第 5 章　批量数据的处理——数组

学习目标

- 能够熟练数组的输入输出；
- 掌握使用循环遍历数组的方法；
- 掌握一维数组求最大值、最小值、求和等基本操作；
- 能够使用冒泡法和选择法对一维数组进行排序；
- 能够在一维、二维数组中查找元素；
- 能够在 VC++ 6.0 中使用调试工具，单步跟踪数组的使用情况。

到目前为止，读者已经学习了整型、实型等基本数据类型，这些类型的变量可以用来表示一个单独的数据项。但是，真实世界中有时需要描述大量的数据项，例如，记录一天中整点时刻的温度，定义 24 个实型变量吗？若是记录一年中每天的整点温度，则需要 24×356 个变量，这显然是不合理的。那么对于大量数据的处理应该如何操作呢？数组就是解决这一问题的。

5.1　统计成绩

【例 5-1】 问题描述：考试结束了，学委小淘要对考试成绩进行统计，得到最高分、最低分和平均分，再将成绩按照从大到小的顺序输出，假设共有 10 名同学，请帮助小淘编写程序完成上述要求。

5.1.1　程序分析

【任务分析】

题目中涉及 10 个成绩，首先要解决成绩的输入和存放。1 个成绩可以定义 1 个整型变量存储，那么 10 个成绩该如何存放呢？这是一个批量数据的处理问题，若采用多个单变量的方法需要定义大量变量，这显然是不方便的，有时甚至不可能实现；因此，我们这里引入一种新的数据类型——数组，来解决批量数据的存储和处理问题。

定义一个整型数组 int score[10]存放 10 个成绩，然后通过遍历数组找到最高分、最低分，计算总分进而得到平均分，最后通过特定算法（冒泡法/选择法）对数组 score 进行排序后输出。

【程序代码】

```
#include<stdio.h>
void main(){
  int i,max,min,sum,index,j,temp;
  int score[10];   //定义数组 score,记录 10 个成绩

  printf("输入 10 个成绩: \n");
  for(i=0; i<10; i++)
        scanf("%d", &score[i]);

  //遍历数组,计算总分,找最高分、最低分
  sum=0;
  max=min=score[0];
  for(i=0; i<10; i++){
      sum+=score[i];
      if(max<score[i])
          max=score[i];
      else if(min>score[i])
          min=score[i];
  }
  printf("最高分: %d, 最低分: %d, 平均分: %.2f\n",max,min,sum/10.0);

  //对 10 个成绩排序
  for(i=0; i<9; i++){
      index=i;
      for(j=i+1;j<10;j++)
  if(score[j] >score[index])  index=j;
      if(index! =i)
  temp=score[index],score[index]=score[i],score[i]=temp;
  }
  //输出排序后的成绩
  printf("排序后: \n");
  for(i=0; i<10; i++)
      printf("%d ", score[i]);
  printf("\n");
}
```

【运行结果】

见图 5-1。

在程序设计中，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数据元素的集合称为数组。在 C 语言中，数组属于构造数据类型。

图 5-1 例 5-1 的运行结果

一个数组可以分解为多个数组元素,这些数组元素可以是基本数据类型或是构造类型。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、结构数组等各种类别。

数组用于解决批量数据的存储和处理,数组的元素在内存中连续存放,每个数组元素由数组名和下标唯一确定,使用数组的优点是变量表述简洁,可读性高,便于使用循环结构访问批量数据。本章介绍一维和二维的数值型数组,一维和二维数组是数组操作的基础,其他常用数组,如字符数组、结构体数组将在后续章节中介绍。

5.1.2 一维数组

1. 一维数组的定义、引用和初始化

(1) 定义

一般格式:

```
类型名数组名[数组长度]
```

其中,类型名表示数组元素的类型;数组名是数组(变量)的名称,按标识符的命名规则定义;数组长度指定数组元素的个数,用常量表达式表示。

例如:

```
int a[10];          定义一个含有 10 个整型元素的数组 a
char c[200];        定义一个含有 200 个字符元素的数组 c
float f[5];         定义一个含有 5 个浮点型元素的数组 f
```

注意:

- 数组的类型实际上是指数组元素的取值类型。对于同一个数组,其所有元素的数据类型都是相同的。
- 数组名后是方括号括起来的常量表达式,不能是变量或变量表达式,即不允许对数组做动态定义。
- 数组名不能与其他变量名相同。
- 允许在同一个类型说明中说明多个数组和多个变量。

