第5章

数　　组

在第4章编写了一个10个数中找最大数的程序，因为每个数据只需要使用一次，因此只使用一个变量来存储输入数据，从而利用循环完成了任务。现在稍微调整一下需求，要求输出所有超过平均数的数据，这时每个数据将不止使用一次（一次用于计算平均数，一次与平均数进行比较），此时将使用数组这一数据存储形式存储输入的数据，然后再利用循环结构解决这一新的问题。

5.1　一维数组

数组是存放在连续内存单元中的一组数据类型相同的元素的有限集合。数组中存放的所有数据的数据类型相同，数组中的每个数据被称为一个数组元素。通过数组名和数组下标来使用数组中的数据，数组下标代表数据在数组中的位置，数组下标从0开始。

1. 一维数组的声明

一维数组有两种声明方式，格式如下：

```
类型标识符　数组名[ ];
```

或

```
类型标识符[ ]　数组名;
```

数组元素的数据类型可以是Java语言中的任何数据类型，如前面学过的基本数据类型（int、float、double、char、boolean等），或者将来学习的类（class）或接口（interface）类型等。数组名是符合Java标识符定义规则的用户自定义标识符。

例如“int　a[ ]; ”或“int[ ]　a;”都表示声明一个一维数组a，该数组存放int类型的数据；而“double[ ] array1,array2;”表示声明两个一维数组array1和array2，这两个数组存放double类型的数据。

数组定义后，系统将给数组名分配一个内存单元，其值为数组在内存中的实际存放地址。由于在数组变量定义时，数组元素本身在内存中的实际存放地

址还没有分配,所以,此时该数组名的值为空(null)。

2. 一维数组的初始化

因为在数组声明中并未明确指出数组元素的个数,系统无法知道需要给这个数组分配多大的内存空间。在使用数组元素之前,必须要对其进行初始化操作,使系统知道数组元素的个数,并为其分配连续的存储空间。初始化数组后,数组就可以使用了,在 Java 语言中用 new 关键字来完成数组的初始化操作,也可以在声明数组时直接指定数组元素的初值。

1) 用 new 关键字初始化数组

用 new 关键字初始化数组,只为数组分配存储空间而不对数组元素赋初值。用 new 关键字来初始化数组有两种方式。

(1) 先声明数组,再初始化数组。这实际上由两条语句构成,格式如下:

```
类型标识符  数组名[ ];
数组名=new 类型标识符[数组长度];
```

其中,第一条语句是数组的声明语句,第二条语句是初始化语句。应该注意的是,两条语句中的数组名、类型标识符必须一致。数组长度是整型常量或整型变量,用以指明数组元素的个数。例如:

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

第一句是声明一个整型数组变量 a,第二句是初始化该数组为存放 10 个数组元素的整型数组。这种初始化方式最为常用。

(2) 在声明数组的同时使用 new 关键字初始化数组。这种初始化实际上是将上面的两条语句合并为一条语句。

例如:

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

Java 语言规定,在数组分配内存单元后,系统将自动给每个数组元素赋值,其中数值类型数组元素的初值为 0,逻辑类型数组元素的初值为 false。

2) 直接指定数组元素初值的方式

用直接指定数组元素初值的方式对数组初始化,是指在声明一个数组的同时将数组元素的初值依次写入赋值运算符后的一对花括号内,给这个数组的所有数组元素赋上初始值。这样,Java 编译器可通过初值的个数确定数组元素的个数,为其分配存储空间并将这些值写入相应的存储单元中。例如:

```
int[ ] a={19,23,29,31,37};
```

这条语句声明类型为 int 的数组 a,数组元素共有 5 个,初始值分别为 a[0]=19,a[1]=23,a[2]=29,a[3]=31,a[4]=37。Java 中的数组下标从 0 开始。

3. 一维数组的使用

当数组初始化后就可通过数组名与数组下标来访问数组中的每个元素。一维数组元素的引用格式如下：

```
数组名[数组下标]
```

其中，数组名是经过声明和初始化的标识符，数组下标是指元素在数组中的位置，数组下标的取值范围是 0～(数组长度－1)。下标值可以是整数型常量或整数型变量表达式。例如，在有了“int[ ] a＝new int[10];”语句后，下面的两条赋值语句是合法的：

```
a[3]=25;
a[3+6]=50;
```

但赋值语句“a[10]＝8;”是错误的，这是因为数组 a 在初始化时确定其长度为 10，第 10 个元素的下标是数字 9，不存在下标为 10 的数组元素 a[10]。

