。


图 4-1 【例 4.1】程序输出窗口

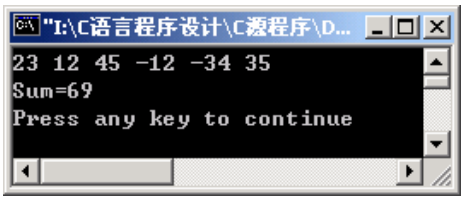


图 4-2 【例 4.2】程序输出窗口

## 4.2 一维数组的定义和引用

### 4.2.1 一维数组的定义

一维数组定义的一般格式为：

类型说明符 数组名[常量表达式];

其中，类型说明符可以是 int、char 和 float 等，指明该数组的类型，即数组中每个元素的类型；数组名的命名规则遵循标识符的命名规则，它代表数组存储时的首地址；常量表达式是指数组的长度，即数组元素的个数。

在【例 4.1】程序实例中“int a[10];”表示数组名是 a，数组元素是整型，数组有 10 个元素。故定义了一个 10 个元素的整型数组 a。

一维数组被定义后，编译系统将为该数组在内存中分配一片连续的存储空间，按一定的顺序连续存储数组的各个元素。故在前例中，数组中的 10 个元素从地址 a 处开始连续存储。因此，在定义数组时应注意，只能用常量表达式定义数组大小，常量表达式可以是常量和符号常量，但不能包含变量。即 C 不允许对数组的大小作动态定义，数组的大小不能在程序执行过程中改变。

定义数组时，应该注意以下几点：

（1）常量表达式的值必须是一个正的整数值。

（2）数组定义后，数组的长度就不能再改变。

（3）定义时，可用一个类型说明符来定义多个相同类型的数组和变量，相互之间用逗号分隔。如【例 4.1】程序实例中“int a[10],i,min;”定义了一维整型数组 a 和整型变量 i 及 min。

### 4.2.2　一维数组元素的引用

数组的使用与变量的使用类似，遵从“先定义，后使用”的原则。数组定义后就可以引用了。数组的引用是通过对数组元素的引用实现的，而不能直接引用整个数组。实际上，每一个数组元素就是一个简单变量。

一维数组的数组元素表示形式为：

数组名[下标]

下标是一个整型常量或整型表达式。一维数组元素的下标从 0 开始，如果该数组长度为 n，则元素的最大下标为 n-1。

在【例 4.1】程序实例中，“int a[10]”定义了一个具有 10 个元素的整型数组 a，则数组的 10 个元素分别为 a[0]、a[1]、a[2]、a[3]、a[4]、a[5]、a[6]、a[7]、a[8]、a[9]，通过 for(i=0;i<10;i++)scanf("%d",&a[i]);语句完成给数组元素赋值，即输入 12 34 56 9 21 –12 34 0 –3 1**<CR>**分别给 10 个元素 a[0]到 a[9]。语句 min=a[0];功能是假设 a[0]元素为最小元素，将其赋值给记录最小值的 min 变量。而语句 for(i=1;i<10;i++) if(min>a[i]) min=a[i];完成从 a[1]到 a[9]逐个元素与 min 变量比较，并将较小的元素赋值给 min 变量。最后循环结束后，min 存储的是最小元素的值。输出语句 printf("MIN=%d\n",min);执行后，输出结果为：

MIN=-12

**注意**：在一维数组引用过程中要防止下标越界问题。如“int a[10]”定义的数组 a，数组 a 中不包括 a[10]元素，下标为 10 已经越界。对于数组下标越界问题，C 语言编译系统不进行检测，即不进行错误报告，只是会造成程序运行结果的出错。

### 4.2.3　一维数组的初始化

在 C 语言中一维数组像一般变量一样，除了用赋值语句或输入语句给数组元素赋值外，还可以在定义数组的同时给数组元素赋值，称为一维数组的初始化。一般形式为：

类型说明符 数组名[常量表达式]={初始值表};

初始值表中数据与数组元素依次对应，初始值表中的数据用逗号（,）分隔。例如：

int a[5]={12,-3,4,0,367};

经过定义和初始化后，数组 a 的 5 个元素依次为 a[0]=12，a[1]=-3，a[2]=4，a[3]=0，a[4]=367。

在一维数组初始化时要注意以下几点：

（1）如果初始化时初始值表给出全部元素初值，则数组长度可以缺省。例如：

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

（2）给数组中的部分元素赋初始值。例如：

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

按照下标递增的顺序依次赋值，后两个元素系统自动赋 0 值。即 a[0]为 1，a[1]为 2，a[2]为 3，而 a[3]和 a[4]均由系统自动赋值为 0。

（3）数组中的全部元素赋初值为 0。

int a[5]={0};

此时，数组中的全部元素均为 0。

## 4.3 二维数组的定义和引用

### 4.3.1 二维数组的定义

二维数组定义的一般格式为：

类型说明符 数组名[常量表达式 1][常量表达式 2];

其中，类型说明符和数组名含义均与一维数组相同。从定义格式不难看出二维数组区别于一维数组的根本在于数组名后有两个方括号，每个方括号后均有一个常量表达式，第一个是常量表达式 1，第二个是常量表达式 2。常量表达式 1 表示行数，而常量表达式 2 表示列数。也就是它们分别指出数组的行长度和列长度。

在【例 4.2】实例中“int a[2][3];”表示数组名是 a，数组元素是整型，数组有 2 行 3 列，共计有 6（2×3）个元素。故定义了一个 2 行 3 列的整型数组 a。

与一维数组相同，二维数组的行列下标均从 0 开始。则数组 a 的 6 个元素是：

a[0][0]，a[0][1]，a[0][2]

a[1][0]，a[1][1]，a[1][2]

二维数组被定义后，编译系统将为该数组在内存中分配一片连续的存储空间，按行的顺序连续存储数组中的各个元素。即先顺序存储第一行元素，从 a[0][0]到 a[0][2]，再存储第二行的元素，从 a[1][0]到 a[1][2]。数组名 a 仍然代表数组的起始地址。

二维数组可以看做特殊的一维数组。例如二维数组 a[2][3]可看成是由两个元素 a[0]和 a[1]组成。而 a[0]和 a[1]又分别是一个一维数组，各有 3 个元素。a[0]的元素是 a[0][0]、a[0][1]、a[0][2]。a[1]的元素是 a[1][0]、a[1][1]、a[1][2]。如图 4-3 所示。

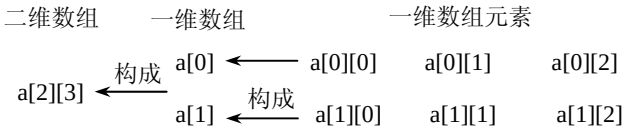


图 4-3 二维数组

### 4.3.2　二维数组元素的引用

二维数组元素的引用形式为：

数组名[行下标][列下标]

其中，行（列）下标表达式可以是整型常量、整型变量及表达式，表示二维数组的行（列）长度。下标值仍然从 0 开始，到行（列）长度减 1。

在【例 4.2】程序实例中“int a[2][3];”定义了一个 2 行 3 列的二维整型数组 a，则在内存中依次连续存储的数组元素为 a[0][0]，a[0][1]，a[0][2]，a[1][0]，a[1][1]，a[1][2]，通过循环嵌套语句：

```
for(i=0;i<2;i++)/*按行向数组 a 输入 6 个元素，并将元素值送入 sum 进行累加求和*/
   for(j=0;j<3;j++)
      {   scanf("%d",&a[i][j]);
          sum+=a[i][j];
      }
```

完成给数组元素赋值，即输入 23 12 45 -12 -34 35**<CR>**，并将输入的值依次送入 sum 变量进行累加求和。循环结束后，由输出语句 printf("Sum=%d\n",sum);完成累加和 sum 的输出，结果为：

```
Sum=69
```

### 4.3.3　二维数组的初始化

定义二维数组时，给数组元素赋初值称为二维数组初始化。二维数组初始化时要特别注意二维数组的各元素所赋初始值的排列顺序。初始值的排列顺序必须与数组各元素在内存的存储顺序完全一致。具体方法如下：

（1）分行给二维数组赋初值。例如：

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

此方法直观，很明显第 1 行的元素的初值由第一个花括号数据提供，第二行的元素的初值由第二个花括号数据提供，按行赋初值。

（2）按数组排列顺序对各元素赋初值。例如：

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

此方法与前一种方法效果相同，但数据所处的行列位置不直观，尤其是数据多时，数据存储所在的行列需要仔细定位，容易出现错误。

（3）对部分元素赋初值。例如：

```
int a[2][3]={{1},{0,4}};
```

此方法是对数组中各行的部分元素赋初值，其余元素值自动为 0。本例的作用是只对 0 行 0 列元素赋初值 1，对 1 行 1 列元素赋初值 4，而其余元素值自动为 0。即赋值后数组 a 的各元素为：

```
1    0    0
```

0 4 0

（4）赋初值时，有些情况可缺省第一维长度，但第二维长度不能缺省。

第一种情况：数组的全部元素都赋初值时，则定义数组时对第一维长度可以缺省。例如：

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

此时，系统会根据元素总个数分配存储空间，一共 6 个元素，每行 3 列，自然确定为 2 行。

第二种情况：在分行赋值时（含对部分元素赋初值情况），可以在定义时省略第一维的长度。例如：

int a[ ][3]={{0},{0,3}};

此时，编译系统知道数组共有两行。赋值后数组 a 的元素分别为：

0 0 0

0 3 0

## 4.4 字符数组与字符串

字符数组是用来存放字符数据的数组，即数组的数据类型是字符型（char）的数组称为字符数组。字符数组的每个元素存放一个字符。在 2.2 节中介绍了字符串常量，实际上字符串的存放和处理是通过字符数组实现的。

