

## 第五章 数组

### 本章导读

数组是一种非常重要的构造类型。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唯一确定。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。

### 本章主要知识点

- (1) 一维数组的定义和引用
- (2) 二维数组的定义和引用

## 第五章 数组

### 5.1 一维数组

### 5.2 二维数组

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、多维数组等。

### 5.1 一维数组

#### 5.1.1 一维数组的定义

#### 5.1.2 一维数组元素的引用

#### 5.1.3 一维数组的初始化

#### 5.1.4 冒泡法排序

#### 5.1.1 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];

说明：

- (1) 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。
- (2) 数组名的命名规则遵循标识符的命名规则，它是常量，代表数组存储时的首地址。
- (3) 常量表达式是指数组的长度，即数组元素的个数。

续上

#### 5.1.1 一维数组的定义

- (4) 数组一旦定义后，其长度值就不能再改变。编译系统按照其长度为之分配连续的存储单元。数组名是一个地址常量，代表了这一连续存储单元的首地址。

- (5) 数组长度必须是一个整型常量表达式

```
int n=10;      int a[10+5];      #define N 10
int a[n];      int a[N];
```

错误, n是变量

正确

正确

- (6) 可以用如下方式，同时定义多个相同类型的数组和变量：“int a[10], i, min;”

#### 5.1.2 一维数组元素的引用

数组的使用仍然遵从“先定义，后使用”的原则。数组使用是通过数组元素引用实现的，而不能直接使用整个数组，每一个数组元素就是一个简单变量。

一维数组的数组元素引用形式如下：

数组名[下标]

#### 5.1.2 一维数组元素的引用

说明：

- (1) 下标是一个整型的表达式，可以是常量表达式，也可以是变量表达式。这一点与数组定义时不同。
- (2) 数组元素的下标从0开始，如果数组长度为n，则元素的最大下标为n-1。
- (3) 在一维数组引用过程中要防止下标越界问题。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错误。

#### 5.1.2 一维数组元素的引用

例1：从键盘输入10个整数到数组中，分别按顺序和逆序输出这10个数。

分析：

- > 10个整数要存放到数组中，因此数组定义为int型，数组长度为10。
- > 输入和输出数组元素的过程都是循环结构。
- > 每个数组元素都是一个简单整型变量。
- > 输入一个数组元素使用 scanf(“%d”, &a[i]);
- > 输出一个数组元素使用 printf(“%d”, a[i]);

续上

#### 5.1.2 一维数组元素的引用

```
#include <stdio.h>
main()
{
    int i, a[10];
    printf("input 10 numbers:\n");
    for(i=0; i<10; i++) //从键盘输入10个整数到数组中
    { scanf("%d", &a[i]); }
    for(i=0; i<=9; i++) //顺序输出数组中的所有元素
    { printf("%d ", a[i]); }
    printf("\n");
    for(i=9; i>=0; i--) //逆序输出数组中的所有元素
    { printf("%d ", a[i]); }
    printf("\n");
}
```

#### 5.1.2 一维数组元素的引用

例2：从键盘输入10个整数放在一维数组中，求这10个数中的最大值和最小值。

分析：

- > 编程时可以先定义一个符号常量，然后定义一维数组时使用该符号常量指定数组的长度。
- > 查找数组中的最大值、最小值的方法一样，都是先默认第一个元素最大或者最小，然后依次用后面的数组元素与保存最大值、最小值的变量进行比较。

#### 5.1.2 一维数组元素的引用

```
#include <stdio.h>
#define N 10
main()
{
    int i, a[N], min, max;
    for(i=0; i<N; i++) //从键盘输入10个整数放到一维数组中
        scanf("%d", &a[i]);
    min = max = a[0]; //先默认第1个元素是最大值、最小值
    for(i=1; i<N; i++)
    {
        if(a[i] < min) min = a[i]; //查找最小值
        if(a[i] > max) max = a[i]; //查找最大值
    }
    printf("min = %d, max = %d\n", min, max);
}
```

#### 5.1.3 一维数组的初始化

在定义一维数组同时给数组元素赋初始值，称为一维数组的初始化。一般格式为：

类型说明符 数组名[常量表达式]={初始值表};