例 5.1　输入 10 个学生的某科成绩，要求输出高于平均分的成绩。

解题思路：显然程序需要两次使用学生的成绩，第一次计算平均成绩，第二次查看所有学生成绩，然后输出高于平均分的成绩。程序代码如下：

```
import java.util.*;
public class Array_01 {
    public static void main(String[] args) {
        int[] a;
        //平均成绩含有一位小数,用实数类型
        double ave=0;
        a=new int[10];
        Scanner reader=new Scanner(System.in);
        //利用循环输入 10 个学生的成绩,注意下标为 0~9
        for(int i=0;i<10;i++)
            a[i]=reader.nextInt();
        //先求成绩总和,再计算平均成绩,可减少除法次数
        for(int i=0;i<10;i++)
            ave=ave+a[i];
        ave=ave/10;
        System.out.println("平均值是"+ave);
        //检查数组中每个元素,输出符合要求的元素
        for(int i=0;i<10;i++)
            if(a[i]> ave)
                System.out.println(a[i]);
    }
}
```

因为本程序数组中的数据是 10 个，所以在 3 个与数组操作有关的循环中(输入数据、计算和、输出数据)的循环判断表达式都是 i＜10。在 Java 语言中，数组经初始化后就确

定了它的长度,对于每个已分配了存储空间的数组,Java 语言用一个数据成员 length 来存储这个数组的长度值,调用方式为“数组名.length”。可以将表达式 i<10 改写为 i<a.length,这样不仅提高了程序的可读性,在修改程序时(例如,学生人数变为 30)也更加方便了。

5.2 一维数组应用

数组用于存放同种数据类型的一批数据,在对这些数据进行处理时,对于每个数据的处理方式往往相同,正好使用循环结构对数组元素进行操作。

例 5.2 数组中存放若干数据,输入一个数据,判断此数据是否在数组中,如果在输出其所在位置,否则输出不在的提示信息。

解题思路:利用循环将数组中存放的每个元素都与要查找的数据进行比较,如果相同则输出当前数组元素的下标,如果数组中所有的元素都与要查找的数据不同,则输出没找到的提示信息。

```
import java.util.*;
public class ArraySearch {
    public static void main(String[] args) {
        int[] a={1,3,5,7,9,2,4,6,8,10};
        int i,x;
        boolean find=false;     //是否找到的标记,false 表示没找到
        Scanner reader=new Scanner(System.in);
        System.out.println("请输入要查找的数:");
        x=reader.nextInt();
        for(i=0;i<a.length;i++)
            if(a[i]==x){
                System.out.println("找到了,下标为"+i+"的数是"+x);
                //数组中有可能有多个 x,如果找到一个即可,可以用 break 语句
                find=true;
            }
        if(!find)
            System.out.println("没找到!");
    }
}
```

例 5.3 数组中存放若干数据,输入一个数据,判断此数据是否在数组中,如果在删除这个数据,否则输出不在的提示信息。

解题思路:采用类似于排队的思路,若删除的是第 3 个数据,则原来第 4 个数据变为第 3 个数据,第 5 个数据变为第 4 个数据,以此类推。程序代码如下:

```
import java.util.*;
public class ArrayDelete {
    public static void main(String[] args) {
```

```
        int[] a={1,3,5,7,9,2,4,6,8,10};
        int i,x;
        boolean find=false;
        Scanner reader=new Scanner(System.in);
        System.out.println("请输入要删除的数:");
        x=reader.nextInt();
        for(i=0;i<a.length;i++)
            if(a[i]==x){
                find=true;
                break;
            }
        if(find){
            for(;i<a.length-1;i++)
                a[i]=a[i+1];
            System.out.println("删除后数据:");
            for(i=0;i<a.length-1;i++)
                System.out.print(a[i]+" ");
        }else{
            System.out.println("数组中不存在该数据。");
        }
    }
}
```

在程序中，如果找到了数据 x，则后面的数据依次向前移动，程序实际上隐含假定了 x 在数组中只出现一次。如果 x 在数组中存在多次，程序还需要进行相应的修改。实际上在大多数情况下，都没有要求必须移动数据，因此只需选择一个不在应用范围内的数据作为当前位置没有数据的标志，例如，如果处理的是成绩，可以用数字－1 作为成绩不存在的标记，这样编写的程序代码如下：

```
import java.util.*;
public class ArrayDelete2 {
    public static void main(String[] args) {
        int[] a={1,3,5,7,9,2,4,6,8,10};
        int i,x;
        boolean find=false;
        Scanner reader=new Scanner(System.in);
        System.out.println("请输入要删除的数:");
        x=reader.nextInt();
        for(i=0;i<a.length;i++)
            if(a[i]==x){
                find=true;
                a[i]=-1;
            }
        if(find){
```

```
                System.out.println("删除后数据:");
                for(i=0;i<a.length;i++)
                    if(a[i]!=-1)
                        System.out.print(a[i]+" ");
            }else{
                System.out.println("数组中不存在该数据。");
            }
        }
    }
```

