第 4 章

数组

技能目标

- 掌握数组的定义
- 掌握数组的初始化
- 掌握数组的遍历
- 掌握 Arrays 类的常用方法
- 掌握二维数组及其使用

本章任务

学习本章，需要完成以下 3 个工作任务。记录学习过程中遇到的问题，可以通过自己的努力或访问 kgc.cn 解决。

任务 1：使用数组进行基本运算

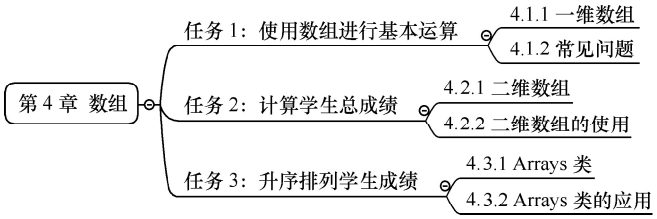
输入 5 个学生的成绩，计算 5 个学生的平均分、最高分和最低分。

任务 2：计算学生总成绩

分别计算 1 班、2 班和 3 班学生的总成绩。

任务 3：升序排列学生成绩

分别对 1 班、2 班和 3 班学生的成绩进行升序排列。



任务 1 使用数组进行基本运算

关键步骤如下。

- 创建一个长度为 5 的整型数组。
- 定义两个 float 类型变量，用于保存总成绩、平均分，初始值均为 0。
- 定义两个 int 类型变量，用于保存最高分和最低分，初始值均为 0。
- 从控制台接收 5 个学生的成绩。
- 通过循环使数组的 5 个元素相加得到总成绩。
- 通过循环遍历数组，比较元素大小，得到最高分及最低分。

4.1.1 一维数组


数组详解

1. 理解数组

在前面的章节中已经学习了诸如整型、字符型和浮点型等数据类型，这些数据类型操作的往往是单个数据，如示例 1 所示。

示例 1

存储 50 个学生某门课程的成绩并求 50 人的平均分。

分析：

采用之前学习的知识点实现，可以定义 50 个变量，分别存放 50 个学生的成绩。

关键代码：

```
int score1 = 95;
int score2 = 89;
int score3 = 79;
int score4 = 64;
int score5 = 76;
int score6 = 88;
//……此处省略 41 个赋值语句
int score48 = 70;
int score49 = 88;
int score50 = 65;
average ＝ (score1+score2+score3+score4+score5+…+score50)/50;
```

示例 1 的代码缺陷很明显，首先是定义的变量的个数太多，如果存储 10000 个学生的成绩，难道真要定义 10000 个变量吗？这显然不可能。另外也不利于数据处理，如要求计算所有成绩之和或最高分，要输出所有成绩，就需要把所有的变量名都写出来，这显然不是一种好的实现方法。

Java 针对此类问题提供了有效的存储方式——数组。在 Java 中，数组是用来存储一组相同类型数据的数据结构。当数组初始化完毕后，Java 为数组在内存中分配一段连续的空间，其在内存中开辟的空间也将随之固定，此时数组的长度就不能再发生改变。即使数组中没有保存任何数据，数组所占据的空间依然存在。数组的存储方式如图 4.1 所示。

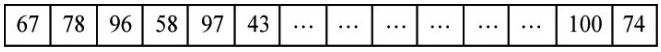


图 4.1　数组存储方式

提示

如果没有特殊说明，在本章中所说的数组均表示一维数组。

2. 定义数组

在 Java 中，定义数组的语法有如下两种。

```
数据类型 [] 数组名 = new 数据类型 [ 数组长度 ];
```

或者：

```
数据类型 数组名 [] = new 数据类型 [ 数组长度 ];
```

- 定义数组时一定要指定数组名和数组类型。
- 必须书写“[]”，表示定义了一个数组，而不是一个普通的变量。
- “[数组长度]”决定连续分配的空间的个数，通过数组的 length 属性可获取此长度。
- 数组的数据类型用于确定分配的每个空间的大小。