几种初始化方法：

- (1) 全部元素赋初值。如：int a[5]={0,1,2,3,4};

- (2) 部分元素赋初值，没有赋值的元素系统自动为其初值零。如：int a[5]={0, 1, 2};

- (3) 全部元素赋初值时，可以缺省数组长度。如：int a[]={0, 1, 2, 3, 4};

#### 5.1.3 一维数组的初始化

例3：先对数组的第1、2个元素进行初始化，然后利用数组求斐波拉契数列的前20项并打印结果。

```
#include <stdio.h>
main()
{
    int i, a[20] = {0, 1}; //只给数组的第1、2项赋初值
    for(i=2; i<20; i++) //求斐波拉契数列的后18项
    {
        a[i] = a[i-1] + a[i-2]; //后一项是前两项之和
    }
    for(i=0; i<20; i++) //顺序输出数组中的20个元素
    {
        printf("%d ", a[i]);
    }
}
```

重点

#### 5.1.4 冒泡法排序

难点

例4：从数组中存放了5个整数，使用冒泡法将这5数按照由小到大的顺序（升序）进行排序。

分析：

- > 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。
- > 从第一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。如果有n个数待排序，则经过n-1轮比较后，就可以完成排序工作。

#### 5.1.4 冒泡法排序

进行升序排序

第1轮排序【待排序的数 20、15、10、5、2】

15	15	15	15	
20	10	10	10	
10	20	5	5	
5	5	20	2	
2	2	2	20	

第1次比较 第2次比较 第3次比较 第4次比较

第1轮所有待比较数中的最大值

#### 5.1.4 冒泡法排序

第2轮排序【待排序的数 15、10、5、2、20】

10	10	10		
15	5	5		
5	15	2		
2	2	15		
20	20	20		

第1次比较 第1次比较 第3次比较

第2轮所有待比较数中的最大值

#### 5.1.4 冒泡法排序

第3轮排序【待排序的数 10、5、2、15、20】

5	5			
10	2			
2	10			
15	15			
20	20			

第1次比较 第2次比较

第3轮所有待比较数中的最大值

#### 5.1.4 冒泡法排序

第4轮排序【待排序的数 5、2、10、15、20】

2

5

第4轮所有待比较数中的最大值

10

15

20

第1次比较

结论:

- 如果有N个数需要排序, 则进行N-1轮排序, 每轮排序时进行比较的次数依次递减。
- 本轮排序结束后, 本轮待比较各数中的最大值或最小值交换到最后, 下一轮该数不再参与比较。
- 如果是进行升序排序, 则相邻两数前者大于后者时就交换; 反之如果进行降序排序, 则相邻两数前者

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

```
#define N 5 //符号常量N用来确定数组的长度
```

```
main()
```

```
{
```

```
    int a[N]={20, 15, 10, 5, 2}, temp, j, k;
```

```
    for(j=0; j<N-1; j++) //外层循环控制排序的轮数
```

```
        for(k=0; k<N-1-j; k++) //内层循环控制每轮排序中比较的次数
```

```
            if(a[k] > a[k+1]) //如果前者大于后者就交换
```

```
            {
```

```
                temp = a[k];
```

```
                a[k] = a[k+1]; //三条首尾相接的完成交换的语句
```

```
                a[k+1] = temp;
```

```
            }
```

```
    for(j=0; j<N; j++) //打印排序后的数组元素
```

```
        printf("%d ", a[j]);
```

```
}
```



## 5.2 二维数组

### 5.2.1 二维数组的定义

### 5.2.2 二维数组元素的引用

### 5.2.3 二维数组的初始化

### 5.2.4 二维数组程序举例

### 5.2.1 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];

说明：

(1) 二维数组区别于一维数组的是数组名后面有两个常量表达式。**常量表达式1代表行数，常量表达式2代表列数。**它们分别指明数组的行长度和列长度。

(2) 二维数组一旦定义后，编译系统就根据其行数和列数为之分配连续的存储单元，数组名代表这一连续存储空间的首地址。数组元素在内存里是**按行存放**的。

### 5.2.1 二维数组的定义

例如定义了二维数组：int a[2][3];

