

上机实验 3

数 组

实验目的：

- ◆ 掌握一维数组的定义和初始化。
- ◆ 掌握一维数组的访问和使用。
- ◆ 掌握二维数组的定义和访问。

3.1 一维数组

数组是一种数据结构,是按一定顺序排列的相同类型的元素集合。数组实际上就是一连串类型相同的变量,这些变量用一个名字命名,即数组名,并用索引区分它们。使用数组时,可以通过索引来访问数组元素,如数组元素的赋值和取值。

实验 3-1 字母正序输出

【内容】

输入一个大写字母,以该字母为第一个字母,正序输出所有的大写字母。例如,输入 F 后,输出 FGHIGKLMNOPQRSTUVWXYZABCDE。如果输入的是其他字符,显示输入错误。

【思路】

(1) 使用 Scanner 类从键盘输入一个字符,使用 Scanner 类的 nextLine()方法读取一行字符,然后使用 charAt()方法获得第一个字符,存放在变量中。

(2) 判断这个字符是大写字母还是其他字符。如果是其他字符,则显示输入错误。

(3) 如果是大写字母,则以这个大写字母为开始,正序把其他大写字母存放在长度为 26 的数组中。

(4) 存在特殊情况,当存放到大写字母 Z 时,下次再存放时应该从大写字母 A 开始,直到数组存满为止。

【代码】

```
package three;
import java.util.Scanner;
public class CharTablesDemo {
```

```
public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    System.out.println("请输入一个字符:");
    String string=s.nextLine();
    char c=string.charAt(0);
    char chartables[]=new char[26];
    if(c>='A'&&c<='Z'){
        for(int i=0;i<chartables.length;){
            if(c<='Z'){
                chartables[i]=c;
                c++;
                i++;
            }
            else
                c='A';
        }
        System.out.println(chartables);
    }else{
        System.out.println("输入错误。");
    }
}
}
```

【运行结果】

输入一个字母 C 后按 Enter 键,以 C 开始正序输出所有的大写字母,如图 3-1 所示。

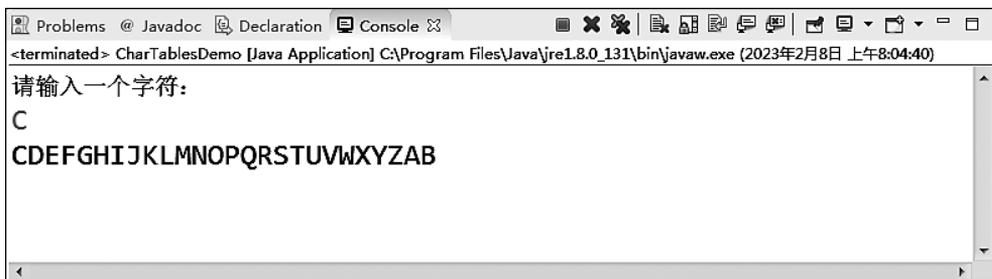


图 3-1 正序输出所有大写字母

实验 3-2 模拟双色球生成案例

【内容】

彩票中有一种是双色球,要求从 1~33 中随机选取 6 个号码作为红球,6 个号码不能重复,从 1~16 中随机选取 1 个号码作为蓝球,6 个红球和 1 个蓝球组合即为中奖号码。

【思路】

- (1) 定义一个长度为 6 的整型数组,初始值是 0。
- (2) 利用 Random 类中的 nextInt() 方法随机产生 6 个 1~33 中的整数,每生成一个随机数都要和已经存放在数组中的元素进行比较,如果这个数已经存在,则需要重新生成,再

次比较,直到生成的这个数没有在数组中出现,把这个数存放在数组中。生成的 6 个不相同的整数作为红球。

(3) 利用 Random 类中的 nextInt()方法随机产生 1 个 1~16 中的整数,作为蓝球。

【代码】

```
package three;
import java.util.Random;
public class BicolorBallDemo {
    public static void main(String[] args) {
        int a[]=new int[6];
        int count=0;
        boolean flag=true;
        for(int i=0;count<6;i++){
            int temp=new Random().nextInt(33)+1;
            for(int j=0;j<6;j++){
                if(temp==a[j]){
                    flag=false;
                }
            }
            if(flag){
                a[count]=temp;
                count++;
            }
            flag=true;
        }
        System.out.print("6 个蓝球分别是:");
        for(int i=0;i<6;i++){
            System.out.print(a[i]+" ");
        }
        System.out.println();
        System.out.print("1 个红球是:");
        System.out.println(new Random().nextInt(16)+1);
    }
}
```

【运行结果】

运行程序,如图 3-2 所示,随机生成一组双色球号码。

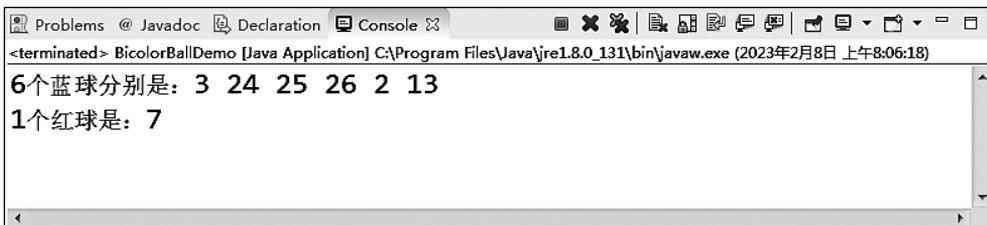


图 3-2 双色球号码

实验 3-3 数组乱序后输出

【内容】

给定一个数组,将这个数组的元素打乱顺序后输出。

【思路】

- (1) 使用 Scanner 类从键盘输入 5 个整数,存放在数组中。
- (2) 使用 for 循环,将数组中每个元素都与其他位置的元素进行交换,其他位置的元素利用 Random 类中的 nextInt()方法随机定位确定。
- (3) 最后输出打乱顺序后的数组。

