第 6 章 数组、排序和查找

在程序设计中，经常需要处理一组相关的数据。例如，记录一个班级学生的成绩、存储一系列商品的价格等。为了更高效地管理和操作这些数据，编程语言提供了数组这一重要的数据结构。同时，排序和查找算法也是处理数据时常用的操作，它们能帮助我们从大量数据中快速找到所需信息或对数据进行整理。本章将详细介绍数组的概念、使用方法，以及常见的排序和查找算法。

6.1 为什么需要数组

先来看一个简单的例子。一个养鸡场有 6 只鸡，它们的体重分别是 3kg, 5kg, 1kg, 3.4kg, 2kg, 50kg 。现在需要计算这六只鸡的总体重和平均体重。

6.1.1 传统方式的不足

如果不使用数组，我们可能会定义 6 个变量来存储这 6 只鸡的体重，然后将它们相加得到总体重，再除以 6 得到平均体重。示例代码如下：

```
public class Array01 {
    public static void main(String[] args) {
        double hen1 = 3;
        double hen2 = 5;
        double hen3 = 1;
        double hen4 = 3.4;
        double hen5 = 2;
        double hen6 = 50;
        double totalWeight = hen1 + hen2 + hen3 + hen4 + hen5 + hen6;
        double avgWeight = totalWeight / 6;
        System.out.println("总体重=" + totalWeight + "平均体重=" + avgWeight);
    }
}
```

这种方式虽然能解决当前问题，但存在明显的不足。如果鸡的数量不是 6 只，而是 600 只甚至更多，难道要定义 600 个变量吗？显然，这种做法既繁琐又不现实。这就引出了数组的概念，数组可以很好地解决这类问题。

6.1.2 数组介绍

数组可以存放多个同一类型的数据。数组也是一种数据类型，属于引用类型。简单来说，数组就是一组数据的集合。例如，可以用数组来解决上述养鸡场鸡体重的问题。示例代码如下：

```
public class Array01 {
    public static void main(String[] args) {
        double[] hens = {3, 5, 1, 3.4, 2, 50, 7.8, 88.8, 1.1, 5.6, 100};
        double totalWeight = 0;
        for (int i = 0; i < hens.length; i++) {
            totalWeight += hens[i];
        }
        System.out.println("总体重=" + totalWeight + "平均体重=" + (totalWeight /
hens.length));
    }
}
```

在这段代码中，`double[] hens` 表示定义了一个 `double` 类型的数组 `hens`，`{3, 5, 1, 3.4, 2, 50, 7.8, 88.8, 1.1, 5.6, 100}` 是数组的值 / 元素。通过 `hens[下标]` 可以访问数组的元素，下标从 0 开始编号，例如 `hens[0]` 表示第一个元素，`hens[1]` 表示第二个元素，依次类推。利用 `for` 循环可以方便地遍历数组的所有元素，从而实现计算总体重和平均体重的功能。

6.2 数组的使用

6.2.1 动态初始化方式 1

数组有多种使用方式，首先介绍动态初始化的第一种方式。语法如下：

```
数据类型 数组名[] = new 数据类型[大小];
```

例如，要创建一个用于存储 5 个学生成绩的 `double` 类型数组，可以这样写：

```
double scores[] = new double[5];
```

下面通过一个完整的示例，演示如何循环输入 5 个成绩，保存到 `double` 数组，并输出。

```
import java.util.Scanner;
public class Array02 {
    public static void main(String[] args) {
        double scores[] = new double[5];
        Scanner myScanner = new Scanner(System.in);
        for (int i = 0; i < scores.length; i++) {
            System.out.println("请输入第" + (i + 1) + "个元素的值");
            scores[i] = myScanner.nextDouble();
        }
        System.out.println("==数组的元素/值的情况如下:===");
        for (int i = 0; i < scores.length; i++) {
            System.out.println("第" + (i + 1) + "个元素的值=" + scores[i]);
        }
    }
}
```