它在内存的存放情况如下图所示

数组元素	假定的地址	
a[0][0]	0x1000	第一行数组元素
a[0][1]	0x1004	
a[0][2]	0x1008	
a[1][0]	0x100C	第二行数组元素
a[1][1]	0x1010	
a[1][2]	0x1014	

### 5.2.2 二维数组元素的引用

二维数组的数组元素引用形式如下：

数组名 [行下标][列下标]

说明：

(1) 二维数组元素引用时，与一维数组元素的引用相似，要注意的是**行下标和列下标都是从 0 开始**。例如定义了一个二维数组：int a[3][4]; 则该数组有3行4列共12个整型变量，行下标的取值范围是0~2、列下标的取值范围是0~3。

(2) 对二维数组元素的引用一般使用两层循环嵌套，可以按行存取，也可以按列存取。

### 5.2.3 二维数组的初始化

二维数组初始化时要注意二维数组的元素排列顺序。初始值的排列顺序必须与数组元素在内存的存储顺序完全一致。具体的方法如下：

几种初始化方法：

(1) 全部元素赋初值——分行初始化

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

(2) 全部元素赋初值——按数组元素排列顺序初始化

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

### 5.2.3 二维数组的初始化

续上

(3) 全部元素赋初值——可以缺省第一维下标

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

**注意：不允许缺省第二维下标**

(4) 部分元素赋初值——分行初始化

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

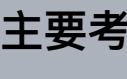
(5) 部分元素赋初值——按数组元素排列顺序初始化

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

### 5.2.4 二维数组的程序举例

例5：程序定义了一个3行3列的二维数组，从键盘输入各元素初值，编程将数组**左下三角（包括对角线）**中的值全部置0。

1	2	3
4	5	6
7	8	9



0	2	3
0	0	6
0	0	0

分析：

➢对二维数组的编程主要考虑的是引用数组元素时的下标值。

➢二维数组左下三角（包括对角线）的所有数组元素有一个共同的特性：**行下标 >= 列下标**

### 5.2.4 二维数组的程序举例

```
#include <stdio.h>
#define N 3
main()
{
    int j, k, a[N][N]={{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    for(j=0; j<N; j++) //第一组循环：打印原始数组
    {
        for(k=0; k<N; k++)
            printf("%d ", a[j][k]);
        printf("\n"); //每一行的末尾需打印一个回车
    }
    for(j=0; j<N; j++) //第二组循环：将左下角元素置 0
        for(k=0; k<=j; k++)
            a[j][k] = 0; //将指定元素赋值为 0
}
```

### 5.2.4 二维数组的程序举例


续上

```
for(j=0; j<N; j++) //第三组循环：打印改变后的数组
{
    for(k=0; k<N; k++)
        printf("%d ", a[j][k]);
    printf("\n");
}
```

### 5.2.4 二维数组的程序举例

例6：将一个2行3列的二维数组的各元素**按列依次**存放到一个一维数组中。

1	2	3
4	5	6



1	4	2	5	3	6
---	---	---	---	---	---

分析：

➢二维数组按列取出各个元素时，外循环变量控制数组元素的列号，内循环变量控制行号。

➢每取一个二维数组元素就存到一维数组中，因此有两个变量是引用二维数组元素时的行、列下标，有一个变量是引用一维数组元素时的

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

```
main( )
```

```
{   int  i=0, j, k, a[2][3]={1, 2, 3, 4, 5, 6}, b[6];  
    for(j=0; j<2; j++) //第一组循环：打印二维数组  
    {  
        for(k=0; k<3; k++)  
            printf("%d ", a[j][k]);  
        printf("\n");  
    }  
    for(j=0; j<3; j++) //第二组循环：按列取二维数组元素  
        for(k=0; k<2; k++)  
        {   b[i] = a[k][j]; //将二维数组元素放到一维数组中  
            i++; //引用一维数组元素的下标自增  
        }  
    for(j=0; j<6; j++) //打印一维数组元素  
        printf("%d ", b[j]);  
}
```