示例 2

使用两种语法分别定义整型数组 scores 与字符串数组 cities，scores 的长度是 5，cities 的长度是 6。

关键代码：

```
int [] scores = new int[5];
String cities[] = new String[6];
```

示例 2 为数组 scores 分配了 5 个连续空间，每个空间存储整型的数据，即占用 4 字节空间，每个空间的值是 0。为数组 cites 分配了 6 个连续空间，用来存储字符串类型的数据，每个空间的值是 null。数组元素分配的初始值如表 4-1 所示。

表4-1　数组元素分配的初始值

数组元素类型	默认初始值
byte、short、int、long	0
float、double	0.0
char	'\u0000'
boolean	false
引用数据类型	null

3. 数组元素的表示与赋值

由于定义数组时内存分配的是连续的空间，所以数组元素在数组里顺序排列编号，该编号即元素下标，它标明了元素在数组中的位置。首元素的编号规定为 0，因此，数组的下标依次为 0、1、2、3、4……依次递增，每次的增长数是 1。数组中的每个元素都可以通过下标来访问。例如，数组 scores 的第一个元素表示为 scores[0]。

获得数组元素的语法格式如下。

```
数组名 [ 下标值 ]
```

例如，下面两行代码分别为 scores 数组的第一个元素和第二个元素赋值。

```
scores[0]=65;// 表示为 scores 数组中的第一个元素赋值 65
scores[1]=87;// 表示为 scores 数组中的第二个元素赋值 87
```

4. 数组的初始化

所谓数组初始化，就是在定义数组的同时一并完成赋值操作。

数组初始化的语法格式如下。

```
数据类型 [] 数组名 = { 值 1, 值 2, 值 3,……, 值 n};
```

或者：

```
数据类型 [] 数组名 = new 数据类型 []{ 值 1, 值 2, 值 3,……, 值 n};
```

下面两个语句都是定义数组并初始化数组。

```
int scores[] = {75,67,90,100,0}; // 创建一个长度为 5 的数组 scores
// 或者
int scores[] = new int[]{75,67,90,100,0};
```

5. 遍历数组

在编写程序时，数组和循环往往结合在一起使用，可以大大地简化代码，提高程序编写效率。通常使用 for 循环遍历数组。

示例 3

创建整型数组，从控制台接收键盘输入的整型数，并对数组进行循环赋值。

实现步骤如下。

（1）创建整型数组。

（2）创建 Scanner 对象。

（3）将循环变量 i 作为数组下标，循环接收键盘输入，并为数组元素赋值。

关键代码：

```
public static void main(String[] args) {
    int scores[] = new int[5];                    // 创建长度为 5 的整型数组
    Scanner input = new Scanner(System.in);
    for(int i = 0; i < scores.length; i++) {      //scores.length 等于数组的长度 5
        score[i] = input.nextInt();               // 从控制台接收键盘输入，  进行循环赋值
    }
}
```

示例 3 中使用 for 循环为数组元素赋值，下面再使用 for 循环输出数组元素。

示例 4

创建整型数组，循环输出数组元素。

实现步骤如下。

（1）初始化整型数组。

（2）以循环变量 i 为数组下标，循环输出数组元素。

关键代码：

```
public static void main(String[] args) {
    int scores[] = {75,67,90,100,0};              // 创建有 5 个元素的整型数组
    for(int i = 0; i < scores.length; i++){       // length 等于 5
        // 每次循环 i 的值相当于数组下标
        System.out.println("scores[" + i + "]=" + scores[i]);
    }
}
```

JDK 1.5 之后提供了增强 for 循环语句，用来实现对数组和集合中数据的访问，增强 for 循环的语法如下。

```
for( 元素类型 变量名 : 要循环的数组或集合名 ){……}
```

第一个元素类型是数组或集合中元素的类型，变量名在循环时用来保存每个元素的值，冒号后面是要循环的数组或集合名称。示例 5 使用增强 for 循环实现逐一输出数组元素的功能。

示例 5

创建整型数组，使用增强 for 循环输出数组元素。

分析如下。

该语句的含义是依次取出数组 scores 中各个元素的值并赋给整型变量 i，同时输出其值。

实现步骤如下。

（1）初始化整型数组。

（2）使用增强 for 循环。

关键代码：

```
public static void main(String[] args) {
   int scores[] = {75,67,90,100,0};
   for(int  i : scores){
      System.out.println(" 数组元素值依次为 ：  " + i);
   }
}
```

