Chapter 5

第5章 利用数组处理批量数据

本章学习目标

- 掌握数组的概念、定义和数组元素的引用方法。
- 掌握一维数组、二维数组在程序设计中的应用。
- 掌握字符数组和字符串的应用。

数组是C语言提供的一种构造类型数据，用来有效地处理一批同类型的相关数据。数组是一组相同类型数据的有序集合，每一个数据称为数组元素，这些数组元素有一个共同的名字即数组名，不同元素由其在数组中的序号即下标(从0开始编号)来标识。本章介绍数组的概念、定义和数组元素的引用方法，以及一维数组、二维数组、字符数组和字符串在程序设计中的应用。

5.1 一维数组的定义和引用

C语言中，变量必须先定义，然后才能使用。数组也一样，在使用数组前要对它加以定义。数组定义的主要目的是确定数组的名称、数组的大小和数组的类型。

数组是一组相同类型数据的有序集合，每一个数据称为数组元素，这些数组元素有一个共同的名字称为数组名，不同元素由其在数组中的序号即下标来标识。用1个下标确定元素的数组称为一维数组，用2个下标确定元素的数组称为二维数组，用3个及3个以上下标确定元素的数组称为多维数组。C程序中用得比较多的是一维数组和二维数组，这一节先介绍一维数组，下一节将介绍二维数组。

5.1.1 一维数组的定义

一维数组定义的一般形式为：

```
数组类型 数组名[常量表达式];
```

其中：数组类型为C语言的类型说明符，标识数组元素的类型；数组名由用户指定，但必须符合C语言标识符命名规则；常量表达式应为正整数常量，表示数组的长度即数组元素的个数；[ ]是数组的标志。

数组定义后，C语言编译系统就在计算机内存中为数组分配一块连续的存储空间，用来依次存放数组中各元素的值。

例如：

```
int a[10];
```

表示定义了数组a，数组的类型是int，该数组包含10个元素，分别是a[0]、a[1]、a[2]、a[3]、a[4]、a[5]、a[6]、a[7]、a[8]、a[9]，每一个元素可以用来表示1个int类型的数据，C编译系统在内存中为int类型数组a分配10 * sizeof(int)个字节的连续空间，作为这10个元素的存储区域，数组名a同时表示这一片连续存储空间的开始地址。

数组定义时需要注意，数组长度的定义不能用变量。例如：

```
int n=10, a[n];
```

是错误的。

在定义数组的同时，也可以同时为数组元素赋初值，称为数组的初始化。例如：

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

则C在为数组分配存储单元的同时，还为数组各元素赋初值，a[0]的初值为1、a[1]的初值为2、……、a[8]的初值为9、a[9]的初值为10。

一维数组的初始化有以下几点规则：

(1) 如对数组的所有元素赋初值，则可以不指定数组的长度。

例如：

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

与上面数组a的定义等价，这时数组的长度根据初值的个数确定。

(2) 如对部分元素赋初值，则从数组第一个元素起按顺序给出初值，未赋初值的数值类型数组元素初值为0、字符类型数组元素初值为'\0'。

例如：

```
int a[10]={1,2,3};
```

使a数组前3个元素的值依次为1、2、3，其余元素值为0。

```
char b[5]={'+','-'};
```

使b数组前两个元素的值依次为'+'、'-'，其余元素值为'\0'。

(3) 初值的个数不能多于数组长度。

例如：语句 int a[5]={1,2,3,4,5,6,7,8,9,10}；是错误的。

5.1.2　一维数组元素的引用

数组定义后，就可以在程序中使用了。对数组的使用是通过引用数组元素实现的，不能将数组作为一个整体加以引用。一维数组元素的引用方式如下：

```
数组名[下标表达式]
```

下标表达式可以是整型表达式或字符型表达式，其取值范围是 0～数组长度-1。一个数组元素就是一个变量，称为下标变量。

例如：

```
int a[10];                          //数组定义
a[0]=5;                             //第 1 个元素赋值为 5
a[1]=2*a[3/4];                      //第 2 个元素值为 2*a[0],即 10
a[5]=a[3%2]+a['f'-'e'];             //第 6 个元素值为 a[1]+a[1],即 20
```

【例 5.1】 用数组求 Fibonacci 数列的前 40 项，并打印输出。

程序设计分析：Fibonacci 数列的前两个数是 1、1，第三个数是前两个数的和，以后的每个数都是其前两个数的和。即：

$$F1=1 \quad (n=1)$$
$$F2=1 \quad (n=2)$$
$$F_n=F_{n-1}+F_{n-2} \quad (n\geqslant 3)$$

```
#include <stdio.h>
void main()
{
    int i;
    long int f[40]={1,1};
    for(i=2;i<40;i++)                   //第 5 行
        f[i]=f[i-2]+f[i-1];
    for(i=0;i<40;i++)
    {
        if(i%5==0) printf("\n");        //第 8 行
        printf("%10ld",f[i]);
    }
}
```