## 5.2.4 二维数组的程序举例

例7：从键盘输入一个4行4列的矩阵，查找其中的最大数以及该数的行、列下标。

```
#include <stdio.h>
void main()
{
    int num[4][4], i, j, max, max_row, max_line;
    printf("请输入一个4行4列的矩阵: \n");
    for(i = 0; i < 4; i++)
    {
        for(j = 0; j < 4; j++)
        {
            scanf("%d", &num[i][j]);
        }
    }

    max = num[0][0];
    max_row = max_line = 0;

    for(i = 0; i < 4; i++)
    {
        for(j = 0; j < 4; j++)
        {
            if(num[i][j] > max)
            {
                max = num[i][j];
                max_row = i;
                max_line = j;
            }
        }
    }
    printf("最大数是%d, 它的行下标是%d, 列下标是%d\n", max, max_row, max_line);
}
```

## 5.2.4 二维数组的程序举例

例8：从键盘输入3个学生4门课的成绩，将输入的信息输出，并计算每门课的平均分。

```
void main()
{
    float score[3][4], aver[4], sum;
    int i, j;
    printf("请输入3个同学4门课的成绩: \n");
    for(i = 0; i < 3; i++)//输入3个同学4门课的成绩
    {
        for(j = 0; j < 4; j++)
        {
            scanf("%f", &score[i][j]);
        }
    }

    printf("输出学生成绩: \n");
    for(i=0; i<3; i++) //输出3个同学4门课的成绩
    {
        for(j=0; j<4; j++)
        {
            printf("%.2f\t", score[i][j]);
            //注意此处的score[i][j]前面没有取地址符&
        }
        printf("\n"); //每一行成绩输出结束后的换行
    }

    for(i=0; i<3; i++) //求每门课的分数的总和
    {
        sum += score[i][0];
    }
    aver[i] = sum / 3.0; //求平均值
    printf("第%d门课的平均成绩是%.2f\n", i+1, aver[i]);
} //main
```

## 5.2.4 二维数组的程序举例

```
//计算4门功课的平均分
for(i=0; i<4; i++)
{
    sum = 0;
    for(j=0; j<3; j++) //求每门课的分数的总和
    {
        sum += score[j][i];
    }
    aver[i] = sum / 3.0; //求平均值
    printf("第%d门课的平均成绩是%.2f\n", i+1, aver[i]);
} //main
```

## 5.3 数组练习题

(1) 以下程序的执行结果是  
main()  
{  
 int p[7]={11, 13, 14, 15, 16, 17, 18}, i=0, k=0;  
 while(i<7 && p[i]%2) { k=k+p[i]; i++; }  
 printf("%d\n", k);  
}  
A) 58      B) 56      C) 45      D)24

## 5.4 综合编程实例

例9：一个整型数组中已经有10个数，从键盘输入1个整数，查找该数是否在该数组中（要求：使用折半查找法）。

```
#include <stdio.h>
void main()
{
    int x, i=0, j=9, k;
    int a[10]={9, 13, 15, 16, 19, 23, 37, 56, 78, 90};
    printf("请输入一个整数: ");
    scanf("%d", &x);
    //查找范围是下标从i到j
}
```

## 5.4 综合编程实例

```
while(i < j)
{
    k = (i + j) / 2;
    if(x < a[k])
    {
        j = k - 1;
    }
    else if(x > a[k])
    {
        i = k + 1;
    }
    else
    {
        printf("该数组中有数值%d\n", x);
        return;
    }
}
printf("该数组中没有数值%d\n", x);
}
```

## 5.4 综合编程实例

例10：从键盘输入10个整数，用选择排序法将这10个整数按照由小到大的顺序排序。

编程思路：选择排序法是指从欲排序的n个数据中，以线性查找的方法找出最小的元素与第一个元素交换，再从剩下的(n-1)个数据中，找出最小的元素与第二个元素交换，以此类推，直到所有元素均已排序完成。即选择排序法不同于冒泡法，它是在每一轮比较结束后，才进行一次交换，因此，从这个角度来看，选择排序法的效率高于冒泡法。

```
#include <stdio.h>
void main()
{
    int t, i, j, k, a[10];
    printf("请输入10个整数: ");
    for(i = 0; i < 10; i++)
    {
        scanf("%d", &a[i]);
    }
    for(i=0; i<9; i++)
    {
        k = i;
        for(j = i + 1; j < 10; j++)
        {
            if(a[k] > a[j])
                k = j;
        }

        if(i != k)
        {
            t = a[k];
            a[k] = a[i];
            a[i] = t;
        }
    }
    printf("升序排序后为: ");
    for(i=0; i<10; i++)
    {
        printf("%d ", a[i]);
    }
    printf("\n");
}
```

## 5.4 综合编程实例

例11：编写程序输出杨辉三角形的前五行，如图所示图形。

编程思路：通过观察，可以将杨辉三角看作一个二维数组中的数据，此二维数组我们只关心左下三角的数据，不涉及右上三角的数据。杨辉三角构成的规则是，第0列和对角线上的数为“1”，其余位置上的数为该数上一行，对应该数所在列以及右侧一列的两数之和。

```
#include <stdio.h>
void main()
{
    int i, j, x[5][5];
    for(i=0; i<5; i++)
    {
        for(j=0; j<=i; j++)
        {
            if(j==0 || i==j)
            {
                x[i][j] = 1;
            }
            else
            {
                x[i][j] = x[i-1][j] + x[i-1][j-1];
            }
        }
    }

    for(i=0; i<5; i++) //输出杨辉三角
    {
        for(j=0; j<=i; j++)
        {
            printf("%d ", x[i][j]);
        }
        printf("\n");
    }
}
```

## 5.4 综合编程实例

1、定义一个大小为10的整型一维数组，初始化所有数组元素，编程查找数组中的所有奇数，并统计奇数的个数。运行效果如图所示：

提示：本题变量定义可参考如下

int a[10]={2,4,5,6,9,11,13,16,17,19}, i, sum=0;

首先用一个for循环打印10个数组元素的初值。再用一个for循环嵌套if语句查找并打印数组中的奇数。

```
#include<stdio.h>
main()
{
    int a[10]={2,4,5,6,9,11,13,16,17,19}, i, sum=0;
    printf("数组的初值是: \n");
    for(i=0;i<10;i++)
    {
        printf("%d ",a[i]);
    }
    printf("\n");
    printf("数值中的奇数是: \n");

    for(i=0;i<10;i++)
    {
        if(a[i]%2!=0)
        {
            printf("%d ",a[i]);
            sum+=1;
        }
    }
    printf("\n");
    printf("奇数的个数是: \n");
    printf("%d ",sum);
    printf("\n");
}
```

2 定义一个大小为10的整形数组，使用初始化方法为数组元素赋初值，编程计算这10个元素的平均值，并查找比平均值大的数组元素。程序运行效果如图所示：

提示：

(1) 本题使用2组for循环，第一组for循环打印数组元素的初值，并计算总和。第二组for循环求比平均值大的数。

(2) 本题的变量定义可参考如下所示

```
int a[10]={3,6,1,7,8,4,9,5,10,2}, i, sum=0;
double aver;

#include<stdio.h>
main()
{
    int a[10]={3,6,1,7,8,4,9,5,10,2}, i, sum=0;
    double aver;
    printf("数组的初值是: \n");
    for(i=0;i<10;i++)
    {
        printf("%d ",a[i]);
        sum+=a[i];
    }
    printf("\n");

    printf("平均值是: %f\n",sum/10.0);
    aver=sum/10.0;
    printf("比平均值大的值是: \n");
    for(i=0;i<10;i++)
    {
        if(a[i]>=aver)
            printf("%d ",a[i]);
    }
    printf("\n");
}
```

3 定义一个4行5列的二维数组，使用二维数组分行初始化的方法为数组元素赋初值，编程求该二维数组周边元素之和。程序运行效果如下图所示：

(1) 本题变量定义可参考如下

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

int i, j, sum = 0;

(2) 思考周边元素的行下标和列下标有何特点

```
#include <stdio.h>
main()
{
    int a[4][5]={1,2,3,4,5}, {6,7,8,9,10}, {11,12,13,14,15}, {16,17,18,19,20};
    int i, j, sum = 0;
    printf("该4行5列的二维数组是: \n");
    for(i=0;i<4;i++)
    {
        for(j=0;j<5;j++)
        {
            printf("%d\t",a[i][j]);
            if((i==0)||(j==0)||(i==3)||(j==4))
                sum+=a[i][j];
        }
    }

    printf("\n");
}
```

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

•