输出结果：

```
数组元素值依次为 ：  75
数组元素值依次为 ：  67
数组元素值依次为 ：  90
数组元素值依次为 ：  100
数组元素值依次为 ：  0
```

注意

变量 i 的类型必须和数组 scores 元素的类型保持一致。

6. 使用数组计算成绩

示例 6

使用数组计算 5 名学生的平均分、最高分和最低分。

实现步骤如下。

（1）定义一个长度为 5 的整型数组。

（2）定义两个 float 类型变量，用于保存总成绩、平均分，初始值均为 0。

（3）定义两个 int 类型变量，用于保存最高分和最低分，初始值均为 0。

（4）从控制台接收 5 名学生的成绩。

（5）通过循环使数组的 5 个元素相加得到总成绩。

（6）通过循环遍历数组并比较元素大小，得到最高分及最低分。

关键代码：

```
public static void main(String[] args){
   int[] scores=new int[5];                                  // 定义长度为 5 的整型数组
   float total=0;                                            // 总成绩
   float avg=0;                                              // 平均分
   int max=0;                                                // 最高分
   int min=0;                                    /           / 最低分
   Scanner input=new Scanner(System.in);
   System.out.println(" 请输入 5 个学生的笔试成绩 ：  ");       // 输出提示信息
   for(int i=0; i<scores.length; i++){
```

```
        scores[i]=input.nextInt();
    }
    // 计算总成绩、 最高分和最低分
    max=scores[0];
    min=scores[0];
    for(int j=0; j<scores.length; j++){
        total+=scores[j];
        if(scores[j]>max){                    // 第一次循环 max 为 scores[0]
            max=scores[j];
        }
        if(scores[j]<min){                    // 第一次循环 min 为 scores[0]
        min=scores[j];
        }
    }
    // 计算平均成绩
    avg=total/scores.length;
    // 输出 5 名学生的总成绩、 最高分、 最低分和平均分
    System.out.println(" 总成绩 ： " + total);
    System.out.println(" 最高分 ： " + max);
    System.out.println(" 最低分 ： " + min);
    System.out.println(" 平均分 ： " + avg);
}
```

本任务的输出结果如图 4.2 所示。

图 4.2　使用数组计算 5 个学生的平均分、 最高分和最低分

在日常使用数组的开发中，除了定义、赋值和遍历操作之外，还有很多其他操作。例如，对数组进行添加、修改、删除操作。

（1）数组添加

示例 7

如图 4.3 所示，当已经存在一个数组“phones”时，如何往数组的“null”位置插入数据呢？大致思路是首先查找位置，然后进行添加。

```
String[] phones = {"iPhone4","iPhone4S"," iPhone5",null};
```

图 4.3　数组添加

关键代码：

```
public class SuppleMent{
   public static void main(String[] args){
      // 数组添加
      int index=-1;
      String[] phones = {"iPhone4","iPhone4S","iPhone5",null};
      for(int i=0;i<phones.length;i++){
          if(phones[i]==null){
             index=i;
             break;
          }
      }
      if(index!=-1){
          phones[index]="iPhone5S";
          for(int i=0;i<phones.length;i++){
             System.out.println(phones[i]);
          }
      }else{
          System.out.print(" 数组已满 ");
      }
   }
}
```

分析：

index 变量相当于一个“监视器”。赋初始值“–1”是为了和数组下标的 0、1、2 等区别开来。遍历数组中的元素，如果发现了 null 就会把 i 赋值给 index，相当于找到 null 的下标，此时使用 break 跳出循环。

随后进入下一个 if 语句，首先判断 index 的值是否发生了变化，如果有变化（不等于 −1 时），说明发现了 null 的元素，“phones[index]=”iPhone5S”;”因为 index 在上一个 if 语句中已经重新赋值为 null 的下标值，这时直接找到那个空的位置赋值为“iPhone5S”即可。