这样编写的程序,无论是删除一个数据还是多个数据,程序都能正确运行。其实计算机应用中的绝大多数删除都是采用类似的处理方式,以减少数据移动的代价。例如,在操作系统中有些被删除的文件可以从回收站中恢复,就是因为在删除磁盘文件时采用了与上述代码类似的方式,只是标记出文件被删除,并没有真正从磁盘上删除文件,所以在文件被覆盖前才能恢复被删除的文件。

5.3 二维数组

一维数组处理的数据相当于一个序列,类似于表中的一行或一列数据,二维数组处理的相当于一个数据表。

1. 二维数组的声明

二维数组的声明与一维数组类似,只是需要给出两对方括号,其格式如下:

```
类型说明符  数组名[ ][ ];
```

或

```
类型说明符[ ][ ]  数组名;
```

例如:

```
int  arr[ ][ ];
```

或

```
int [ ][ ]  arr;
```

2. 二维数组的初始化

二维数组的初始化也分为直接指定初值和用 new 关键字两种方式。

1) 用 new 关键字初始化数组

用 new 关键字初始化数组有两种方式。

(1) 先声明数组再初始化数组。在数组已经声明以后,可用下述两种格式中的任意一种来初始化二维数组。

```
数组名=new 类型说明符[数组长度][数组长度];
```

或

```
数组名=new 类型说明符[数组长度][ ];
```

其中，对数组名、类型说明符和数组长度的要求与一维数组一致。

例如：

```
int arr[ ][ ];                          //声明二维数组
arr=new int[3][4];                      //初始化二维数组
```

上述两条语句声明并创建了一个 3 行 4 列的二维数组 arr。也就是说，arr 数组有 3 行，每行有 4 个元素，实际上每行也是长度为 4 的一维数组。数组共有 12 个元素，共占用 12×4＝48 字节的连续存储空间。

使用第二种格式初始化二维数组，实际上可以声明每行数据元素数量不同的二维数组，例如可以声明一个三角形的二维数组。

例如：

“arr＝new int[3][ ];”创建一个有 3 行的二维数组，也可以说是创建一个有 3 个元素的数组，并且每个元素也是一个数组。

“arr[0]＝new int[3];”创建 arr[0]元素的数组，它有 3 个元素，这是二维数组的第一行。

“arr[1]＝new int[4];”创建 arr[1]元素的数组，它有 4 个元素，这是二维数组的第二行。

“arr[2]＝new int[5];”创建 arr[2]元素的数组，它有 5 个元素，这是二维数组的第三行。

也就是说，在初始化二维数组时也可以只指定数组的行数而不给出数组的列数，每行的长度由二维数组引用时决定。但不能只指定列数而不指定行数。

(2) 在声明数组时初始化数组。格式如下：

```
类型说明符[ ][ ] 数组名=new 类型说明符[数组长度][ ];
```

或

```
类型说明符 数组名[ ][ ]=new 类型说明符[数组长度][数组长度];
```

例如：

```
int[ ][ ] arr=new int[4][ ];
int arr[ ][ ]=new int[4][3];
```

可以不指定列数只指定行数，但是，不指定行数而指定列数是错误的。例如，下面的初始化是错误的。

```
int[ ][ ] arr=new int[ ][4];
```

2) 直接指定数组元素初值的方式

在数组声明时对数据元素赋初值就是用指定的初值对数组初始化。例如:

```
int[ ][ ] arr={{3, -9,6},{8,0,1},{11,9,8} };
```

声明并初始化数组 arr,它有 3 个元素,每个元素又都是有 3 个元素的一维数组。用指定初值的方式对数组初始化时,各子数组元素的个数可以不同。

3. 二维数组的长度及数组赋值

与一维数组一样,也可以用 length 属性获得二维数组的长度,即元素的个数。需要注意的是,当使用"数组名.length"的形式获得的是数组的行数;而使用"数组名[i].length"的形式获得的是该行数组元素的个数。

5.4 二维数组应用

例 5.4 输入 n×n 的二维数组数据,输出其主对角线和副对角线数据之和。

解题思路:访问二维数组元素采用双重循环,外层循环控制访问每行,内层循环控制访问该行中的元素。主对角线元素行和列的下标相同,副对角线元素行和列的下标之和为 n−1。程序代码如下:

```
import java.util.*;
public class ArrayDiagonal {
    public static void main(String[] args) {
        int[][] a;
        int i,j,n,sum1=0,sum2=0;
        Scanner reader=new Scanner(System.in);
        n=reader.nextInt();
        a=new int[n][n];
        //输入二维数组数据
        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
                a[i][j]=reader.nextInt();
        //求主对角线和副对角线之和
        for(i=0;i<n;i++){
            sum1+=a[i][i];
            sum2+=a[i][n-1-i];
        }
        //按行输出二维数组中的数据
        for(i=0;i<n;i++){
            for(j=0;j<n;j++)
                System.out.print(a[i][j]+"\t");
            System.out.println();
        }
```

```
        //输出主对角线和副对角线之和
        System.out.println(sum1+","+sum2);
    }
}
```