例如:

```
int a,b,c,d,k1[10],k2[20];
```

(2) 引用

引用数组实质是引用数组中的数组元素，数组元素是组成数组的基本单元，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。

数组元素的一般形式为：

```
数组名[下标]
```

其中，“下标”只能为整型常量或整型表达式，取值范围：[0，数组长度－1]；方括号“[]”是一个运算符，名为“下标运算符”，下标运算符前是一个数组名，即对某个数组做取下标元素的运算。

注意区分方括号在数组定义和数组元素引用时的作用：

- 在数组定义时，方括号是一个说明符，说明定义的变量是一个数组型变量，方括号里的值是数组元素的个数。
- 在数组元素引用时，方括号是下标运算符，方括号中的值在[0，数组长度－1]之间，不能越界。

例如：

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

- 整型数组 a 有 10 个元素：a[0]、a[1]、…、a[9]。
- a[5]、a[i+j]、a[i++]都是合法的数组元素。

数组元素的使用方法与同类型的变量相同：

```
scanf("%d",&a[i]);printf("%d ", a[i]);
temp=a[index]; a[index]=a[k]; a[k]=temp;
```

在 C 语言中只能逐个地使用数组元素，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数组元素：

```
for(i=0; i<10; i++)
printf("%d",a[i]);
```

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);
```

(3) 一维数组在内存中的存放

一维数组在内存中是用连续空间存放的。例如：

```
int score[10];
```

1000	10	score[0]
1004	20	score[1]
1008	30	score[2]
1012	40	score[3]
…	…	…
1036	100	score[9]

每个数组元素从低地址到高地址依次存放，占用的字节数就是数组类型的字节数，数组长度为(数组元素的数据类型所占空间×数组元素个数)，一个 int 元素占 4 个字节，则 score 数组的长度为 4×10 字节，共 40 字节。

数组元素连续存放，为数组元素的随机访问提供了支持，可以使用下标运算符随机访问数组元素。

(4) 一维数组的初始化

一维数组初始化是指在定义数组时，对数组元素赋初值，形式如下：

```
类型名数组名[数组长度]={初值表};
```

一维数组的初始化具体有三种情况：

- 全部元素初始化：

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

- 针对部分元素的初始化：

```
int b[5]={1, 2, 3};则 b[0]=1, b[1]=2, b[2]=3, b[3]=0, b[4]=0
```

- 如果对全部元素都赋初值，可以省略数组长度：

```
int a[]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

注意：

- 只有在对数组进行初始化，并给出了全部初值时，才允许省略数组长度。
  以下表示是错误的：

```
int a[];
```

- 只用在数组定义时，才能对数组元素批量初始化，数组定义后，只能逐个进行初始化。
  以下表示是错误的：

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

2. 使用一维数组编程

(1) 一维数组的输入和输出

使用循环实现数组的输入和输出，数组下标作为循环变量，通过循环逐个处理数组元素。

例如：

```
int a[5],i;
```

输入	输出

for(i=0; i<5; i++)
  scanf("%d",&a[i]);
	for(i=0; i<5; i++)
	  printf("%d\t", a[i]);

说明：

- 循环变量的初值从 i=0 开始，而不是从 i=1 开始，循环控制表达式是 i<5，而不是 i<=5，下标不能越界。
- 输入时，格式控制字符串"%d"中不要添加其他普通字符，否则输入要对照输入格式原样输入。
- 输出时，数组元素之间应该有间隔，可以使用"\t"或空格实现间隔。

(2) 遍历一维数组

一维数组的遍历是对数组元素的逐个访问，遍历最常用的方式是通过 for 循环遍历数组元素的下标的方式来实现的。

【例 5-2】 求含有 10 个成绩的成绩数组的最高分。

【程序代码】

```
void main(){
    int i,score[10],max;
    …//输入

    max=score[0];
    for(i=0; i<10; i++)
        if(score[i]>max)
            max=score[i];

    …//输出
}
```

代码中，用 max 记录最高分，在遍历数组前对 max 赋初值 score[0]，即假设第一个成绩就是最高分，然后使用 for 循环遍历数组，将每个元素与 max 进行比较，若有比 max 大的值，则更新 max，遍历结束后，max 中记录的就是最高分。遍历数组时可以从 i=0 开始，也可以从 i=1(第二个元素)开始。同理，查找最低分和计算总分，也用类似的办法遍历数组。