输出结果如图 4.4 所示。

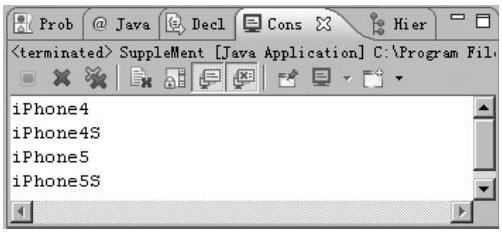


图 4.4　数组添加输出

（2）数组修改

示例 8

如图 4.5 所示，当已经存在一个数组“phones”时，如何修改“iPhone5”的值呢？大致思路是首先查找位置，然后进行修改。

```
String[] phones = {"iPhone3GS 经典","iPhone4 革新","iPhone4S 变化不大","iPhone5"};
```

图 4.5　数组修改

关键代码：

```
public class SuppleMent{
    public static void main(String[] args){
        // 数组修改
        int indexNew=-1;
        String[] phones = {"iPhone3GS 经典 ","iPhone4 革新 ","iPhone4S 变化不大 ",
"iPhone5"};
        for(int i=0;i<phones.length;i++){
            if(phones[i].equals("iPhone5")){//equals( ) 方法用来比较值是否相等
                indexNew=i;
                break;
            }
        }
        if(indexNew!=-1){
            phones[indexNew]="iPhone5 掉漆 ";
            for(int i=0;i<phones.length;i++){
                System.out.println(phones[i]);
            }
        }else{
                System.out.print(" 不存在 iPhone5");
        }
    }
}
```

分析：

第一个 if 语句的作用与数据添加类似，第二个 if 语句的作用是找到修改的位置，对该位置重新赋值。

输出结果如图 4.6 所示。

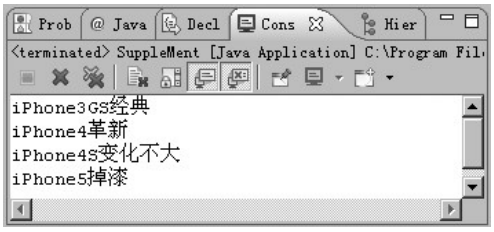


图 4.6　数组修改输出

（3）数组删除

示例 9

如图 4.7 所示，当已经存在一个数组“phones”时，如何删除“iPhone3GS 经典”的值呢？大致思路是首先找到删除的位置，删除后把后面的数据依次前移，将最后一位设置为 null。

```
String[] phones = {"iPhone3GS经典","iPhone4革新","iPhone4S变化不大","iPhone5掉漆"};
```

图 4.7　数组删除

关键代码：

```
public class SuppleMent{
    public static void main(String[] args){
        // 数组删除
        String[] phones = {"iPhone3GS 经典 ","iPhone4 革新 ","iPhone4S 变化不大 ",
"iPhone5掉漆"};
        int index=-1;
        for(int i=0;i<phones.length;i++){
            if(phones[i].equals("iPhone3GS 经典 ")){
                index=i;
                break;
            }
        }
        if(index!=-1){
            for(int i=index;i<phones.length-1;i++){
                phones[i]=phones[i+1];
            }
            phones[phones.length-1]=null;
        }else{
            System.out.println(" 没有您要删除的内容！ ");
        }
        for(int k = 0;k<phones.length;k++){
            System.out.println(phones[k]);
        }
    }
}
```

“phones [i] = phones [i+1];”表示此程序从 0 位置开始把 1 位置的值向前移一位，“phones.length-1”等于 3，当 i 的值等于 3 的时候，停止 for 循环，这时把最后一位赋值为 null，此时数组中的“iPhone3GS”被删除，后面的数据也完成了移位。

输出结果如图 4.8 所示。

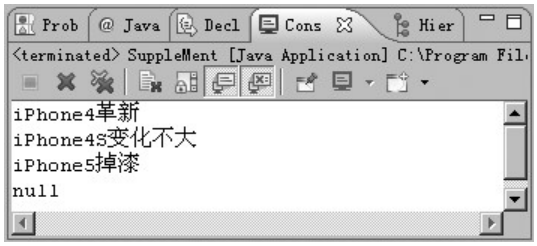


图 4.8　数组删除输出

4.1.2　常见问题

数组是编程中常用的存储数据的结构，但在使用的过程中常出现一些错误，这里做一个归纳，希望能够引起重视。

示例 10

请指出以下代码中出错的位置。

```
class ArrayTest4 {
    public static void main(String[] args) {
        int a[] = new int[] { 1, 2, 3, 4, 5 };
        System.out.println(a[5]);
    }
}
```