程序运行：

```
       1         1         2         3         5
       8        13        21        34        55
      89       144       233       377       610
     987      1597      2584      4181      6765
   10946     17711     28657     46368     75025
  121393    196418    317811    514229    832040
 1346269   2178309   3524578   5702887   9227465
14930352  24157817  39088169  63245986 102334155
```

程序说明：

数组 f 的前两个元素 f[0]、f[1](注意 C 语言数组的下标从 0 开始)通过初始化赋初值，所以第 5 行的循环语句 i 从 2 起；程序第 8 行的 if 语句用于控制每一行输出 5 个数。

【例 5.2】 编程，输入一个班级学生(不多于 50 人)的一门功课成绩，计算平均成绩，并统计高于平均成绩的学生人数。

```
#include <stdio.h>
void main()
{
    float score[50],sum=0,aver;
    int i,n,k=0;
    printf("请输入学生人数:\n");
    scanf("%d",&n);
    printf("请输入%d 个学生成绩:\n",n);
    for(i=0;i<n;i++)          //第 8 行
    {
        scanf("%f",&score[i]);
        sum+=score[i];
    }
    aver=sum/n;
    for(i=0;i<n;i++)          //第 13 行
        if(score[i]>=aver) k++;
    printf("平均成绩: %.2f\t 高于平均分人数: %d\n",aver,k);
}
```

程序运行：

```
请输入学生人数:
10↙
请输入 10 个学生成绩:
60 70 65 75 78 90 54 77 93 68↙
平均成绩: 73.00 高于平均分人数: 5
```

程序说明：

要处理的数据个数不确定，但有一个范围，即不多于50个，因此所定义的数组长度为50。实际所处理的数据要根据人数n决定，n在运行时首先输入。第8行的循环语句用来输入学生成绩和求成绩累加和，score[i]为数组元素，相当于一个普通变量，它的地址为&score[i]；第13行的循环语句用来统计高于平均成绩的人数。

初学者往往将这类问题的数组按下面的方法定义：

```
float sum=0,aver;
int i,n,k=0;
printf("请输入学生人数:\n");
scanf("%d",&n);
float score[n];
```

这是不行的，数组长度的定义不能用变量。

5.2　二维数组的定义和引用

二维数组的每个数据有行列之分，用两个下标标识一个数组元素，适合于矩阵处理等应用。二维数组的每一行元素其实就是一个一维数组，所以二维数组也被称作“数组的数组”。

5.2.1　二维数组的定义

与一维数组相同，二维数组也必须先定义，后使用。二维数组定义形式：

```
数组类型 数组名[常量表达式][常量表达式];
```

其中：数组类型为C语言的类型说明符，标识数组元素的类型；数组名也由用户命名；常量表达式应为正整数常量，第一个常量表达式表示数组的行数，第二个常量表达式表示每一行的元素个数；[]是数组的标志。

例如：

```
float a[3][4];
```

表示定义了二维数组a，数组的类型是float，该数组包含3行4列共12个元素，分别是a[0][0]、a[0][1]、a[0][2]、a[0][3]、a[1][0]、a[1][1]、a[1][2]、a[1][3]、a[2][0]、a[2][1]、a[2][2]、a[2][3]，每一个元素可以用来表示1个float类型的数据，C语言编译系统在内存中为float类型数组a分配3*4*sizeof(float)个字节的连续空间，作为这12个元素的存储区域。

同样，不能用变量定义数组长度。如int n=3，m=4，a[n][m]；是错误的；另外在定义二维数组时，行列长度要分别用一对方括号括起来，写成int a[3,4]是不正确的。

定义数组后，C语言编译系统就在计算机内存中为二维数组分配一块连续的存储空间，

各元素在该区域内的存放顺序为以行为主序存放，即先依次存放第 1 行各元素，再依次存放第 2 行各元素，以此类推。

在定义二维数组的同时也可以初始化各数组元素，方法与一维数组初始化相似，格式如下：

```
二维数组定义={{表达式列表 1},{表达式列表 2},… }
```

或

```
二维数组定义={表达式列表}
```

第一种方法按行初始化，将表达式列表 1 中的数值给数组的第一行、表达式列表 2 中的数值给数组的第二行……；第二种方法将所有初始化数据写在一对花括号内，自动按行初始化，没有初始化到的数组元素值为 0。

例如：

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

则为数组 a 第 1 行各元素 a[0][0]、a[0][1]、a[0][2]、a[0][3]依次赋值 1、2、3、0；为第 2 行各元素 a[1][0]、a[1][1]、a[1][2]、a[1][3]依次赋值 4、5、0、0；而第 3 行各元素初值均为 0。