(3) 一维数组排序

【例 5-3】 采用“选择法”对任意输入的 10 个成绩按从大到小进行排序后输出。

算法思想：在要排序的一组数中，将 n 个数依次比较，保存最大数的下标位置，然后将最大数和与第一个位置的数交换；然后在剩下的 n－1 个数当中再找最大数的下标，然后将次大数与第二个位置的数交换，如此循环直到倒数第二个数和最后一个数比较完为止。例如：“8,6,9,2,3,7”的选择法排序过程如下：

原始数据	8	6	9	2	3	7
第1趟交换后	9	6	8	2	3	7
第2趟交换后	9	8	6	2	3	7
第3趟交换后	9	8	7	2	3	6
第4趟交换后	9	8	7	6	3	2
第5趟交换后	9	8	7	6	3	2

算法分析：每趟选出一个最值和无序序列的第一个数交换，n 个数共选 n－1 趟。第 i 趟假设 i 为最值下标，然后将最值和 i＋1 至最后一个数比较，找出最值的下标，若最值下标不为初设值，则将最值元素和下标为 i 的元素交换。

【程序代码】

```
void main(){
    int i, score[10], j, index, temp, n;
    …//输入
    //选择法对 10 个成绩排序
    n=10;
    for(i=0; i<n-1; i++){
        index=i;
        for(j=i+1;j<n;j++)
            if(score[j]>score[index])  index=j;
        if(index!=i)
            temp=score[index],score[index]=score[i],score[i]=temp;
    }
    …//输出
}
```

代码中，index 记录最大值下标，for(j)的内循环用于查找无序序列中的最大值。

【例 5-4】 采用“冒泡法”对任意输入的 10 个成绩按从大到小进行排序后输出。

算法思想：在要排序的一组数中，对当前还未排好序的范围内的全部数，自上而下对相邻的两个数依次进行比较和调整，让较小的数往下沉，较大的数往上冒。即每当两相邻的数比较后发现它们的排序与排序要求相反时，就将它们互换。例如：“8，6，9，2，3，7”的选择法排序过程如下：

第1次比较	8	6	9	2	3	7	
第2次比较	8	6	9	2	3	7	
第3次比较	8	9	6	2	3	7	
第4次比较	8	9	6	2	3	7	
第5次比较	8	9	6	3	2	7	
第1趟比较结果	8	9	6	3	7	2	最小值沉到最后

算法分析：如果有 n 个数，则要进行 n－1 趟比较。在第 1 趟比较中要进行 n－1 次相邻元素的两两比较，在第 j 趟比较中要进行 n－j 次两两比较。比较的顺序从前往后，经过一趟比较后，将最值沉底（换到最后一个元素位置），最大值沉底为升序，最小值沉底为降序。

【程序代码】

```
void main(){
    int i, score[10], j, temp;
    …//输入
    //冒泡法对 10 个成绩排序
    n=10;
    for(i=0; i<n-1; i++){
```

```
        for(j=0; j<n-1-i; j++) {
            if(scores[j]>scores[j+1]) {
                //交换元素
                temp=scores[j];
                scores[j]=scores[j+1];
                scores[j+1]=temp;
            }
        }
    }
    …//输出
}
```

冒泡排序速记口诀(升序)：n 个数字来排队，两两相比大靠前，外层循环 n－1，内层循环 n－1－i。

(4) 查找一维数组中某个元素的位置

【例 5-5】 输入一个成绩，在成绩数组中查找该成绩，若找到了则输出成绩在数组中存放位置的下标，若未找到则输出“未找到”信息。

问题分析：需要使用循环对数组进行遍历，循环有两个出口：一是找到该成绩，通过 break 提前结束循环；二是数组全部遍历完，循环条件 i＜size(size 是数组长度)为假时结束循环。两个出口代表两个不同结果：break 提前结束循环代表“找到了”，此时循环变量 i 的值即为成绩在数组中存放位置的下标；循环条件 i＜size 为假结束循环代表“未找到”。因此在循环结束后，要判断循环是从哪个出口结束的。这种结构与之前的素数判断类似。

【程序代码】

```
void main(){
    int i, score[10],grade;
    …//输入 10 个成绩
    printf("输入要查找的成绩: ")
    scanf("%d",&grade);
    for(i=0; i<10; i++)
        if(score[i]==grade)  break;
    if(i<10)
        printf("找到了,成绩%d 的下标为%d: \n",grade,i);
    else
        printf("未找到\n");
}
```