在这个示例中，首先创建了一个大小为 5 的 `double` 类型数组 `scores`。然后，使用 `Scanner` 类从控制台读取用户输入的 5 个成绩，并将它们依次存入数组中。最后，通过 `for` 循环遍历数组，将数组中的所有成绩输出。

6.2.2 动态初始化方式 2

动态初始化的第二种方式是先声明数组，再创建数组。语法如下：

- 声明数组：

```
数据类型 数组名[]; 或者 数据类型[] 数组名;
```

例如：

```
int a[]; 或者 int[] a;
```

- 创建数组：

```
数组名 = new 数据类型[大小];
```

例如：

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

下面是一个示例：

```
public class Array02 {
    public static void main(String[] args) {
        double scores[];
        scores = new double[5];
        Scanner myScanner = new Scanner(System.in);
        for (int i = 0; i < scores.length; i++) {
            System.out.println("请输入第" + (i + 1) + "个元素的值");
            scores[i] = myScanner.nextDouble();
        }
        System.out.println("==数组的元素/值的情况如下:===");
        for (int i = 0; i < scores.length; i++) {
            System.out.println("第" + (i + 1) + "个元素的值=" + scores[i]);
        }
    }
}
```

这段代码与前面的示例功能相同，只是采用了先声明后创建数组的方式。

6.2.3 静态初始化

静态初始化是在定义数组的同时为数组元素赋值。语法如下：

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

例如：

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

下面是一个完整的静态初始化示例：

```
public class ArrayStaticInit {
    public static void main(String[] args) {
        int[] numbers = {10, 20, 30, 40, 50};
        for (int i = 0; i < numbers.length; i++) {
            System.out.println("数组元素: " + numbers[i]);
        }
    }
}
```

在这个示例中，定义了一个 `int` 类型的数组 `numbers`，并通过静态初始化的方式为其赋值。然后，使用 `for` 循环遍历数组并输出每个元素。

6.3 数组使用注意事项和细节

1. **数组元素类型一致性**：数组是多个相同类型数据的组合，实现对这些数据的统一管理。例如：

```
int[] arr1 = {1, 2, 3, 4, 5};//正确，元素都是int类型
//int[] arr2 = {1, 2, 3, "hello"};//错误，元素类型不一致，不能混用
```

1. **数组元素类型范围**：数组中的元素可以是任何数据类型，包括基本类型和引用类型，但不能混用。
   例如：

```
double[] arr3 = {1.1, 2.2, 3.3, 4.4, 5.5};//基本类型数组
String[] arr4 = {"apple", "banana", "cherry"};//引用类型数组
```

1. 数组默认值
   ：数组创建后，如果没有赋值，会有默认值。不同数据类型的默认值如下：
   - `int`: 0
   - `short`: 0
   - `byte`: 0
   - `long`: 0
   - `float`: 0.0
   - `double`: 0.0
   - `char`: `\u0000`
   - `boolean`: `false`
   - `String`: `null`
     例如：

```
short[] arr5 = new short[3];
for (int i = 0; i < arr5.length; i++) {
    System.out.println(arr5[i]);//输出默认值0
}
```

1. 使用数组的步骤
   ：
   - 声明数组并开辟空间。
   - 给数组各个元素赋值。
   - 使用数组。
     例如：

```
int[] arr6;
arr6 = new int[5];
arr6[0] = 10;
arr6[1] = 20;
//……
for (int i = 0; i < arr6.length; i++) {
    System.out.println(arr6[i]);
}
```

1. **数组下标范围**：数组的下标是从 0 开始的。例如，对于一个大小为 5 的数组 `int[] arr = new int[5];`，有效下标为 0 - 4。
2. **防止下标越界**：数组下标必须在指定范围内使用，否则会报下标越界异常。例如：

```
int[] arr7 = new int[5];
//System.out.println(arr7[5]);//会抛出ArrayIndexOutOfBoundsException异常
```

1. **数组的引用类型本质**：数组属引用类型，数组型数据是对象（object）。例如：

```
int[] arr8 = {1, 2, 3};
int[] arr9 = arr8;
```

在这个例子中，`arr9` 赋值为 `arr8`，实际上是将 `arr8` 的地址赋给了 `arr9`，它们指向同一块内存空间。

6.4 数组应用案例

6.4.1 创建并遍历字符数组

创建一个 `char` 类型的 26 个元素的数组，分别放置 'A' - 'Z'。使用 `for` 循环访问所有元素并打印出来。

```
public class ArrayExercise01 {
    public static void main(String[] args) {
        char[] chars = new char[26];
        for (int i = 0; i < chars.length; i++) {
            chars[i] = (char) ('A' + i);
        }
        System.out.println("===chars数组===");
        for (int i = 0; i < chars.length; i++) {
            System.out.print(chars[i] + " ");
        }
    }
}
```

在这段代码中，首先定义了一个大小为 26 的 `char` 类型数组 `chars`。然后，通过 `for` 循环利用 `'A' + i` 的运算，将 'A' 到 'Z' 的字符依次存入数组中。最后，再次使用 `for` 循环遍历数组并将所有字符打印出来。

6.4.2 求数组的最大值及下标

请求出一个数组 `int[]` 的最大值 `{4, -1, 9, 10, 23}`，并得到对应的下标。

```
public class ArrayExercise02 {
    public static void main(String[] args) {
        int[] arr = {4, -1, 9, 10, 23};
        int max = arr[0];
        int maxIndex = 0;
        for (int i = 1; i < arr.length; i++) {
            if (max < arr[i]) {
                max = arr[i];
                maxIndex = i;
            }
        }
        System.out.println("max=" + max + " maxIndex=" + maxIndex);
    }
}
```

代码中，首先假定数组的第一个元素 `arr[0]` 为最大值，并记录其下标 `maxIndex = 0`。然后，从下标 1 开始遍历数组，将当前元素与 `max` 进行比较。如果当前元素大于 `max`，则更新 `max` 为当前元素，并记录其下标。当遍历完整个数组后，`max` 即为数组的最大值，`maxIndex` 为最大值对应的下标。

10

1 3 5 6 7 2 2 6 7 10

6.4.3 求数组的和与平均值

请求出一个数组的和和平均值，例如养鸡场鸡体重数组。

```
public class ArraySumAndAvg {
    public static void main(String[] args) {
        double[] hens = {3, 5, 1, 3.4, 2, 50};
        double sum = 0;
        for (int i = 0; i < hens.length; i++) {
            sum += hens[i];
        }
        double avg = sum / hens.length;
        System.out.println("数组的和=" + sum + " 平均值=" + avg);
    }
}
```

这段代码通过 `for` 循环遍历数组 `hens`，将每个元素累加到变量 `sum` 中，得到数组的总和。然后，将总和除以数组的长度，得到平均值。最后，输出数组的和与平均值。

6.5 数组赋值机制

6.5.1 基本数据类型赋值

基本数据类型赋值时，这个值就是具体的数据，而且相互不影响。例如：

```
int n1 = 2;
int n2 = n1;
n2 = 5;
System.out.println("n1 = " + n1); //输出2
System.out.println("n2 = " + n2); //输出5
```

在这个例子中，`n1` 赋值为 2，然后将 `n1` 的值赋给 `n2`。之后修改 `n2` 的值为 5，`n1` 的值不受影响。

6.5.2 数组赋值（引用传递）

数组在默认情况下是引用传递，赋的值是地址。例如：

```
int[] arr1 = {1, 2, 3};
int[] arr2 = arr1;
arr2[0] = 100;
System.out.println("arr1[0] = " + arr1[0]); //输出100
```

在这段代码中，`arr2` 赋值为 `arr1`，实际上是将 `arr1` 的地址赋给了 `arr2`，它们指向同一块内存空间。因此，当修改 `arr2` 的第一个元素时，`arr1` 的第一个元素也会随之改变。

6.6 数组拷贝

编写代码实现数组拷贝（内容复制），要求将 `int[] arr1 = {10, 20, 30};` 拷贝到 `arr2` 数组，且数据空间是独立的。

```
public class ArrayCopy {
    public static void main(String[] args) {
        int[] arr1 = {10, 20, 30};
        int[] arr2 = new int[arr1.length];
        for (int i = 0; i < arr1.length; i++) {
            arr2[i] = arr1[i];
        }
        arr2[0] = 100;
        System.out.println("====arr1的元素====");
        for (int i = 0; i < arr1.length; i++) {
            System.out.println(arr1[i]);//输出10, 20, 30
        }
        System.out.println("====arr2的元素====");
        for (int i = 0; i < arr2.length; i++) {
            System.out.println(arr2[i]);//输出100, 20, 30
        }
    }
}
```

在这个示例中，首先创建了一个与 `arr1` 大小相同的新数组 `arr2`。然后，通过 `for` 循环将 `arr1` 的每个元素依次拷贝到 `arr2` 对应的位置。这样，`arr1` 和 `arr2` 虽然内容相同，但它们的数据空间是独立的。当修改 `arr2` 的第一个元素为 100 时，`arr1` 的元素不受影响。

6.7 数组反转

6.7.1 方式 1：通过找规律反转

要求把数组 `arr {11, 22, 33, 44, 55, 66}` 反转成 `{66, 55, 44, 33, 22, 11}`。

```
public class ArrayReverse {
    public static void main(String[] args) {
        int[] arr = {11, 22, 33, 44, 55, 66};
        int temp = 0;
        int len = arr.length;
        for (int i = 0; i < len / 2; i++) {
            temp = arr[len - 1 - i];
            arr[len - 1 - i] = arr[i];
            arr[i] = temp;
        }
        System.out.println("===翻转后数组===");
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + "\t");
        }
    }
}
```

练习题

一、选择题

1. 以下关于数组的声明，正确的是（）
   A. int arr;
   B. int [] arr = new int;
   C. int [] arr = new int [5];
   D. int arr [5];
2. 数组 `int[] arr = {1, 2, 3, 4, 5};` 中，`arr[3]` 的值是（）
   A. 3
   B. 4
   C. 5
   D. 2
3. 数组创建后，如果没有赋值，`double` 类型数组的默认值是（）
   A. 0
   B. 0.0
   C. null
   D. false
4. 以下代码输出结果是（）

```
int[] arr = {1, 2, 3};
int[] arr2 = arr;
arr2[1] = 100;
System.out.println(arr[1]);
```

A. 2
B. 100
C. 3
D. 1
\5. 定义一个存储 10 个 `String` 类型元素的数组，正确的是（ ）
A. String [] arr = new String ();
B. String arr [] = new String [10];
C. String [] arr = new String [10] {};
D. String arr [] = new String;
\6. 数组的下标从（ ）开始
A. 0
B. 1
C. -1
D. 随机
\7. 以下能正确创建并初始化一个字符数组，存放 'a' - 'e' 的是（ ）
A. char [] arr = {'a', 'b', 'c', 'd', 'e'};
B. char [] arr = new char [5]; arr = {'a', 'b', 'c', 'd', 'e'};
C. char [] arr = new char [5] {'a', 'b', 'c', 'd', 'e'};
D. char arr [] = new char {'a', 'b', 'c', 'd', 'e'};
\8. 数组属于（ ）类型
A. 基本数据类型
B. 引用类型
C. 原始数据类型
D. 以上都不对
\9. 要获取数组 `int[] arr = {1, 2, 3, 4, 5};` 的长度，以下正确的是（ ）
A. arr.length ()
B. arr.len
C. arr.length
D. length (arr)
\10. 以下代码运行时会抛出什么异常（ ）

```
int[] arr = new int[5];
System.out.println(arr[5]);
```

A. NullPointerException
B. ArrayIndexOutOfBoundsException
C. ClassCastException
D. NumberFormatException

二、填空题

1. 数组是多个相同__数据的组合，实现对这些数据的统一管理。
2. 动态初始化数组的第一种方式语法为：数据类型 数组名 [] = new 数据类型 [__];
3. 静态初始化数组的语法为：数据类型 数组名 [] = {__};
4. 数组创建后，如果没有赋值，`boolean` 类型数组的默认值是__。
5. 定义一个 `int` 类型数组 `arr`，并开辟 8 个元素空间的代码为：__。
6. 数组的下标必须在指定范围内使用，否则会报__异常。
7. 已知 `int[] arr = {1, 2, 3, 4, 5};`，将数组元素全部乘以 2 的代码如下：

```
for(int i = 0; i < arr.length; i++){
    arr[i] = ______;
}
```

1. 数组在默认情况下是__传递，赋的值是地址。
2. 要将数组 `int[] arr1 = {1, 2, 3};` 拷贝到 `arr2` 数组，且数据空间独立，首先要创建 `arr2` 数组，代码为：__。
3. 把数组 `int[] arr = {1, 2, 3, 4};` 反转成 `{4, 3, 2, 1}`，在反转循环中，循环条件可以设置为 `for(int i = 0; i < ______; i++)`。

三、简答题

1. 简述数组的作用以及与传统定义多个变量方式相比的优势。
2. 请说明数组动态初始化的两种方式，并各举一个例子。
3. 解释数组静态初始化的概念，并举例说明。
4. 数组创建后，如果没有赋值，不同数据类型的默认值分别是什么？请列举至少 5 种数据类型及其默认值。
5. 使用数组的一般步骤是什么？
6. 为什么数组下标从 0 开始？这对数组操作有什么影响？
7. 举例说明数组的引用类型本质，以及与基本数据类型赋值的区别。
8. 如何实现数组拷贝，使拷贝后的数组数据空间独立？请简述步骤。
9. 请描述将一个数组反转的思路（以 `int` 类型数组为例）。
10. 简述在 Java 中，数组元素类型一致性的要求及违反该要求的后果。

四、编程题

1. 创建一个 `int` 类型数组，存储 10 个随机整数（范围 1 - 100），然后输出数组中的所有元素。
2. 定义一个 `double` 类型数组，存储 5 个学生的成绩，计算并输出这 5 个学生成绩的总和与平均值。
3. 已知一个 `int` 类型数组 `int[] arr = {3, 5, 1, 7, 9};`，编写程序找出数组中的最小值及其下标。
4. 创建一个字符数组，存储字符串 "Hello World" 中的每个字符，然后使用 `for` 循环输出该数组中的所有字符。
5. 编写程序，将数组 `int[] arr = {1, 2, 3, 4, 5};` 中的元素顺序颠倒，即变为 `{5, 4, 3, 2, 1}`，并输出颠倒后的数组。
6. 定义一个数组 `int[] arr = {1, 2, 2, 3, 3, 3, 4, 4, 4, 4};`，编写程序统计数组中每个数字出现的次数，并输出结果（例如：1 出现 1 次，2 出现 2 次等）。
7. 编写代码，将两个 `int` 类型数组 `int[] arr1 = {1, 2, 3};` 和 `int[] arr2 = {4, 5, 6};` 合并成一个新数组 `int[] arr3 = {1, 2, 3, 4, 5, 6};`，并输出新数组。
8. 创建一个 `String` 类型数组，存储 3 个城市名称，然后使用 `for - each` 循环输出数组中的所有城市名称。
9. 已知一个 `int` 类型数组 `int[] arr = {10, 20, 30, 40, 50};`，编写程序将数组中所有元素的值都增加 10，然后输出修改后的数组。
10. 编写程序，从控制台读取 5 个整数，存入一个 `int` 类型数组，然后找出数组中的第二大的数并输出。