输出结果如图 4.9 所示。

图 4.9　数组下标越界

系统提示出现数组下标越界异常，并指出了错误语句的位置。发生异常的原因是 a 数组的下标最大值是 4，不存在下标为 5 的元素。

注意

数组下标从 0 开始，而不是从 1 开始。如果访问数组元素时指定的下标小于 0，或者大于等于数组的长度，都将出现数组下标越界异常。

示例 11

请指出以下代码中出错的位置。

关键代码：

```
class ArrayTest5 {
    public static void main(String[] args) {
        int arr1[4];
        arr1={1,2,3,4};
        int [] arr2=new int[4]{1,2,3,4};
    }
}
```

分析如下。

示例 11 的代码存在两处错误，均是初始化数组格式的错误。

正确的初始化数组格式如下：

```
int arr1[] = {1,2,3,4};                    // 不可分成两步初始化
int [] arr2 = new int[]{1,2,3,4};          // new int 后边的 [] 中必须为空
```

任务 2 计算学生总成绩

关键步骤如下。

- 初始化一个 int 类型二维数组。
- 定义一个 int 类型变量，用于保存总成绩。
- 通过嵌套循环遍历二维数组，并累加成绩。

4.2.1 二维数组

Java 中定义和操作多维数组的语法与一维数组类似。在实际应用中，三维及以上的数组很少使用，主要使用二维数组。下面主要以二维数组为例进行讲解。

定义二维数组的语法格式如下。

```
数据类型 [][] 数组名 ;
```

或者：

```
数据类型 数组名 [][];
```

- 数据类型为数组元素的类型。
- “[][]”用于表明定义了一个二维数组，通过多个下标进行数据访问。

示例 12

定义一个整型二维数组。

关键代码：

```
int[][] scores;                    // 定义二维数组
scores=new int[5][50];             // 分配内存空间
```

```
// 或者
int[][] scores = new int[5][50];
```

需要强调的是，虽然从语法上看 Java 支持多维数组，但从内存分配原理的角度看，Java 中只有一维数组，没有多维数组。或者说，表面上是多维数组，实质上都是一维数组。

示例 13

定义一个整型二维数组，并为其分配内存空间。

关键代码：

```
int[][] s = new int[3][5];
```

示例 13 中的语句表面看来是定义了一个二维数组，但是从内存分配原理角度，实际上是定义了一个一维数组，数组名是 s，包括 3 个元素，分别为 s[0]、s[1]、s[2]，每个元素是整型数组类型，即一维数组类型。而 s[0] 又是一个数组的名称，包括 5 个元素，分别为 s[0][0]、s[0][1]、s[0][2]、s[0][3]、s[0][4]，每个元素都是整数类型。s[1]、s[2] 与 s[0] 的情况相同，其存储方式如图 4.10 所示。

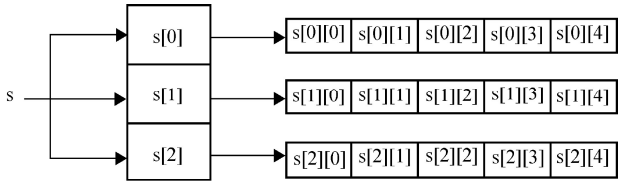


图 4.10　二维数组存储方式示意图

二维数组实际上是一个一维数组，它的每个元素又是一个一维数组。

4.2.2　二维数组的使用

1. 初始化二维数组

二维数组也可以进行初始化操作，与一维数组类似，同样可采用两种方式，请注意大括号的结构及书写顺序。

示例 14

定义二维数组并初始化数组元素的两种方法。

关键代码：

```
int[][] scores=new int[][]{ { 90, 85, 92, 78, 54 }, { 76, 63,80 }, { 87 }};
// 或者
int scores[][] = {{ 90, 85, 92, 78, 54 }, { 76, 63,80 }, { 87 } };
```

