第5章　数　　组

【学习目标】

- 掌握数组的定义、初始化、数组元素的引用。
- 掌握二维数组的定义、初始化和引用。
- 掌握字符数组的定义、初始化和引用。

数组是一组具有相同类型的有序数据的集合。数组中的每个数据称为数组元素。

数组可以分为一维数组和多维数组，常用的多维数组为二维数组。使用数组使得在利用计算机程序解决某些问题时变得更加方便、灵活，程序更具可读性。

5.1　一维数组

1. 一维数组的定义

一维数组也称为向量。在实际应用中需要按照某种顺序对一组相同类型数据进行操作的场合很多。比如要处理一个班级学生的年龄，当人数较少时，可以将每个学生的年龄用一个整型变量来表示，如 x、y、z 等，但当人数较多时，这样使用起来会很不方便，而且很容易出错。如果通过数组来处理这些数据，就会很方便。

在 C++ 中，数组和普通变量一样，在使用之前必须加以明确的定义说明，以便编译程序在内存中给它们分配空间。定义一维数组的一般形式为

```
类型说明符 数组名[常量表达式];
```

其中，数组名取名规则和变量名相同，遵循标识符命名规则；常量表达式的值指定了该数组中数组元素的个数，即指定了数组长度；类型说明符指定该数组所有元素的数据类型。例如：

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

定义了一个一维数组 a，数组中有 5 个元素，每个元素均为整型。5 个元素分别依次记为 a[0]、a[1]、a[2]、a[3]、a[4]。在内存中开辟了 5 个连续的存储单元用来存放这 5 个元素。

在定义数组时，需要注意以下几点。

(1) 数组名不能与其他变量名重名。数组名后是用方括号括起来的常量表达式，不能用圆括号。

(2) C++ 语言的数组元素下标从 0 开始。数组 a 的 5 个元素为 a[0]、a[1]、a[2]、a[3]、a[4]，而不包含 a[5]。在 C++ 语言中，编译时不对数组做边界检查，如果程序中出现了下标越界，可能会造成程序运行结果的错误。因此要注意下标不能过界。

(3) C++ 语言不允许对数组的长度做动态定义，即数组长度不能是变量。例如：

```
#include <iostream.h>
```

```
void main()
{  int n=5;
   int a[n];
   …
}
```

是错误的。

(4) 可以一次同时定义多个同类型数组。例如：

```
int a[5],b[10];
```

2. 一维数组的引用

一个数组一旦被定义，编译程序便在内存中为之分配一段连续的内存空间，数组元素按照次序存放在里面，比如前面定义的整型数组 a 在内存中的存储情况如图 5-1 所示。

数组元素	存储区
a[0]	2
a[1]	3
a[2]	4
a[3]	5
a[4]	6

图 5-1　数组 a 在内存中的存储情况

数组一经定义，其元素就可以被引用。数组元素通常也称为下标变量，可以参加各种运算，这与简单变量的使用是一样的，其标识方法为数组名后跟一个下标。下标表示了元素在数组中的顺序号。

数组元素的一般表示形式为

```
数组名[下标]
```

其中，下标只能为整型常量或整型表达式。例如，a[3]、a[i+j]、a[2 * 2]等都是合法的数组元素。

数组必须先定义，后使用。在 C++ 中，只能逐个引用数组元素而不能一次引用整个数组。例如，输出有 5 个元素的数组必须使用循环语句逐个输出各数组元素：

```
for(i=0; i<5; i++)  cout<<a[i];
```

而不能用一条语句输出整个数组，下面的写法是错误的：

```
cout<<a;
```

【案例 5-1】 从键盘输入 10 个整数，找出其中的最大数并输出。

程序如下：

```
#include <iostream.h>
void main()
{  int a[10],i,max;
   for(i=0; i<10; i++)
   cin>>a[i];
   max=a[0];
   for(i=1; i<10; i++)
   if(max<a[i])max=a[i];
   cout<<"MAX="<<max;
}
```

运行结果如下：

```
1  2  11  22  6  4  40  16  90  55↙
MAX=90
```

3. 一维数组的初始化

为了使用上的方便，人们常常希望使数组具有初值。与变量的初始化一样，C++ 语言允许在定义数组的同时对数组元素赋以初值，这称为数组的初始化。初始化是在编译时进行的，故不占用运行时间。

下面介绍数组的几种初始化方法。

(1) 将数组全部的元素进行赋初值。例如：

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

即在定义数组时用一对花括号将要赋给数组各元素的值括起来，各值之间用逗号间隔，按其顺序赋给该数组。经定义和初始化后，a[0]=1，a[1]=2，a[2]=3，a[3]=4，a[4]=5。