### 4.4.1 字符数组的定义

字符数组的定义形式为：

一维字符数组定义：char 数组名[常量表达式];

二维字符数组定义：char 数组名[常量表达式 1][常量表达式 2];

可见，定义方法与前面介绍的类似。例如：

```
char c[10];          /*定义了一个具有 10 个元素的一维字符数组 c*/
char c[2][10];       /*定义了一个两行 10 列的二维字符数组 c*/
```

由于字符型与整型是互相通用的，因此上面的定义也可改为：

```
int c[10];
int c[2][10];
```

但是，此种定义方法会浪费存储空间。

### 4.4.2 字符数组的初始化

在定义字符数组时，可以对字符数组初始化。字符数组初始化有以下几种方法：

（1）用字符常量初始化数组。

1）一维字符数组初始化，例如：

```
char c[10]={'a','b','c','d','e','f','g','h','i','j'};
```

字符数组 c 定义后，每个元素赋值为：c[0]='a'，c[1]='b'，c[2]='c'，c[3]='d'，c[4]='e'，c[5]='f'，c[6]='g'，c[7]='h'，c[8]='i'，c[9]='j'

2）二维字符数组初始化，例如：

char c[2][10]={{'a','b','c'},{'d','e', ' '}};

字符数组 c 定义后，数组元素赋值为：

'a'　'b'　'c'　'\0'　'\0'　'\0'　'\0'　'\0'　'\0'　'\0'

'd'　'e'　' '　'\0'　'\0'　'\0'　'\0'　'\0'　'\0'　'\0'

3）当初值个数与字符数组长度相同，在定义时可以省略数组长度，系统会自动根据初值个数确定数组长度。例如：

char c[ ]={'H','o','w',',','a','r','e',',','y','o','u'};

字符数组 c 的长度由定义时元素初始值个数确定，即该初始化语句相当于：

char c[11]={'H','o','w',' ','a','r','e',' ','y','o','u'};

4）用整型常量初始化字符数组。例如：

char c[5]={'H',111,119};

由于整型数据与字符型数据有通用性，所以 c[1]用整型 111 初始化，c[2]用整型 119 初始化，实际上 c[1]和 c[2]分别为'o'和'w'字符。

（2）用字符串常量初始化数组。字符串是用双引号括起来的字符序列，在 C 语言中，字符串是利用字符数组来存放和处理的。用字符串常量初始化一个字符数组，要比用单个字符常量初始化字符数组简单得多。例如：

char c[ ]={"How are you"};

其中，左右花括号可以省略，即可写成：char c[ ]="How are you";

经过初始化后，数组 c 共有 12 个元素，它们分别为：

c[0]='H'，c[1]='o'，c[2]='w'，c[3]=' '，c[4]='a'，c[5]='r'，c[6]='e'，c[7]=' '，c[8]='y'，c[9]='o'，c[10]='u'，c[11]='\0'

二维数组初始化时，也可以使用字符串进行初始化。例如：

char c[ ][ 8]={ "white","black"};

此时，数组 c 分解成两个一维数组 c[0]和 c[1]，它们各自有 10 个元素，每一个以一个字符串初始化，如图 4-4 所示。

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| c[0] | w | h | i | t | e | \0 | \0 | \0 |
| c[1] | b | l | a | c | k | \0 | \0 | \0 |

图 4-4　二维数组结构

### 4.4.3　字符数组的引用

字符数组的引用也是通过对数组逐个元素引用实现的。引用数组的元素可以得到一个字符。

【例 4.3】应用一维字符数组输入一个字符串。

```
/*程序名为 l4_3.cpp*/
#include "stdio.h"
main()
{   char c[ ]="How are you!";          /*定义一维数组 c 有 13 个元素*/
    int i;
    for(i=0;i<13;i++)                  /*通过循环控制输出数组每个元素*/
  printf("%c",c[i]);
    printf("\n");
}
```

运行结果：

```
How are you!
```

### 4.4.4 字符数组的输入输出

字符数组的输入输出可以通过 scanf()或 getchar()对字符数组进行赋值，通过 printf()或 putchar()对字符数组进行输出，在 1.4 节中作了介绍。在利用 scanf()和 printf()完成输入输出时，%c 格式是逐个字符元素进行输入和输出，而%s 格式是对整个字符串一次完成输入和输出。

用格式说明符%s 进行整串输入和输出时应注意以下问题：

（1）在 scanf 中使用格式说明符%s 实现整串的输入。例如：

```
char   c[20];
scanf("%s"，c);
```

执行上面的语句，如果输入：abcd**<CR>**

则这个字符串将从数组 c 的起始地址（&c[0]或数组名 c）开始依次放入数组 c 中，数组剩余空间补‘\0’。

（2）用%s 格式符输入字符串时，空格、Tab 符和回车符只能作为分隔符而不能输入到数组中。例如：

如果输入字符串为：How are you!

则只有字符串“How”存入到数组中，其余被截掉。

（3）当输入项为数组元素的地址时，输入数据将从这一元素开始存放。

（4）输入字符串时，应避免发生越界。

在 printf()中使用格式说明符%s 可以实现整串的输出。其调用形式如下：

```
printf("%s",c);
```

这里，c 是存储单元的首地址。调用这个函数时，将从 c 地址开始输出存储单元中的字符，直到遇到第一个‘\0’为止。输出结束后不自动换行。

【例 4.4】应用一维字符数组输入一个字符串。

```
/*程序名为 l4_4.cpp*/
#include "stdio.h"
main()
{   char c[6];
    int i;
```

```
    for(i=0;i<5;i++)                    /*利用 getchar()给数组 c 输入字符*/
    c[i]=getchar();
    c[i]='\0';                          /*将字符串结束标志'\0'赋值给数组 c 的 c[5]元素*/
    printf("%s",c);                     /*输出字符串*/
}
```

### 4.4.5　字符串处理函数

C 语言编译系统提供了大量处理字符串的库函数，方便了字符串问题的处理，下面介绍几种常用的函数。应用字符串输入函数 gets()和输出函数 puts()时，需要使用#include 命令将 stdio.h 头文件包含到源文件中。而其他的字符串处理函数在使用时，需要用#include 命令将 string.h 头文件包含到源文件中。

1. 字符串输入函数 gets()

调用 gets()函数实现字符串的输入，其调用形式为：

gets(字符数组);

其作用是从终端输入一个字符串（包括空格）赋给从字符数组起始的存储单元中，直到读入一个回车符为止。回车符读入后，不作为字符串的内容，系统将自动用‘\0’替换，作为字符串结束的标志。例如：

```
char   c[20];
gets(c);
```

执行上面的语句，如果输入：How are you!<**CR**>

则将读入的 12 个字符依次存入到 c[0]开始的存储单元中，并在其后自动加入一个字符串结束标志‘\0’。

2. 字符串输出函数 puts()

调用 puts()函数实现字符串的输出，其调用形式为：

puts(字符数组);

其作用是将字符数组起始地址开始的一个字符串（以‘\0’结束的字符序列）输出到终端，并将字符串结束标志‘\0’转化成‘\n’，自动输出一个换行符。例如：

```
char c[ ]= "How\nare\nyou!";
puts(c);
```

输出结果：

```
How
are
you!
```

3. 字符串长度函数 strlen()

调用 strlen()函数实现字符串长度的测试，其调用形式为：

strlen(字符数组或字符串);

其作用是测试字符数组起始地址开始的字符串（以‘\0’结束的字符序列）有效长度。函数值为字符数组或字符串的有效字符个数，不包括‘\0’在内。例如：

```
char c[20 ]= "How\nare\nyou!";
printf("%d\n",strlen(c));
```

输出结果：12

4. 字符串连接函数 strcat()

调用 strcat()函数实现两个字符串的连接，其调用形式为：

strcat(字符数组 1,字符数组 2 或字符串);

其作用是将字符数组 2（字符串）连接到字符数组 1 的后面，函数值为字符数组 1 的地址。例如：

```
char c1[30 ]= "How are you!\n";
char c1[ ]= "I am fine!";
printf("%s ",strcat(c1,c2));
```

输出结果：

How are you!

I am fine!

在使用 strcat()函数时，需要注意以下问题：

（1）连接前字符数组 1 和字符数组 2 的尾部都有一个‘\0’，连接时将字符数组 1 后的‘\0’自动取消，字符数组 2 后的‘\0’一并连接到字符数组 1 后。

（2）字符数组 1 必须有足够的长度，以便在其有效字符后能够容纳下字符数组 2 中的字符串。

5. 字符串复制函数 strcpy()

调用 strcpy()函数实现字符串的复制，其调用形式为：

strcyp(字符数组 1,字符数组 2 或字符串);

其作用是将字符数组 2（字符串）复制到字符数组 1 中去。函数值为字符数组 1 的起始地址。例如：

```
char c1[30 ],c2="How are you!\n";
printf("%s ",strcpy(c1,c2));
```

输出结果：How are you!

在使用 strcpy()函数时，需要注意以下问题：

（1）字符数组 1 的长度必须大于或等于字符数组 2 的长度，以便容纳字符数组 2 中的字符串。

（2）复制时将字符数组 2 中字符串的结束标志‘\0’一起复制到字符数组 1 中。

（3）strcpy()函数能够实现将字符数组 2 中前面若干个字符复制到字符数组 1 中的功能。例如：

```
strcpy(c1,c2,4);
```

该语句的作用是将 c2 中前面的 4 个字符复制到 c1 中去，然后系统自动添入一个字符串结束标志‘\0’。

6. 字符串比较函数 strcmp()

调用 strcmp()函数实现字符串的大小比较，其调用形式为：

strcmp(字符数组 1 或字符串 1,字符数组 2 或字符串 2);

其作用是将两个字符数组（字符串）自左向右对应的字符逐个进行比较（按 ASCII 码值大小比较），直到出现不同字符或遇到‘\0’字符为止，函数值为一个整型数。

当字符串中的对应字符全部相等且同时遇到‘\0’字符时，则两个字符串相等，否则，以第一个不相同的字符的比较结果作为整个字符串的比较结果，比较结果由函数值带回，具体情况如表 4-1 所示。

表 4-1　字符串比较 strcmp()函数值情况

| 字符串大小情况 | 函数值 | |
|---|---|---|
| | Visual C++ 6.0 函数值 | Turbo C 2.0 函数值 |
| 字符串 1 等于字符串 2 | 0 | 0 |
| 字符串 1 大于字符串 2 | 1 | 首个不同字符的 ASCII 码差值（正整数） |
| 字符串 1 小于字符串 2 | -1 | 首个不同字符的 ASCII 码差值（负整数） |

## 4.5　综合实训

【例 4.5】输入 10 名学生的成绩，求出平均分，并输出高于平均分的学生成绩。

```
/*程序名为 l4_5.cpp*/
#include "stdio.h"
main()
{   int i;
    float score[10],aver=0.0;
    printf("Please input scores of 10 students:");
    for(i=0;i<10;i++)                          /*输入 10 名学生成绩并累加和*/
      {   scanf("%f",&score[i]);
        aver+=score[i];
      }
      aver/=10;                                /*求出 10 名学生的平均成绩*/
      printf("The average score is:%.2f\n",aver);
      printf("They are:");
      for(i=0;i<10;i++)                        /*输出高于平均成绩的学生成绩*/
          if(score[i]>aver)
              printf("%6.2f",score[i]);
}
```

运行结果如图 4-5 所示。

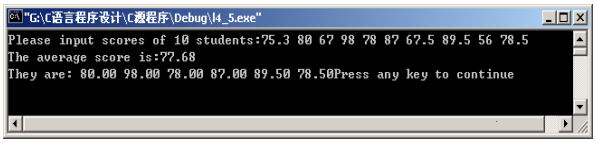


图 4-5　【例 4.5】运行结果

【例 4.6】用冒泡法对 10 个整型数按升序进行排序。

冒泡法是使较小的值像空气泡一样逐渐“上浮”到数组的顶部，而较大的值逐渐下沉到数组的底部。具体思路是：从第一个数开始将相邻的两个数比较，较大的数向后移动，较小的数向“上浮”一个，经过一轮的比较，最大的数移动到末尾。对剩下的数继续下一轮的比较和移动。如果 n 个数比较，这样 n-1 轮后，就完成了排序工作。例如，对 5 个数据（9,8,5,3,0）进行排序，排序过程如图 4-6 所示。

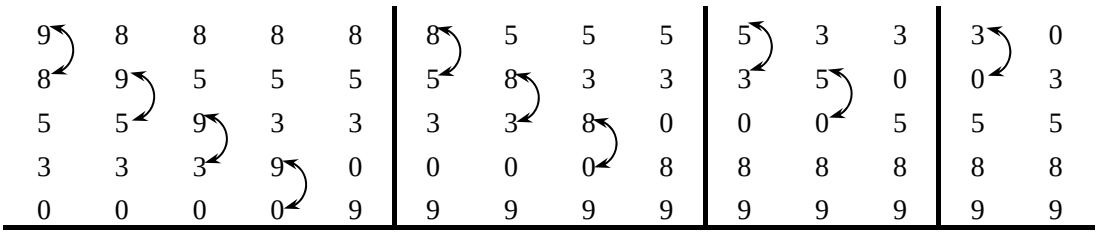


图 4-6　排序过程

```
/*程序名为 l4_6.cpp*/
#include "stdio.h"
main()
{   int i,j,t,a[10];
    printf("Please input 10 numbers:\n");
    for(i=0;i<10;i++)                  /*输入 10 个整数存入数组 a 中*/
   scanf("%d",&a[i]);
    for(i=0;i<9;i++)                   /*对数组 a 中的 10 个整数排序*/
   for(j=0;j<9-i;j++)
        if(a[j]>a[j+1])                /*前面的元素大于后面的元素则交换*/
        {   t=a[j];
            a[j]=a[j+1];
             a[j+1]=t;
        }
printf("The sorted numbers are:");
for(i=0;i<10;i++)                      /*输出数组 a 中的 10 个元素*/
    printf("%d ",a[i]);
printf("\n");
}
```

运行时输入：89 67 45 87 32 1 0 45 2 3<**CR**>，则结果如图 4-7 所示。

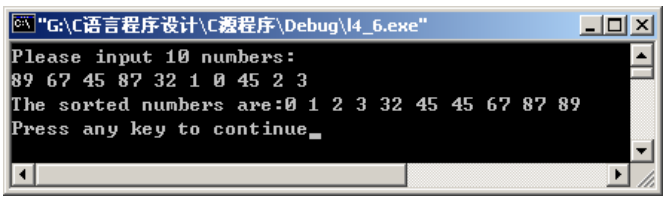


图 4-7 【例 4.6】运行结果

【例 4.7】用选择法对 10 个整型数按升序进行排序。

选择法的思路是：从第一个元素开始逐个元素进行比较，并记录下较小元素的下标，经过一轮的比较和记录后，记录的下标为最小元素的下标，将该元素与第一个元素交换。对剩下的数继续下一轮的比较和记录。如果 n 个数比较，这样 n-1 轮后，就完成了排序工作。例如，对 5 个数据（9,4,5,3,0）进行排序，排序过程如图 4-8 所示。

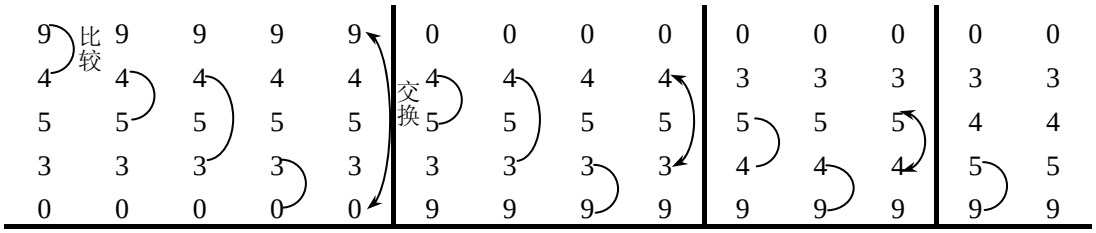


图 4-8　排序过程

```
/*程序名为 l4_7.cpp*/
#include "stdio.h"
main()
{   int i,k,j,t,a[10];
    printf("Please input 10 numbers:\n");
    for(i=0;i<10;i++)                    /*输入 10 个整数存入数组 a 中*/
   scanf("%d",&a[i]);
    for(i=0;i<9;i++)                     /*对数组 a 中的 10 个整数排序*/
{   k=i;
    for(j=i+1;j<10;j++)
        if(a[k]>a[j])
            k=j;
    t=a[k];                              /*本轮最小的元素与本轮首个元素交换*/
    a[k]=a[i];
    a[i]=t;
}
printf("The sorted numbers are:");
for(i=0;i<10;i++)                        /*输出排序后数组 a 中的 10 个元素*/
    printf("%d ",a[i]);
printf("\n");
}
```

运行时输入：89 67 45 87 32 1 0 45 2 3<**CR**>，则结果如图 4-7 所示。

【例 4.8】应用二维数组实现简单的学生成绩查询系统的设计。具体要求如下：

（1）根据用户输入的学号，能够给出该生各科成绩及平均分。

（2）根据用户输入的课程代号，能够给出该课程中每位学生的成绩及课程平均分。

（3）能够查询出某个学生某一门的考试成绩。

程序分析：学生成绩用二维数组存储，各行代表各位学生的信息，各列代表学生学号

和各门课程代号的信息。程序依次有 5 项功能：学生信息的输入；输入学号，输出该生各科成绩及平均分；输入课程代号，输出该课程每位学生成绩及课程平均分；输入学生学号及课程代号，输出成绩；设计功能选择菜单及利用 switch-case 语句实现以上功能。

```
/*程序名为 l4_8.cpp*/
#include "stdio.h"
#define N 4                                    /*定义符号常量 N 等于 4*/
#define M 5                                    /*定义符号常量 M 等于 5*/
main()
{   int select;
    int score[N][M],i,j,aver,num;
    while (1)                                  /*菜单循环显示和功能循环处理*/
    {   printf("              欢迎使用学生查询系统!\n\n");/*菜单显示*/
       printf("* * * * * * * * * * * * * * * * * * * * * * *\n");
        printf("*                  请选择查询对象:                  *\n");
        printf("* 1.据学生学号查询学生成绩及平均成绩          *\n");
       printf("* 2.据课程代号查询学生课程成绩及平均成绩    *\n");
        printf("* 3.据学生学号和课程代号查询学生成绩          *\n");
        printf("* 4.学生原始数据的依次输入                    *\n");
        printf("* 5.谢谢使用，退出本系统!                     *\n");
        INPUT:
        printf("* * * * * * * * * * * * * * * * * * * * * * *\n");
        printf("              请选择功能号 1—5: ");
        scanf("%d",&select);
        if(select==5)          /*退出系统*/
            break;
        if(select>=6||select<=0)                /*输入功能号有误处理*/
       {   printf("* * *  输入有误，请重新功能号 1-5：* * *\n");
            goto INPUT;
       }
       aver=0;
        switch(select)
       {   case 1:                              /*按学号查询功能*/
            printf("请输入查询学生的学号\n");
           scanf("%d",&num);
           for(i=0;i<N;i++)
             if(num==score[i][0])
                 for(j=1;j<M;j++)
                 {   printf("%d 号课程成绩是%d\n",j,score[i][j]);
                     aver+=score[i][j];
                 }
           printf("%d 号学生的平均成绩为%d\n",num,aver/(M-1));
           break;
            case 2:                             /*按课程查询功能*/
           printf("请输入查询课程的代号(1-%d)：\n",M-1);
           scanf("%d",&j);
           for(i=0;i<N;i++)
```

```
        {   printf("%d 号学生的成绩为%d\n",i+1,score[i][j]);
            aver+=score[i][j];
        }
        printf("%d 号课程的平均成绩为%d\n",j,aver/N);
        break;
        case 3:                                    /*按学号和课程查询功能*/
        printf("请输入查询的学生学号和课程代号：\n");
        scanf("%d%d",&num,&j);
      for(i=0;i<N;i++)
          if(num==score[i][0])
              printf("%d 号学生%d 号课程成绩为%d\n",num,j,score[i][j]);
        break;
        case 4:                                    /*输入学生原始数据*/
        printf("请依次输入%d 个学生的%d 门成绩(学号、各门成绩）\n",N,M-1);
        for(i=0;i<N;i++)
            for(j=0;j<M;j++)
                scanf("%d",&score[i][j]);
        break;
        default:break;
      }
    }
}
```

## 本章小结

数组是若干具有相同数据类型且按一定存储顺序排列的一组变量。数组中的变量称为数组元素。数组分为一维数组、二维数组、三维数组等，二维以上的数组也称为多维数组。

**一维数组**

一维数组定义的一般格式为：

类型说明符　数组名[常量表达式];

其中，“常量表达式”表示数组的大小，即该数组元素个数。

一维数组的数组元素引用形式为：

数组名［下标］一个整型常量或整型表达式

一维数组元素的下标从 0 开始，如果该数组长度为 n，则元素的最大下标为 n-1。数组定义后，在内存中按数组元素下标从小到大的次序，连续为数组各个元素分配相应数据类型的存储空间。数组名是数组存储的起始地址。

**二维数组**

二维数组定义的一般格式为：

类型说明符　数组名[常量表达式 1][常量表达式 2];

其中，“常量表达式 1”和“常量表达式 2”分别表示数组的行和列的大小，即该数组元素的行数和列数。二维数组的数组元素按行存储。

二维数组元素的引用形式为：

数组名[行下标][列下标]

其中，行（列）下标表达式可以是整型常量、整型变量及表达式，表示二维数组的行（列）长度。下标值仍然从 0 开始，到行（列）长度减 1。

数组名代表数组的起始地址。数组的引用是通过对数组元素的引用实现的，而不能直接引用整个数组。

**字符数组**

字符数组是用来存放字符数据的数组，即数组的数据类型是字符型（char）的数组称为字符数组，字符数组的每个元素存放一个字符。字符串的存放和处理是通过字符数组实现的，可以利用 C 提供的字符串处理函数 gets()、puts()、strlen()、strcat()、strcmp()、strcpy()等方便快捷地实现字符串处理。

# 习题四

## 一、思考题

1．输入 10 个同学的成绩，统计 90 分以上和不及格的人数，并输出平均成绩。

2．输入 10 个整数，查找出最大数和最小数并指出它们所处的输入位置。

3．编程实现将输入的字符串逆序存放。

4．用插入排序，向 N 个已经从小到大排顺的整数中，输入一个新数插入到适当的位置，使 N+1 个整数仍然有序。

5．编程计算一个 3×3 矩阵的主对角线元素之和。

6．编程将一个 2×3 矩阵转置后输出。

## 二、选择题

1．下列数组定义中，正确的是（　　）。

（A）#define N 8
　　float a[N];

（B）int n; scanf("%d",&n);float a[n];

（C）int n=10,a[n];

（D）int a(10);

2．下列一维数组初始化语句中，正确的是（　　）。

（A）int a[5]={,2,3, ,5};

（B）int a[5]={};

（C）int a[5]={5*2};

（D）int a[5]={1,2,3,4,5,6};

3．下列正确的程序段是（　　）。

（A）int i,a[5];for(i=0;i<5;i++) a[i]=(i+1)*10;

（B）int a[5];a={10,20,30,40,50};

（C）int a[5];a[1]=10;a[2]=20;a[3]=30;a[4]=40;a[5]=50;

（D）int a[5];a[5]={10,20,30,40,50};

4．若有说明 int a[3][4];则对其数组元素的正确引用是（　　）。

（A）a[2][1+2]　　（B）a(2)(3)　　（C）a[2,3]　　（D）a[3][4]

5．下列二维数组初始化语句中，正确的是（ ）。

（A）int a[2][3 ]={{0,1},{2,3},{4,5}};

（B）float a[3][ ]={1,2,3,4,5};

（C）int a[ ][3]={1,2,3,4,5};

（D）int a[2][3]={(0,1),(2,3)};

6．有以下定义：

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

char c1='a',c2='1';

则数值不为 3 的表达式是（　　）。

（A）a[2]　　（B）'d'-c1　　（C）a['4'-c2]　　（D）c2+2

7．有以下定义：

char x[ ]="abcedfg";

char y[ ]={'a','b','c','d','e','f'};

则正确的叙述是（　　）。

（A）数组 x 和数组 y 等价　　（B）数组 x 和数组 y 的长度相同

（C）数组 x 的长度大于数组 y 的长度　　（D）数组 x 的长度小于数组 y 的长度

8．下面各语句行中，能正确进行赋字符串操作的语句行是（　　）。

（A）char st[4][5]={"ABCDE"};

（B）char s[5]={'A','B','C','D','E'};

（C）char s[5];scanf{"%s",s};

（D）char s[5];s="ABCD";

9．下面程序的输出结果是（　　）。

```
#include "stdio.h"
main()
{ char s1[10],s2[10],s3[10],s4[10];
   scanf("%s%s",s1,s2);gets(s3);gets(s4);
   puts(s1);puts(s2);puts(s3);puts(s4);
}
```

输入以下数据：

```
aaaa bbbb<CR>
cccc dddd<CR>
```

（A）aaaa　　（B）aaaa　　（C）aaaa　　（D）aaaa bbbb

bbbb　　bbbb　　bbbb　　cccc

cccc cccc dddd dddd
cccc dddd dddd cccc

10．请选择以下程序段的输出结果（　　）。

```
#include "stdio.h"
main()
{ int i=0;
  char s[ ]="ABCD";
  for(;i<4;i++)
  printf("%s\n",s+i);
}
```

（A）ABCD　　（B）A　　（C）D　　（D）ABCD
BCD　　B　　C　　ABC
CD　　C　　B　　AB
D　　D　　A　　A

11．下列叙述中错误的是（　　）。

（A）字符数组中的字符串可以整体输入、输出

（B）字符数组可以存放字符串

（C）可以在赋值语句中通过赋值运算符“=”对字符数组整体赋值

（D）不可以用关系运算符对字符数组中的字符串进行比较

## 三、填空题

1．有以下定义：

float b[20];

则数组 b 共有_____个元素，其中第一个元素为_____，最后一个元素为_____。数组 b 的起始地址为_____，数组在内存中连续占用_____个存储单元。

2．有以下定义：

int b[3][5];

则数组 b 共有_____个元素，其中第一个元素为_____，最后一个元素为_____。数组 b 列下标的上限为_____，下限为_____。数组元素存储时按_____次序存储。数组 b 的起始地址为_____，整个数组在内存中连续占用_____个存储单元。

3．以下程序的执行结果是_____。

```
#include "stdio.h"
main()
{   int i;
    int a[ ]={10,20,30,40,50};
    for(i=4;i>=0;i--)
    printf("%d ",a[i]);
}
```

4．以下程序的执行结果是_____。

```
#include "stdio.h"
```

```
main()
{   int i;
    int a[3][2]={10,20,30,40,50};
    for(i=0;i<2;i++)
    printf("%d ",a[2-i][1-i]);
}
```

5．以下程序的执行结果是_____。

```
#include "stdio.h"
#include "string.h"
main()
{   char str[10]={'H','a','p','p','y','!'};
    printf("%d    %d",strlen(str),strlen(str+2));
}
```

6．以下程序执行时输入 Good morning !<CR>，则运行结果是_____。

```
#include "stdio.h"
main()
{   char str[20];
    scanf("%s",str);
    printf("str=%s\n",str);
}
```

7．以下程序执行时输入 Good morning !<CR>，则运行结果是_____。

```
#include "stdio.h"
main()
{   char str[20];
    gets(str);
    printf("str=%s\n",str);
}
```

8．以下程序对数组中的值进行排序，请填空。

```
#include "stdio.h"
#define M 10
main()
{   int a[M],i,j,k;
    for(i=0;i<M;i++)
    scanf("%d",a+i);
    for(k=0;_____;k++)
    for(_____;i<M-k-1;i++)
            if(a[i]<a[i+1])
            {j=a[i];
             _____;
              _____;
            }
    for(i=0;i<M;i++)
    printf((i%4)?"%4d":"\n%4d",a[i]);
    printf("\n");
}
```