2. 二维数组的遍历

示例 15

分别计算每个班级的学生总成绩。

实现步骤如下。

（1）初始化整型二维数组。

（2）定义保存总成绩的变量。

（3）使用 for 循环遍历二维数组。

关键代码：

```
public static void main(String[] args) {
   int [][] array = new int[][]{{80,66},{70,54,98},{77,59}};
                                                    //定义二维数组、分配空间、赋值
   int total;                                       // 保存总成绩
   for(int i = 0; i < array.length; i++) {
         String str = (i+1) + " 班 ";
         total = 0;                                 // 每次循环到此都将其归 0
         for(int j = 0; j < array[i].length; j++) {
            total += array[i][j];                   // 成绩累加
         }
         System.out.println(str+" 总成绩 ：  " + total);
   }
}
```

输出结果：

```
1 班总成绩 ：  146
2 班总成绩 ：  222
3 班总成绩 ：  136
```

本任务运行效果如图 4.11 所示。

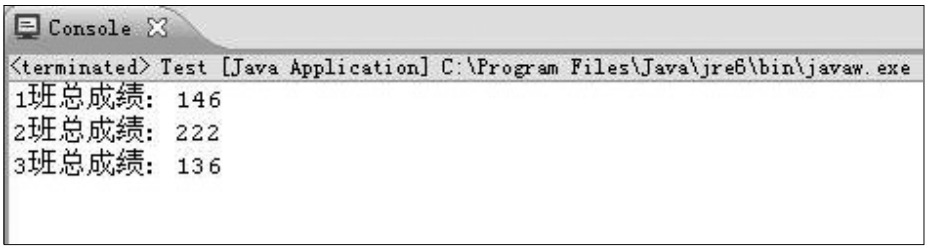


图 4.11　计算每个班级的学生总成绩

任务 3　升序排列学生成绩

关键步骤如下：

- 初始化一个整型二维数组。

- 使用 for 循环遍历二维数组。
- 使用 Arrays 类的 sort() 方法对二维数组的元素进行升序排列。
- 使用 for 循环遍历二维数组的元素并输出。

4.3.1　Arrays 类

JDK 中提供了一个专门用于操作数组的工具类，即 Arrays 类，位于 java.util 包中。该类提供了一系列方法来操作数组，如排序、复制、比较、填充等，用户直接调用这些方法即可，不需自己编码实现，降低了开发难度。Arrays 类的常用方法如表 4-2 所示。

表4-2　Arrays类的常用方法介绍

方法	返回类型	说明
equals(array1,array2)	boolean	比较两个数组是否相等
sort(array)	void	对数组array的元素进行升序排列
toString(array)	String	将一个数组array转换成一个字符串
fill(array,val)	void	把数组array的所有元素都赋值为val
copyOf(array,length)	与array数据类型一致	把数组array复制成一个长度为length的新数组
binarySearch(array, val)	int	查询元素值val在数组array中的下标

4.3.2　Arrays 类的应用

1. 比较两个数组是否相等

Arrays 类的 equals( ) 方法用于比较两个数组是否相等。只有当两个数组长度相等，对应位置的元素也一一相等时，该方法返回 true；否则返回 false。

示例 16

初始化 3 个整型一维数组，使用 Arrays 类的 equals() 方法判断是否两两相等，并输出比较结果。

实现步骤如下。

（1）初始化 3 个整型一维数组。

（2）使用 Arrays 类的 equals() 方法判断是否两两相等。

关键代码：

```
public static void main(String[] args) {
   int [] arr1 = {10,50,40,30};
   int [] arr2 = {10,50,40,30};
   int [] arr3 = {60,50,85};
  System.out.println(Arrays.equals(arr1, arr2)); // 判断 arr1 与 arr2 的长度及元素是否相等
  System.out.println(Arrays.equals(arr1, arr3)); // 判断 arr1 与 arr3 的长度及元素是否相等
}
```