使用时注意：既使各数组元素的值全部相等，也必须逐个赋值，而不允许给数组整体赋初值。例如，想让整型数组 a[5]的 5 个元素全都为 1，初始化时应写成

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

不能写成

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

(2) 将数组部分的元素进行赋初值，未赋值元素自动取 0 值。例如：

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

经定义和初始化后，a[0]=1，a[1]=3，a[2]=5，a[3]=0，a[4]=0。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，注意数组长度不可以缺省。当初始值数据缺少前几个或中间的某个数据时，相应的逗号不能省略，默认的数据自动为 0。

(3) 对数组的全部元素赋初值时，也可以不指定数组长度。例如：

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

可写为

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

系统会根据{}中的数值个数，自动定义 a 数组长度为 5。

4. 一维数组程序设计举例

【案例 5-2】 从键盘输入 10 个学生的成绩，求出他们的平均分。

程序如下：

```
#include <iostream.h>
void main()
{
   int i;
```

```
    float stu_score[10];
    float sum=0,aver=0.0;
    for(i=0;i<10;i++)
    {  cin>>stu_score[i];
       sum=sum+stu_score[i];
    }
    aver=sum/10.0;
    cout<<"平均分为"<<aver;
}
```

运行结果：

```
82 78 67 95 79 89 91 75 66 86↙
平均值为      80.80
```

【案例 5-3】 用冒泡排序法对 10 个整数排序(由小到大)。

算法提示：冒泡排序法是一种常用的排序方法，将相邻两个数比较，将小数调到前头。设有 n 个数要求从小到大排列，冒泡排序法的算法分为如下 $n-1$ 个步骤。

第 1 步：由上向下，相邻两数比较，将小数调到前头。反复执行 $n-1$ 次，第 n 个数最大，即大数沉底。

第 2 步：由上向下，相邻两数比较，将小数调到前头。反复执行 $n-2$ 次，后 2 个数排好。

⋮

第 k 步：由上向下，相邻两数比较，将小数调到前头。反复执行 $n-k$ 次，后 k 个数排好。

⋮

第 $n-1$ 步：由上向下，相邻两数比较，将小数调到前头。执行 1 次，排序结束。

例如，“56,40,43,25,18”的排序过程示意如下，其中加下划线的数字是已排好的，不用再比较。

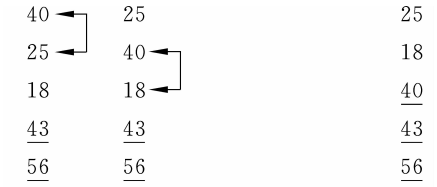
第 1 步：比较 4 次。

56 ←	40	40	40	40
40 ←	56 ←	43	43	43
43	43 ←	56 ←	25	25
25	25	25 ←	56 ←	18
18	18	18	18 ←	56
第 1 次	第 2 次	第 3 次	第 4 次	结果

第 2 步：比较 3 次。

40 ←	40	40	40
43 ←	43 ←	25	25
25	25 ←	43 ←	18
18	18	18 ←	43
56	56	56	56
第 1 次	第 2 次	第 3 次	结果

第 3 步：比较 2 次。



第 4 步：比较 1 次。

可以看出，如果有 n 个数，则要进行 $n-1$ 步比较。在第 i 步比较中要进行 $n-i$ 次两两比较，现设 $n=10$。

程序如下：

```
#include <iostream.h>
void main()
{
    int x[10];
    int i,j,s;
    cout<<"输入 10 个整数:";
    for(i=0;i<10;i++)
        cin>>x[i];
    cout<<"\n";
    for(i=0;i<9;i++)
        for(j=0;j<9-i;j++)
            if(x[j]>x[j+1])
                {s=x[j];x[j]=x[j+1]; x[j+1]=s; }
    cout<<"排序结果为:";
    for(i=0;i<10;i++)
        cout<<x[i]<<"  ";
}
```

运行结果如下：

```
输入 10 个整数:
5 3 2 -5 12 7 10 98 -4 -1↙
排序结果为:
-5  -4  -1  2  3  5  7  10  12  98
```

【案例 5-4】 用选择排序法对 10 个整数排序(由小到大)。

算法提示：选择排序法也是一种常用的排序方法，首先从 10 个数中选择一个最小的

数，将它和最前面的那个数交换位置，然后再从剩下数中选择一个最小的数和第二个数交换位置，依次类推，经过 9 次选择可以将 10 个数按照从小到大排序。例如：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	
2	9	17	26	0	5	53	22	91	40	未排序时的情况
0	9	17	26	2	5	53	22	91	40	将 10 个数中最小的数与 a[0]交换
0	2	17	26	9	5	53	22	91	40	将余下的 9 个数中最小的数与 a[1]交换
0	2	5	26	9	17	53	22	91	40	将余下的 8 个数中最小的数与 a[2]交换
0	2	5	9	26	17	53	22	91	40	将余下的 7 个数中最小的数与 a[3]交换
0	2	5	9	17	26	53	22	91	40	将余下的 6 个数中最小的数与 a[4]交换
0	2	5	9	17	22	53	26	91	40	将余下的 5 个数中最小的数与 a[5]交换
0	2	5	9	17	22	26	53	91	40	将余下的 4 个数中最小的数与 a[6]交换
0	2	5	9	17	22	26	40	91	53	将余下的 3 个数中最小的数与 a[7]交换
0	2	5	9	17	22	26	40	53	91	将余下的 2 个数中最小的数与 a[8]交换

程序如下：

```
#include <iostream.h>
void main()
{
    int i,j,k,temp;
    int x[10];
    cout<<"输入 10 个整数:";
    for(i=0;i<10;i++)
        cin>>x[i];
    cout<<"\n";
    for(i=0;i<9;i++)
    {  k=i;
       for(j=i+1;j<10;j++)
          if(x[i]>x[j])
          {  k=j;
             if(k!=i)
                {t=x[i];x[i]=x[j];x[j]=t;}
          }
   }
   cout<<"排序结果为:";
   for(i=0;i<10;i++)
   cout<<x[i]<<"  ";
}
```

运行结果如下：

```
输入 10 个整数:
2  9  17  26  0  5  53  22  91  40↙
排序结果为:
0  2  5  9  17  22  26  40  53  91
```

5.2 二维数组

1. 二维数组的定义

数组是用于按顺序存储同类型的数据结构。如果有一个一维数组，它的每个元素都是同类型的一维数组（数组的类型相同包含两层含义：大小相同且各元素的类型相同）时，就形成了二维数组。如图 5-2 所示，当一维数组 a 的各元素 a[0]、a[1]、a[2]又分别是一个一维数组时，便可以用一个二维数组来描述。

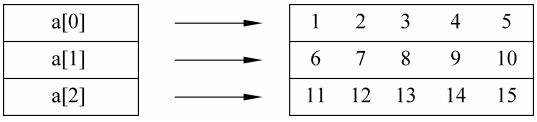


图 5-2 二维数组

在程序设计的过程中，除一维数组外，最常用的就是二维数组了。二维数组定义的一般形式为

```
类型说明符 数组名[常量表达式][常量表达式];
```

例如，图 5-2 中的二维数组 a 的定义形式如下：

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

定义了 a 为 3 行 5 列的二维数组，共有 3×5＝15 个元素。

说明：

(1) 二维数组可以看作是一种特殊的一维数组，它的每个元素又是一个一维数组。

例如，可以把 a 看作是一个一维数组，它有 3 个元素：a[0]、a[1]、a[2]；而 a[0]、a[1]、a[2] 又可以看作是 3 个一维数组的名字。其中，a[0]由 a[0][0]、a[0][1]、a[0][2]、a[0][3]、a[0][4] 5 个元素组成；a[1]由 a[1][0]、a[1][1]、a[1][2]、a[1][3]、a[1][4] 5 个元素组成；a[2]由 a[2][0]、a[2][1]、a[2][2] 、a[2][3]、a[2][4] 5 个元素组成。即：

$$a\begin{cases} a[0]: a[0][0] & a[0][1] & a[0][2] & a[0][3] & a[0][4] \\ a[1]: a[1][0] & a[1][1] & a[1][2] & a[1][3] & a[1][4] \\ a[2]: a[2][0] & a[2][1] & a[2][2] & a[2][3] & a[2][4] \end{cases}$$

注意：二维数组和一维数组一样，数组元素的下标从 0 开始，因此 a 的下标最大的元素是 a[2][4]，而不是 a[3][5]。

(2) C++ 语言中，二维数组的元素在内存中排列的顺序是按行存放。

二维数组中的两个下标自然地形成了表格中的行列对应关系。而实际在计算机中，由于存储器是连续编址的，即存储单元是按一维线性排列的，所以二维数组在计算机内存中是：先顺序存放第一行的元素，再存放第二行的元素……即按行存放。例如，二维数组 a 的元素在内存中的存放顺序如下：

a[0][0]→a[0][1]→a[0][2]→a[0][3]→a[0][4]（第 1 行）→a[1][0]→a[1][1]→a[1][2] →a[1][3]→a[1][4]（第 2 行）→a[2][0]→a[2][1]→a[2][2] →a[2][3]→a[2][4]（第 3 行）