代码中，判断循环由哪个出口结束时，判断条件要么与循环条件相同，要么与循环条件恰好相反。本例中，采用的是与循环条件相同；若选择与循环条件恰好相反可写成“i＞＝10”或“!(i＜10)”。

5.1.3　程序的调试与运行

(1) 建立一个 C 程序，将例 5-1 的代码添加进去。

(2) 分别在 06、11、21、31 行设置断点，分功能段调试程序。

（3）将变量观察窗口展开至图 5-2 形式，单步执行，随程序运行观察数组元素的变化情况。

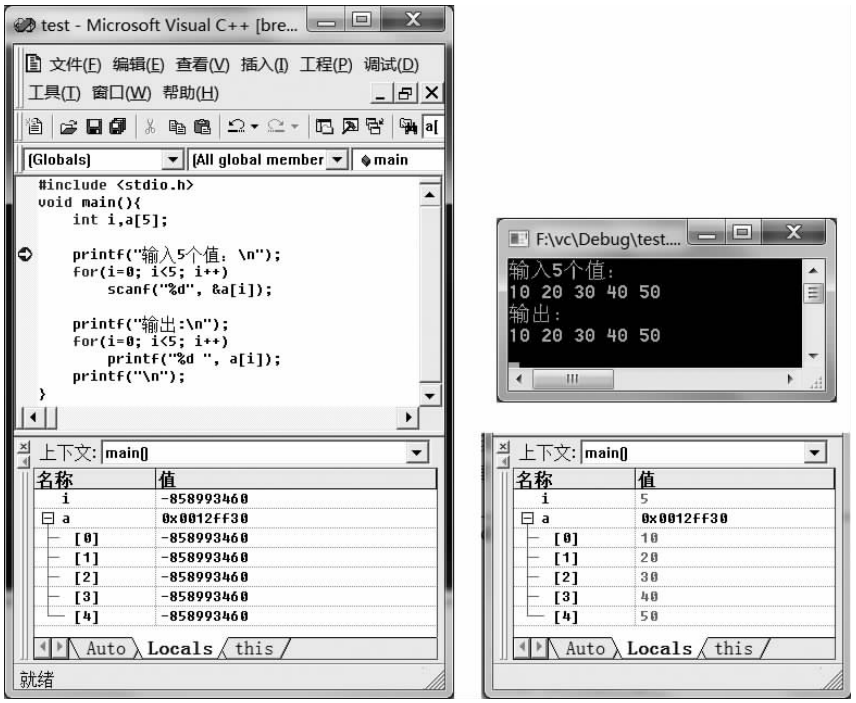

图 5-2　一维数组输入输出调试运行情况

5.2　打印杨辉三角

【例 5-6】 任务描述

杨辉三角如下所示：

1

1　1

1　2　1

1　3　3　1

…

本任务是打印杨辉三角前 n 行，n∈[1,16]。

5.2.1　程序分析

【任务分析】

题目中涉及 n 行数列，第一行 1 个数，第二行 2 个数……第 n 行 n 个数。需要定义一个二维数组存储 n 行 n 列的数据，杨辉三角的两条斜边都是由数字 1 组成的，而其余的数

则是等于它肩上的两个数之和。

【程序代码】

```
#include<stdio.h>
void main(){
    int i,j,n=0,a[16][16]={0};
    while(n<1 || n>16){
        printf("请输入杨辉三角的行数:");
        scanf("%d",&n);
    }
    //计算 n 行杨辉三角
    for(i=0;i<n;i++)
        a[i][0]=1,a[i][i]=1;
    for(i=1;i<n;i++)
        for(j=1;j<i;j++)
            a[i][j]=a[i-1][j-1]+a[i-1][j];
    //输出 n 行杨辉三角
    for(i=0;i<n;i++){
        for(j=0;j<=i;j++)
            printf("%5d",a[i][j]);
        printf("\n");
    }
}
```

【运行结果】

见图 5-3。

图 5-3　例 5-6 的运行结果

有些问题，如表格数据的存储处理，仅用一维数组是无法完成的，因为表格中的数据位置由行号和列号共同描述，这需要用二维数组来处理。二维数组本质上是以数组作为数组元素的数组，即“数组的数组”。所以也可以把二维数组看成一个“特殊的一维数组”，它的每一个元素都是一个一维数组。