输出结果：

```
true
false
```

2. 对数组的元素进行升序排列

Arrays 类的 sort() 方法对数组的元素进行升序排列，即以从小到大的顺序排列。

示例 17

分别对 1 班、2 班和 3 班的学员成绩进行升序排列。

实现步骤：

（1）初始化一个整型二维数组。

（2）使用 for 循环遍历二维数组。

（3）使用 Arrays 类的 sort() 方法对二维数组的元素进行升序排列。

（4）使用 for 循环遍历二维数组的元素并输出。

关键代码：

```
public static void main(String[] args) {
   int [][] array = new int[][]{{80,66},{70,54,98},{77,59}};
   for(int i = 0; i < array.length; i++) {
      String str = (i+1) + " 班 ";
      Arrays.sort(array[i]);
      System.out.println(str+" 成绩排序后 ： ");
      for(int j = 0; j < array[i].length; j++) {
         System.out.println(array[i][j]);
      }
   }
}
```

输出结果：

```
1 班成绩排序后 ：
66
80
2 班成绩排序后 ：
54
70
98
3 班成绩排序后 ：
59
77
```

本任务运行效果如图 4.12 所示。

3. 将数组转换成字符串

Arrays 类中提供了专门输出数组内容的方法——toString() 方法。该方法用于将一

个数组转换成一个字符串。它按顺序把多个数组元素连在一起，多个数组元素之间使用英文逗号和空格隔开。利用这种方法可以很清楚地观察到各个数组元素的值。

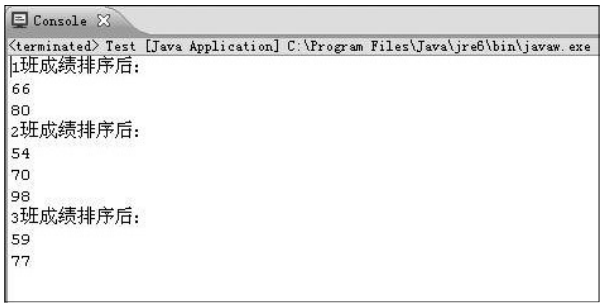


图 4.12　按升序排列每个班级的学生成绩

示例 18

初始化一个整型一维数组，使用 Arrays 类的 toString() 方法将数组转换为字符串并输出。

实现步骤如下。

（1）初始化一个整型一维数组。

（2）使用 Arrays 类的 toString() 方法将数组转换为字符串。

关键代码：

```
public static void main(String[] args) {
    int[] arr = { 10, 50, 40, 30 };
    Arrays.sort(arr);                              // 将数组按升序排列
    System.out.println(Arrays.toString(arr));      // 将数组 arr 转换为字符串并输出
}
```

输出结果：

```
[10, 30, 40, 50]
```

4. 将数组所有元素赋值为相同的值

Arrays 类的 fill(array,val) 方法用于把数组 array 的所有元素都赋值为 val。

示例 19

初始化一个整型一维数组，使用 Arrays 类的 fill() 方法替换数组的所有元素为相同的元素。

实现步骤如下。

（1）初始化一个整型一维数组。

（2）使用 Arrays 类的 fill() 方法替换数组元素。

关键代码：

```
public static void main(String[] args) {
    int[] arr = { 10, 50, 40, 30 };                // 初始化整型数组
```

```
    Arrays.fill(arr, 40);                          // 替换数组元素
    System.out.println(Arrays.toString(arr));      // 将数组 arr 转换为字符串并输出
}
```

输出结果：

```
[40, 40, 40, 40]
```

5. 将数组复制成一个长度为设定值的新数组

示例 20

初始化一个整型一维数组，使用 Arrays 类的 copyOf() 方法把数组复制成一个长度为设定值的新数组。

实现步骤如下。

（1）初始化一个长度为 4 的整型一维数组。

（2）使用 Arrays 类的 copyOf() 方法复制成一个长度为 3 的新数组，并输出新数组元素。

（3）使用 Arrays 类的 copyOf() 方法复制成一个长度为 4 的新数组，并输出新数组元素。

（4）使用 Arrays 类的 copyOf() 方法复制成一个长度为 6 的新数组，并输出新数组元素。

关键代码：

```
public static void main(String[] args) {
    int[] arr1 = { 10, 50, 40, 30 };
    int[] arr2 = Arrays.copyOf(arr1, 3);  // 将 arr1 复制成长度为 3 的新数组 arr2
    System.out.println(Arrays.toString(arr2));
    int[] arr3 = Arrays.copyOf(arr1, 4);  // 将 arr1 复制成长度为 4 的新数组 arr3
    System.out.println(Arrays.toString(arr3));
    int[] arr4 = Arrays.copyOf(arr1, 6);  // 将 arr1 复制成长度为 6 的新数组 arr4
    System.out.println(Arrays.toString(arr4));
}
```