当对二维数组初始化时，可以省略数组定义中的第一维长度，此时数组行数由初始化数据的个数和列数决定。例如：

```
int a[][4]={{1,2},{3,4,5},{6}}; 与 int a[3][4]={{1,2},{3,4,5},{6}};等价。
int b[][3]={1,2,3,4,5,6,7}; 表示数组 b 是 3 行 3 列数组。
而 int a[][]={1,2,3,4,5,6}; 不正确，不能确定每一行数组元素的个数。
```

这一节开始时讲过，二维数组是“数组的数组”，每一行元素就是一个一维数组。例如：

```
float a[3][4];
```

定义了 3 行 4 列的二维数组，它由三个一维数组组成，即：

(1) a[0][0]、a[0][1]、a[0][2]、a[0][3]，数组名 a[0]。

(2) a[1][0]、a[1][1]、a[1][2]、a[1][3]，数组名 a[1]。

(3) a[2][0]、a[2][1]、a[2][2]、a[2][3]，数组名 a[2]。

5.2.2 二维数组元素的引用

与一维数组相同，二维数组也是由一组相同类型元素组成的，每一元素是一个下标变量，对这些变量的引用和操作与一维数组相似。二维数组元素的引用方式为：

```
数组名[下标表达式 1][下标表达式 2]
```

其中，下标表达式 1 和下标表达式 2 为整型表达式或字符型表达式，取值应限制在 0～行长度-1 和 0～列长度-1。

例如，有定义：

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

则数组元素为 x[0][0]、x[0][1]、x[0][2]、x[1][0]、x[1][1]、x[1][2]。每一元素是一个整型变量，可以参加相应运算。如：

```
scanf("%d",&x[0][0]);          //输入数组元素 x[0][0]的值
x[1][3-1]=x[0][0]%10;          //将 x[0][0]的个位数赋值给 x[1][2]
x[1][2]++;                     //x[1][2]自增 1
```

【例 5.3】 矩阵转置。

程序设计分析：矩阵转置即行列互换，原矩阵第 i 行数据转置后变成第 i 列。用二维数组表示矩阵，矩阵的转置操作就是将数组元素 a[i][j]与 a[j][i]互换。程序如下：

```
#define N 4
#include <stdio.h>
void main()
{
    float a[N][N],temp;
    int i,j;
    for(i=0;i<N;i++)                        //双重循环读入二维数组各元素
        for(j=0;j<N;j++)
            scanf("%f",&a[i][j]);
    for(i=0;i<N;i++)                        //第 10 行，这一双重循环完成行列互换
        for(j=0;j<i;j++)
            { temp=a[i][j]; a[i][j]=a[j][i]; a[j][i]=temp; }
    for(i=0;i<N;i++)                        //输出转置后的矩阵
        {
            for(j=0;j<N;j++)
                printf("%8.2f",a[i][j]);    //以行列对齐格式输出处理后的矩阵
            printf("\n");                   //输出 1 行后换行
        }
}
```

程序说明：

本程序完成二维数组 a 的行列互换，由第 10 行所在的嵌套循环完成，注意第 10 行开始的双重循环，内循环条件为 j<i，不能写成 j<N，否则转置两次后矩阵恢复原样，内循环 for 语句也可以写成 for(j=i+1;j<N;j++)。

【例 5.4】 找出二维数组的最大值并与最后一个元素进行交换。

程序设计分析：本程序所要完成的主要功能是要找出二维数组中的最大值及它所在的行、列号。在查找二维数组的最大值时，首先取第一个元素为存放最大值变量的初始值，然

后逐一与其他元素比较，重新记录最大值，并记录下它们的行、列号，比较完所有元素就可确定最大值以及它所在的位置，再与数组中的最后一个元素交换即可。程序如下：

```
#define M 3
#define N 4
#include <stdio.h>
void main()
{
    float a[M][N],max;
    int i,j,maxi,maxj;
    for(i=0;i<M;i++)
        for(j=0;j<N;j++)
            scanf("%f",&a[i][j]);                    //输入数据
    max=a[0][0];                                     //第 11 行
    maxi=maxj=0;                                     //第 12 行
    for(i=0;i<M;i++)                                 //第 13 行
        for(j=0;j<N;j++)
            if(max<a[i][j])                          //第 15 行
            { max=a[i][j]; maxi=i; maxj=j; }
    a[maxi][maxj]=a[M-1][N-1];
    a[M-1][N-1]=max;
    for(i=0;i<M;i++)
    {
        for(j=0;j<N;j++)
            printf("%8.2f",a[i][j]);
        printf("\n");
    }
}
```

程序说明：

在查找一组数中的最大值时，需要设置相应变量的初始值，作为比较的基础。由于一组数中的最大值肯定是这组数中的某一个，在程序设计时往往先假定第 1 个数是最大数，程序第 11、12 行完成此功能，注意所赋的初值必须是这组数据之一，不能是无关的常量。如第 11 行写成 max＝0;，会有什么样的结果？请读者自己思考。

程序第 13 行开始的双重循环完成查找过程，第 15 行 if 语句在查找到目前为止的最大值时，改变 max 值，同时要记录下它的位置即行、列号，否则最后无法确定它的位置；程序最后是交换和输出整个二维数组的全体元素。

5.3　字符数组

5.3.1　字符数组的定义和使用

字符数组是元素类型为字符型的数组，字符数组的每一个元素可以存放一个字符。字符数组也有一维、二维等。

字符数组的定义、初始化和引用方法同前面介绍的其他类型的数组是类似的。

例如：

```
char a[10]; a[0]='C';a[1]='H';a[2]='I';a[3]='N';a[4]='A';
```

定义了长度为 10 的字符数组，对前 5 个元素进行了赋值。

此时数组 a 的内存形式如图 5.1 所示，未被赋值的数组元素值不确定。

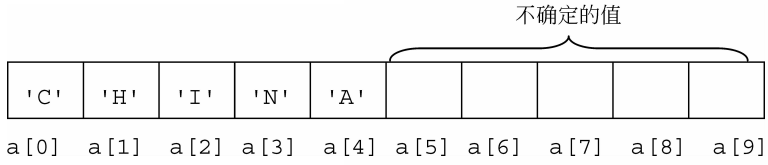

图 5.1　未被赋值的数组元素值不确定

字符数组在定义的同时也可以初始化数组元素，例如：

```
char a[10]={'C','H','I','N','A'};
```

前 5 个元素被依次初始化为：'C'、'H'、'I'、'N' 和 'A'，后 5 个元素都被初始化为 '\0' 。数组 a 的内存形式如图 5.2 所示。

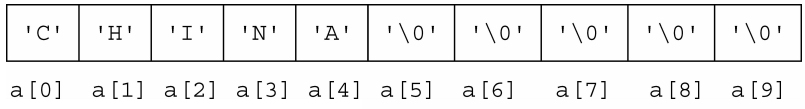

图 5.2　后面的元素被初始化为'\0'

5.3.2　字符数组和字符串

1. 字符串

字符串是一个用双引号括起来的以'\0'结束的字符序列，其中的字符可以包含字母、数字、其他字符、转义字符、汉字(一个汉字占 2 个字节)。

字符串在内存存放时自动在最后加上一个 ASCII 码值为 0 的字符即 '\0'，此字符被称为字符串结束标志。C 语言中利用字符数组来表示字符串，在字符串处理时不以字符数组的长度为准，而是检测字符 '\0' 来判别字符串是否处理完毕。

定义字符数组时可用字符串初始化字符数组，例如：

```
char c[6]={"CHINA"}; 或 char c[6]="CHINA";
```

用字符串初始化字符数组时最后自动添加字符串结束标志'\0'，上面的两种定义和初始化方式等价于：

```
char c[6]={'C','H','I','N','A','\0'};
```

此时数组 c 的内存形式如图 5.3 所示。

'C'	'H'	'I'	'N'	'A'	'\0'
C[0]	C[1]	C[2]	C[3]	C[4]	C[5]

图 5.3　用字符串初始化字符数组

几点注意事项：

(1) 数组的长度必须比字符串的元素个数多 1，用以存放字符串结束标志'\0'。如定义语句 char c[5]="CHINA"；是错误的。

(2) 用字符串初始化字符数组时，可以省略数组长度的定义，如 char c[ ]="CHINA";。

(3) 数组名是地址常量(它表示 C 语言编译系统分配给该数组连续存储空间的首地址，详见第 10 章指针)，不能将字符串直接赋给数组名。

例如：char c[6]；c="CHINA"；是错误的。

(4) 字符串到第一个'\0' 结束。

例如：char c[ ]="abc\0xyz"；则数组 c 的长度为 8，而其中存放的字符串为“abc”。

前面介绍的一维字符数组中可以存放一个字符串，如有若干个字符串则可以用多个一维字符数组或一个二维字符数组来存放。一个 n×m 的二维字符数组可以理解为由 n 个一维数组所组成，可以存放 n 个字符串，每个字符串的最多字符个数为 m−1，因为最后还要存放字符串的结束标志'\0'。

例如：

```
char str[3][9]={"Hangzhou","ShangHai","BeiJing"};
```

定义了一个二维字符数组 str，在内存中的存放形式如图 5.4 所示。

str[0] →	'H'	'a'	'n'	'g'	'Z'	'h'	'o'	'u'	'\0'
str[1] →	'S'	'h'	'a'	'n'	'g'	'H'	'a'	'i'	'\0'
str[2] →	'B'	'e'	'i'	'J'	'i'	'n'	'g'	'\0'	'\0'

图 5.4　定义数组 str 后的内容