2. 二维数组的引用

二维数组元素的引用形式为

```
数组名[下标][下标]
```

即用数组名和两个带方括号的下标来引用数组元素，下标可以是整型常量或整型表达式，取值从 0 开始。如 a[1][2]、a[3－1][2＊2－3]为正确的引用形式。

和一维数组一样，也可对二维数组的元素进行与变量相同的操作。例如：

```
a[0][0]=a[0][1]/2+100;
```

注意：

(1) 二维数组同样只能逐个引用数组元素，不能一次引用整个数组。例如，可以用 for 循环的嵌套对数组元素逐个操作：

```
int a[3][5];
for(i=0;i<3;i++)
    for(j=0;j<5;j++)
        cin>>a[i][j];
```

(2) 在使用数组元素时，应注意下标值不要越界。例如：

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

二维数组的行下标和列下标和一维数组一样是从 0 开始的，既然定义 a 为 3 行 5 列的数组，它可用的行下标值最大为 2，列下标值最大为 4。用 a[3][5]超过了数组的范围，是错误的。因此，在程序设计中检查数组元素下标是否越界是十分重要的。

3. 二维数组的初始化

二维数组同样存在初始化的问题。二维数组的初始化有以下 4 种形式。

(1) 按行对二维数组初始化。例如：

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

常量表中的第一对花括号中的初始化数据将赋给数组 a 的第一行元素，第二对花括号中的初始化数据将赋给数值 a 的第二行元素，即按行赋值，这种赋值方式清楚直观。

(2) 按数组元素的存放顺序对二维数组初始化。例如：

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

这种方式将所有初始化值写在一个花括号中，依次赋给数组的各元素，初始化结果与前一种方式相同。

(3) 同一维数组一样，可以对二维数组的部分元素赋初值，未赋初值的元素将自动设为零。例如：

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

相当于：

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

再如：

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

赋初值后数组各元素为

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

(4) 如果对全部元素都赋初值，则数组的第一维长度可以省略，但第二维长度不能省略。例如：

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

可以写成

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

编译系统在编译程序时通过对初始值表中所包含的元素值的个数进行检测，能够自动确定这个二维数组的第一维长度。

也可以写成只对部分元素赋初值而省略第一维长度，但应分行赋初值。例如：

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

系统能根据初始值分行情况自动确定该数组第一维的长度为 2。

4. 多维数组

多维数组的定义和引用方法与二维数组相类似。例如：

```
int a[2][3][2];
```

定义了一个三维数组 a，它有 12 个元素，其在内存中存放顺序如图 5-3 所示。可以看出：由上到下先变化第三个下标，然后变化第二个下标，最后变化第一个下标。

三维数组的初始化也可以仿照二维数组的形式进行。

(1) 按行对三维数组初始化。例如：

```
int a[2][3][4]={{{1,2,3,4},{5,6,7,8},{9,10,11,12}},
{{13,14,15,16},{17,18,19,20},{21,22,23,24}}};
```

(2) 按数组元素的存放顺序对三维数组初始化。例如：

```
int a[2][3][4]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,
17,18,19,20,21,22,23,24};
```

这种方式将所有初始化值写在一个花括号中，依次赋给数组的各元素，初始化结果与前一种方式相同。

也可以省略第一维的大小。上面的定义可改写成

```
int a[ ][3][4]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,
17,18,19,20,21,22,23,24};
```

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

图 5-3 三维数组在内存中的存放顺序

(3) 同一维数组和二维数组一样,也可以对三维数组的部分元素赋初值,未赋初值的元素将自动设为零。例如:

```
int  a[2][3][4]={{{1,2},{0,0,7,8},{}},{{13,14},{17},{21,0,0,24}}};
```

依次类推,C++ 语言中也可以定义和使用四维数组、五维数组等。

5. 二维数组程序设计举例

【案例 5-5】 在二维数组 a[3][4]中找出最大的元素和最小的元素,并把它们输出。

源程序如下:

```
#include <iostream.h>
void main()
{
    int a[3][4];
    int i,j,max,min;
    for(i=0;i<3;i++)
        for(j=0;j<4;j++)
        cin>>a[i][j];
    max=min=a[0][0];
    for(i=0;i<3;i++)
        for(j=0;j<4;j++)
        {  if(a[i][j]>max)max=a[i][j];
           else  if(a[i][j]<min)
                     min=a[i][j];
        }
    cout<<"The MAX is "<<max;
    cout<<"The MIN is"<<min;
    cout<<"\n";
}
```

输出结果:

```
  25   88   69   72↙
  33   29   78   96↙
   9   54   48   90↙
The MAX is 96
The MIN is 9
```

【案例 5-6】 编写程序,输入 5 名学生的 3 门课程的成绩,计算并输出每名学生的平均分和每门课程的平均分。

算法提示:可设一个二维数组 score[5][3]存放 5 名学生的 3 门课程的成绩。再设两个一维数组 aver1[5]和 aver2[3],其中,aver1 存放每名学生的平均分,aver2 存放每门课程的平均分。

源程序如下:

```
#include <iostream.h>
void main()
```