输出结果：

```
[10, 50, 40]
[10, 50, 40, 30]
[10, 50, 40, 30, 0, 0]
```

Arrays 类的 copyOf(array,length) 方法可以进行数组复制，把原数组复制成一个新数组，其中 length 是新数组的长度。如果 length 小于原数组的长度，则新数组就是原数组的前面 length 个元素；如果 length 大于原数组的长度，则新数组前面的元素就是原数组的所有元素，后面的元素是按数组类型补充默认的初始值，如整型补充 0，浮点型补充 0.0 等。

6. 查询元素在数组中的下标

Arrays 类的 binarySearch() 方法用于查询数组元素在数组中的下标。调用该方法时

要求数组中的元素已经按升序排列，这样才能得到正确的结果。

示例 21

初始化一个整型数组，使用 Arrays 类的 binarySearch() 方法查询数组元素在数组中的下标。

实现步骤如下。

（1）初始化一个整型数组。

（2）使用 Arrays 类的 sort() 方法按升序排列数组。

（3）使用 Arrays 类的 binarySearch() 方法查找某个元素的下标，并输出。

关键代码：

```
public static void main(String[] args) {
    int[] arr = { 10, 50, 40, 30 };
    Arrays.sort(arr);                          // 先按升序排列
    int index=Arrays.binarySearch(arr, 30);    // 查找 30 的下标
    System.out.println(index);
    index=Arrays.binarySearch(arr, 50);        // 查找 50 的下标
    System.out.println(index);
}
```

输出结果：

```
1
3
```

本章总结

本章介绍了以下知识点。

- 数组是可以在内存中连续存储多个元素的结构。数组中的所有元素必须属于相同的数据类型。
- 数组中的元素通过数组下标进行访问，数组下标从 0 开始。
- 二维数组实际上是一个一维数组，它的每个元素是一个一维数组。
- 使用 Arrays 类提供的方法可以方便地对数组中的元素进行排序、查询等操作。
- JDK 1.5 之后提供了增强 for 循环，可以用来实现对数组和集合中数据的访问。

本章练习

1．依次输入 5 句话后将它们逆序输出，输出结果如图 4.13 所示。

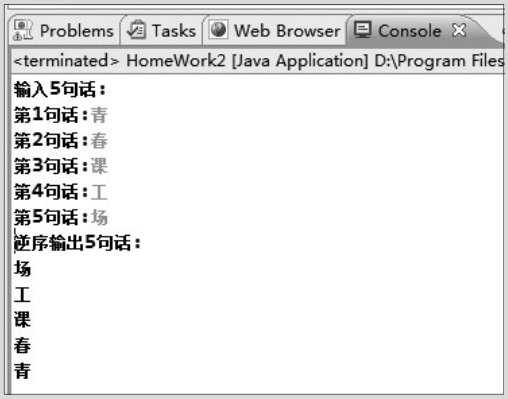


图 4.13　逆序输出

2．假设有一个长度为 5 的数组，如下所示：

```
int[] array=new int[]{1,3,-1,5,-2};
```

现要创建一个新数组 newArray[]，要求新数组中的元素是对原数组中的元素升序排列后所得。编程输出新数组中的元素，输出结果如图 4.14 所示。

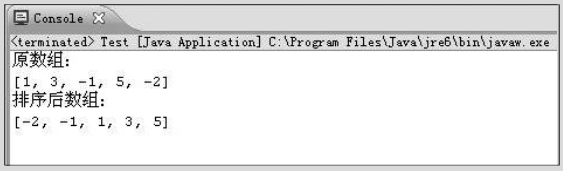


图 4.14　数组排序

3．用键盘输入 10 个数，合法数字为 1、2 或 3，不是这 3 个数则为非法数字。编程统计每个合法数字和非法数字的个数，输出结果如图 4.15 所示。

图 4.15　统计数字个数