例 5.5　假定在 3×4 的矩阵中的最大值和最小值都仅有一个，编写程序，交换矩阵中最大值与最小值所在的行。

解题思路：用二维数组存储矩阵，先找到最大值和最小值，记住其所在的行，查找方式与一维数组时相同；交换这两行中对应元素的数据，对于这两行中的每个对位元素，都要交换，显然需要使用针对该行元素个数的一个循环。程序代码如下：

```
import java.util.*;
public class ArraySwap {
    public static void main(String[] args) {
        int[][] a;
        int i,j,max,min,maxI,minI,t;
        a=new int[3][4];
        Scanner reader=new Scanner(System.in);
        for(i=0;i<3;i++)
            for(j=0;j<4;j++)
                a[i][j]=reader.nextInt();
        //第一个数据既可能是最大值,也可能是最小值
        max=min=a[0][0];
        //如果第一个数据是最大值或最小值,记下当前最值所在的行
        maxI=minI=0;
        //对于每行
        for(i=0;i<3;i++)
            //对于行中的每个数据
            for(j=0;j<4;j++){
                //找最大值
                if(max<a[i][j]){
                    max=a[i][j];maxI=i;
                }
                //找最小值
                if(min> a[i][j]){
                     min=a[i][j];minI=i;
                }
            }
        //交换最大值和最小值所在行的元素
        for(j=0;j<4;j++){
            t=a[maxI][j];a[maxI][j]=a[minI][j];a[minI][j]=t;
        }
        //按行输出数组数据
        for(i=0;i<3;i++){
```

```
                for(j=0;j<4;j++)
                    System.out.print(a[i][j]+"\t");
                System.out.println();
            }
        }
    }
```

在程序中,在查找最大值和最小值的双重循环之前有语句“max=min=a[0][0];”,如果不写这一条语句,且 a[0][0]恰好是最大值或最小值之一,那么程序逻辑上有错误。

5.5 查找和排序

查找和排序是在程序设计中经常使用的两种技术,下面通过学习一种快速的查找算法——二分查找,以及两种简单直观的排序算法——选择排序和冒泡排序,来了解简单的查找和排序程序的设计。

1. 二分查找算法

本书之前编写的在数据序列中查找数据的程序,采用的是从头到尾的顺序查找方式,顺序查找的效率比较低,适合数据量比较小的情况。如果有 n 个数据,使用顺序查找最多需要判断 n 次,才能给出数据在或不在数据序列中。如果数据量比较大,如像图书馆中的书籍信息,这时管理人员通常会采取一定方式保证数据是有序的,如按从小到大的顺序进行排序,这样就可以使用一些效率更高的方法进行数据的查找了。二分查找算法就是最常用的一种快速查找算法。

如图 5.1(a)所示,要查找的数据是由小到大有序的,现要查找数据 x 是否在这批数据中;已知待查找数据的起始位置 s(start)、结束位置 e(end),可计算得到二分位置 m(mid,中间位置,m=(s+e)/2),如图 5.1(b)所示;将 x 与 m 位置上的数据进行比较,不妨假定 x 比 m 位置上的数大,则如果数据 x 在该序列中,一定在 m 位置之后的部分中(因为数据是有序的,m 位置之前的数都比 m 位置上的数小),如图 5.1(c)所示;现在 e 不变,新的 s=m+1,根据新的 s 可计算新的 m 位置,如图 5.1(d)所示;这次假定 x 比 m 位置上的数小,则如果数据 x 在该序列中,一定在 m 位置之前的部分中,如图 5.1(e)所示;此时 s 不变,新的 e=m-1,根据新的 e 可计算新的 m 位置,如图中 f 所示;这样一直查找下去,直到找到数据或确定 x 不在数据序列中。

二分查找的结束条件:

(1) 找到了,此时 m 位置上的数就是 x;

(2) 没找到,新的待查找序列中已经没有数据了,判定条件为 s>e,因为事先约定 s 是开始位置、e 是结束位置,开始位置不应在结束位置之后。

例 5.6 利用二分查找算法在升序数组中(为输入数据方便,假定为 10 个元素)查找数据。

根据上述思想编写的二分查找程序代码如下: