第 5 章　数组和指针

5.1　一维数组的定义和引用

为了处理方便，把具有相同类型的若干变量按有序的形式组织起来，这些按序排列的同类变量的集合称为数组。在 C 语言中，数组属于构造数据类型。一个数组可以分解为多个数组元素（简称为元素），这些数组元素可以是基本数据类型或构造类型。因此按数组元素的类型不同，数组又可分为数值数组、字符数组、指针数组等各种类别。

对于序列 $a_0,a_1,a_2,\cdots,a_{n-1}$，在 C 语言中就用数组来表示，$a$ 为数组名，下标代表数组中元素的序号，数组元素也称为下标变量，a_i 用 $a[i]$ 表示，i 就为下标，一维数组是最简单的数组，它的元素只有一个下标，如 $a[8]$，此外还有二维数组（元素有二个下标，如 $a[2][3]$），三维数组（元素有 3 个下标，如 $a[2][3][6]$）和多维数组（元素有多个下标），它们的概念和使用方法相似，本节先介绍一维数组。

5.1.1　定义一维数组

定义一维数组的格式如下：

```
数组元素类型 数组名[常量表达式],…;
```

其中，数组元素类型可以是任意一种基本数据类型或构造数据类型，数组名是用户定义的数组标识符。“[”和“]”之间常量表达式表示数组元素的个数，也称为数组的长度。

例如：

```
int a[18];                  /*定义整型数组 a,有 18 个元素*/
float b[16],c[28];          /*定义实型数组 b,有 16 个元素,实型数组 c,有 28 个元素*/
char ch[29];                /*定义字符数组 ch,有 29 个元素*/
```

对于数组的定义应注意以下几点。

(1) 对于同一个数组，其所有元素的类型都相同。

(2) 数组名不能与其他变量名相同，例如：

```
int main(void)              /*主函数 main()*/
{
    int a;                  /*定义变量 a*/
    float a[10];            /*定义数组*/
    …
```

```
}
```

是错误的。

(3) “[”和“]”之间的常量表达式表示数组元素的个数,如 a[18]表示数组 a 有 18 个元素。但是其下标从 0 开始计算。因此 18 个元素分别为 a[0],a[1],a[2],…,a[17]。

(4) 不能在“[”和“]”之间用变量来表示元素的个数,但是可以是常量或常量表达式。例如:

```
#define FD 5                        /*定义常量 FD*/
int main(void)                      /*主函数 main()*/
{
    int a[3+2],b[7+FD];             /*定义数组 a、b*/
    …
}
```

是合法的。但是下面的使用方式是错误的:

```
int main(void)                      /*主函数 main()*/
{
    int n=5;                        /*定义整型变量 n*/
    int a[n];                       /*定义数组,错,元素个数 n 不是常量表达式*/
    …
}
```

(5) 在同一个类型的定义中,可以同时定义多个变量和多个数组。例如:

```
int a,b,c,d,k1[10],k2[20];          /*定义多个变量与多个数组*/
```

5.1.2 引用一维数组的元素

数组元素是组成数组的基本单元。实际上,数组元素也是一种变量,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。引用一维数组元素的一般格式如下:

```
数组名[下标]
```

其中,下标只能为整型常量、整型变量或整型表达式。例如,a[6]、a[i+j]、a[k]都是合法的数组元素。数组元素通常也称为下标变量。在 C 语言中只能逐个地使用下标变量,而不能一次引用整个数组。例如,输出有 10 个元素的数组必须使用循环语句逐个输出各下标变量:

```
for (i=0;i<10;i++)
    printf("%d",a[i]);              /*循环输出各数组元素*/
```

而不能用一个语句输出整个数组,下面的写法是错误的:

```
printf("%d",a);                     /*错,不能用一个语句输出整个数组*/
```

例 5.1 一维数组示例，程序演示见 5011.mp4～5013.mp4。


```
/* 文件路径名:e5_1\main.c */
#include<stdio.h>                    /* 标准输入输出头文件 */

int main(void)                       /* 主函数 main() */
{
    int i,a[10];                     /* 定义变量 i 与数组 a */

    for (i=0;i<10;i++)
        a[i]=2*i+1;                  /* 循环为数组元素赋值 */
    for (i=9;i>=0;i--)
        printf("%d",a[i]);           /* 循环逆序输出数组元素 */
    printf("\n");                    /* 换行 */

    return 0;                        /* 返回值 0,返回操作系统 */
}
```

程序运行时屏幕输出如下：

```
19 17 15 13 11 9 7 5 3 1
```

本例中用一个循环语句给 a 数组各元素存入奇数值，然后用第二个循环语句按逆序从大到小输出各个奇数。

给数组赋值的方法除了用赋值语句对数组元素逐个赋值外，还可采用初始化赋值和动态赋值的方法。数组初始化赋值是指在数组定义时给数组元素赋予初值。数组初始化是在编译阶段进行的。这样将减少运行时间，提高效率。一维数组初始化赋值的一般格式如下：

```
数组元素类型 数组名[常量表达式]={值 1,值 2,…,};
```

在“{”和“}”中的各数据值即为各元素的初值，各值之间用“,”间隔。例如：

```
int a[10]={0,1,2,3,4,5,6,7,8,9 };          /* 相当于 a[0]=0;a[1]=1;…;a[9]=9 */
```

C 语言对一维数组的初始赋值还有以下几点规定。

(1) 可以只给部分元素赋初值。当“{”和“}”中值的个数少于元素个数时，只给前面部分元素赋值，后面的元素自动赋值为 0，例如：

```
int a[10]={0,1,2};     /* 表示给 a[0]~a[2]共 3 个元素赋值,而后 7 个元素自动赋 0 值 */
```

(2) 只能给元素逐个赋值，不能给数组整体赋值。例如，给 10 个元素全部赋 1 值，只

能写为

```
int a[10]={1,1,1,1,1,1,1,1,1,1};          /*10元素初始化为1*/
```

而不能写为

```
int a[10]=1;
```

(3) 如给全部元素赋值,则在数组定义中,可以不给出数组元素的个数。例如:

```
int a[5]={1,2,3,4,5};            /*初始化5个数组元素值*/
```

可写为

```
int a[]={1,2,3,4,5};             /*初始化5个数组元素值,省略元素个数*/
```

可以在程序执行过程中,对数组作动态赋值。这时可用循环语句配合 scanf()函数逐个对数组元素赋值。

例 5.2 编程求数组元素最大值,程序演示见 5021.mp4～5023.mp4。


```
/*文件路径名:e5_2\main.c*/
#pragma warning(disable:4996)            /*禁止对代号为4996的警告*/
#include<stdio.h>                        /*标准输入输出头文件*/

int main(void)                           /*主函数main()*/
{
    int i,max,a[10];                     /*定义变量与数组*/

    printf("输入10个数:");               /*输入提示*/
    for (i=0;i<10;i++)
        scanf("%d",&a[i]);               /*循环输入数组元素*/
    max=a[0];                            /*设a[0]最大*/
    for (i=1;i<10;i++)
        if (a[i]>max)                    /*如果a[i]更大*/
            max=a[i];                    /*将a[i]赋值给max*/
    printf("最大值:%d\n",max);           /*输出最大值*/

    return 0;                            /*返回值0,返回操作系统*/
}
```

程序运行时屏幕输出如下:

```
输入10个数: 1 7 4 9 12 8 18 56 98 0
```

```
最大值：98
```

本例程序中第一个 for 语句逐个输入 10 个数到数组 a 中。然后把 a[0]送入 max 中。在第二个 for 语句中，a[1]～a[9]逐个与 max 中的值进行比较，若比 max 的值大，则把该下标变量送入 max 中，因此 max 总是在已比较过的下标变量中为最大者的值。比较结束，输出 max 的值。

5.2 二维数组

前面介绍了一维数组，在实际问题中有很多量是二维的或多维的，本节只介绍二维数组，多维数组可由二维数组类推得到。

5.2.1 定义二维数组

定义二维数组一般形式如下：

```
数组元素类型 数组名[常量表达式 1][常量表达式 2],…;
```

其中，常量表达式 1 表示第一维下标的长度，常量表达式 2 表示第二维下标的长度。例如：

```
int a[3][4];                                    /* 定义二维数组 */
```

定义了一个 3 行 4 列的数组，数组名为 a，其下标变量的类型为整型。该数组的下标变量(元素)共有 3×4 个，即

```
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]
```

二维数组在概念上是二维的，即是说其下标在两个方向上变化，下标变量在数组中的位置也处于一个平面之中，而不是像一维数组只是一个方向上变化。但是实际的硬件存储器却是连续编址的，也就是存储器单元是按一维线性排列的。在一维存储器中存放二维数组，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下一行；另一种是按列排列，即存储完一列之后再顺次放入下一列。在 C 语言中，二维数组是按行排列的。对于上面定义的数组，按行顺次存放，先存储第 1 行，再存储第 2 行，最后存储第 3 行。每行中有 4 个元素也是依次存储。具体存储顺序如下：

```
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]
```

5.2.2 引用二维数组的元素

二维数组的元素也称为双下标变量，其引用形式如下：

数组名[下标 1][下标 2]

其中,下标 1 和下标 2 应为整型常量、整型变量或整型表达式。例如,a[3][4] 表示 a 数组 3 行 4 列的元素。引用下标变量与定义数组在形式中有些相似,但两者具有完全不同的含义。定义数组的“[”和“]”之间给出的是某一维的长度;而数组元素中的下标是该元素在数组中的位置标识。前者只能是常量,后者可以是常量、变量或表达式。

例 5.3 一个学习小组有 5 个人,每个人有 3 门课的考试成绩如下所示。求全组分科的平均成绩和各科总平均成绩。

姓名	数学	C 语言	数据库系统
张明	80	75	92
王小艳	68	78	88
李杰	89	98	61
赵刚	78	86	69
周勇	98	96	89

可设一个二维数组 a[5][3]存放 5 个人 3 门课的成绩。再设一个一维数组 v[3]存放所求得各分科平均成绩,设变量 l 为全组各科总平均成绩。程序演示见 5031.mp4～5033.mp4,具体编程如下:


```
/* 文件路径名:e5_3\main.c */
#pragma warning(disable:4996)                /* 禁止对代号为 4996 的警告 */
#include<stdio.h>                            /* 标准输入输出头文件 */

int main(void)                               /* 主函数 main() */
{
    int i,j,s=0,l,v[3]={0},a[5][3];          /* 定义变量 */

    printf("输入成绩:\n");                   /* 输入提示 */
    for (i=0;i<5;i++)
    {   /* 行 */
        for (j=0;j<3;j++)
        {   /* 列 */
            scanf("%d",&a[i][j]);            /* 输入成绩 */
            s=s+a[i][j];                     /* 求和 */
            v[j]=v[j]+a[i][j];               /* 对第 j 门课程成绩求和 */
        }
    }

    for (i=0;i<3;i++)
```

```
        v[i]=v[i]/5;                                    /*求平均成绩*/
    l=s/15;                                             /*总平均成绩*/

    printf("数学:%d\nC语言:%d\n数据库系统:%d\n",v[0],v[1],v[2]);
                                                        /*输出各科平均成绩*/
    printf("总平均成绩:%d\n",l);                        /*输出总平均成绩*/

    return 0;                                           /*返回值0,返回操作系统*/
}
```

程序运行时屏幕输出如下：

```
输入成绩:
80 75 92
68 78 88
89 98 61
78 86 69
98 96 89
数学:82
C语言:86
数据库系统:79
总平均成绩:83
```

本例程序首先用了一个双重循环。在内循环中依次读入某学生3门课程的成绩，并把这些成绩累加起来以及累加各门课程的成绩，退出内循环后再用一个循环求各门课程的平均成绩，然后再求总平均成绩，最后按题意输出各成绩。

5.2.3 二维数组的初始化

二维数组初始化也是在数组定义时给各元素赋以初值。二维数组可按行分段赋值，也可按行连续赋值。例如对数组a[5][3]，按行分段赋值可写为

```
int a[5][3]={{80,75,92},{68,78,88},{89,98,61},{78,86,69},{98,96,89}};
                                                        /*按行分段赋值*/
```

按行连续赋值可写为

```
int a[5][3]={80,75,92,68,78,88,89,98,61,78,86,69,98,96,89};  /*按行连续赋值*/
```

这两种赋初值的结果是完全相同的。

例5.4 将例5.3的成绩通过初始化方式赋值，改写例5.3的程序，求全组各科的平均成绩和各科总平均成绩，程序演示见5041.mp4～5043.mp4。

```
/* 文件路径名:e5_4\main.c */
#include<stdio.h>                          /* 标准输入输出头文件 */

int main(void)                             /* 主函数 main() */
{
    int i,j,s=0,l,v[3]={0};                /* 定义变量 */
    int a[5][3]={                          /* 按行分段赋值 */
        {80,75,92},
        {68,78,88},
        {89,98,61},
        {78,86,69},
        {98,96,89}};

    for (i=0;i<5;i++)
    {   /* 行 */
        for (j=0;j<3;j++)
        {   /* 列 */
            s=s+a[i][j];                   /* 求和 */
            v[j]=v[j]+a[i][j];             /* 对第 j 门课程成绩求和 */
        }
    }

    for (i=0;i<3;i++)
        v[i]=v[i]/5;                       /* 求平均成绩 */
    l=s/15;                                /* 总平均成绩 */

    printf("数学:%d\nC 语言:%d\n 数据库系统:%d\n",v[0],v[1],v[2]);
                                           /* 输出各科平均成绩 */
    printf("总平均成绩:%d\n",l);           /* 输出总平均成绩 */

    return 0;                              /* 返回值 0,返回操作系统 */
}
```

程序运行时屏幕输出如下：

```
数学:82
C 语言:86
数据库系统:79
总平均成绩:83
```

对于二维数组初始化赋值有如下说明。

(1) 可以只对部分元素赋初值,未赋初值的元素自动取 0 值。

例如：

```
int a[3][3]={{1},{2},{3}};          /* 只对部分元素赋初值,其他元素为 0 */
```

对每一行的第一列元素赋值，未赋值的元素取 0 值。赋值后各元素的值为

```
1 0 0 2 0 0 3 0 0
```

(2) 如对全部元素赋初值，则第一维的长度可以不给出。

例如：

```
int a[3][3]={1,2,3,4,5,6,7,8,9};          /* 对全部元素赋初值 */
```

可以写为

```
int a[][3]={1,2,3,4,5,6,7,8,9};           /* 对全部元素赋初值，可省略第一维的长度 */
```

二维数组可以看作是由一维数组的嵌套而构成的。设一维数组的每个元素都是一个一维数组，就组成了二维数组。一个二维数组可以分解为多个一维数组。二维数组 a[3][4] 可分解为 3 个一维数组，其数组名分别为 a[0]、a[1]、a[2]。对这 3 个一维数组无须再做说明即可使用。这 3 个一维数组都有 4 个元素，例如一维数组 a[0]的元素为 a[0][0]、a[0][1]、a[0][2]、a[0][3]。注意 a[0]、a[1]、a[2]是数组名，不是一个单纯的下标变量。

5.3 用数组作为函数的参数

用一维数组名作为函数参数时，要求形参和相对应的实参都必须是元素类型相同的数组，在函数形参表中，允许不给出形参数组的长度，或用一个变量来表示数组元素的个数。

5.3.1 用数组元素作为函数的参数

由于函数实参可以是表达式，数组元素可以是表达式的组成部分，因此数组元素可以作函数的实参，按普通变量的方式对数组元素进行处理。

例 5.5 编写程序实现求给定的数组元素表示的工资中大于 2000 元的人数，程序演示见 5051.mp4～5053.mp4。


```
/* 文件路径名:e5_5\main.c */
#include<stdio.h>                          /* 标准输入输出头文件 */

#define N 10                               /* 定义常量 N */

int main(void)                             /* 主函数 main() */
{
```

```
    int wage[N]={1500,1890,2100,2600,1980,2980,3100,3600,2180,3208};
                                        /*定义工资数组*/
    int i,num=0;                        /*定义变量 i,num*/
    int Count2000(int w);               /*函数声明*/

    for (i=0;i<N;i++)
        num=num+Count2000(wage[i]);     /*统计工资大于 2000 元的人数*/
    printf("共有%d 人的工资大于 2000 元.\n",num);
                                        /*输出工资大于 2000 元的人数*/

    return 0;                           /*返回值 0,返回操作系统*/
}

int Count2000(int w)                    /*函数定义*/
{
    return w>2000 ?1:0;                 /*当 w 大于 2000 时,返回 1,否则返回 0*/
}
```

程序运行时屏幕输出如下：

```
共有 7 人的工资大于 2000 元.
```

本例程序中,Count2000(int w)用于判断 w 的值是否大于 2000,当 w＞2000 时,返回 1,否则返回 0,在主函数 main()中,扫描工资数组 wage,对每个数组元素,调用 Count2000(),并累加返回值。

5.3.2 用一维数组名作为函数的参数

用数组名作为函数参数时,要求形参和相对应的实参都必须是元素类型相同的数组,在函数形参表中,允许不给出形参数组的长度或用一个变量来表示数组元素的个数。

例 5.6 数组 a 中存放了一个学生 5 门课程的成绩,求平均成绩,程序演示见 5061.mp4～5063.mp4。


```
/*文件路径名:e5_6\main.c*/
#include<stdio.h>                       /*标准输入输出头文件*/

float Average(float a[],int n)          /*定义函数*/
{
    int i;                              /*定义整型变量*/
    float av,s=0;                       /*定义实型变量*/
```