【代码】

```
package three;
import java.util.Random;
import java.util.Scanner;
public class ShuffleArrayDemo {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("请输入 5 个整数:");
        int a[]=new int[5];
        for(int i=0;i<5;i++) {
            a[i]=s.nextInt();
        }
        System.out.print("输入的数组为:");
        for(int i=0;i<5;i++) {
            System.out.print (a[i]+" ");
        }
        System.out.println();
        for(int i=0;i<a.length;i++) {
            int temp=new Random().nextInt(a.length);
            int t=a[i];
            a[i]=a[temp];
            a[temp]=t;
        }
        System.out.print("乱序后的数组为:");
        for(int i=0;i<5;i++) {
            System.out.print (a[i]+" ");
        }
    }
}
```

【运行结果】

如图 3-3 所示,输入 5 个整数,存放在数组中,程序运行后数组乱序输出。

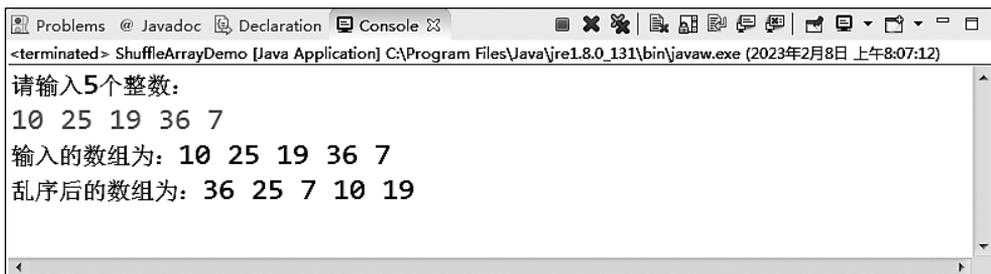


图 3-3 数组乱序输出

实验 3-4 数组中查找数据

【内容】

定义一个整型数组,输入一个整数,查找该整数在数组中第一次出现的位置,并输出该整数在数组中出现的总次数。如果没有找到该整数,则显示数组中没有要查找的数据。

【思路】

- (1) 定义一个整型数组,使用 Scanner 类输入一个整数。
- (2) 使用 for 循环将数组的第一个元素和输入的整数进行比较,如果相等,则返回这个元素的下标,并跳出循环;如果不相等则比较下一个元素,直至数组结束;若没有找到,则输出数组中没有要查找的数据。
- (3) 定义一个变量用来记载该整数和数组中元素相等的次数,使用 for 循环和数组中的每一个元素比较,若相等,则该变量加 1,直到数组结束;若没有相等元素,则变量的值为 0。

【代码】

```
package three;
import java.util.Scanner;
public class SimpleSearchDemo {
    public static void main(String[] args) {
        int a[]={20,17,65,28,40,125,69,28};
        Scanner s=new Scanner(System.in);
        System.out.println("请输入要查找的数据:");
        int data=s.nextInt();
        int b=-1;
        for(int i=0;i<a.length;i++) {
            if(a[i]==data) {
                b=i;
                break;
            }
        }
        int count=0;
        for(int i=0;i<a.length;i++) {
            if(a[i]==data) {
                count++;
            }
        }
    }
}
```

```
    }  
    }  
    if(b!=-1) {  
        System.out.println(data+"第一次出现在数组的第"+b+"个位置。");  
        System.out.println("这个数在数组中出现了"+count+"次。");  
    }else {  
        System.out.println("数组中没有要查找的数据。");  
    }  
    }  
}
```

【运行结果】

输入一个数据 69, 查找该数据在数组中的位置以及出现的次数, 如图 3-4 所示。

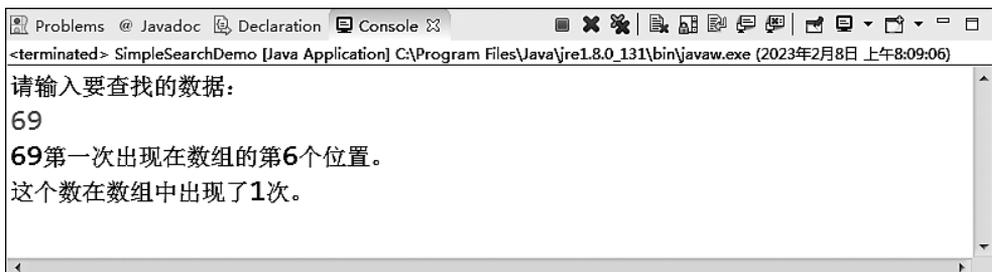


图 3-4 数组中查找数据

实验 3-5 在有序数组中插入数据

【内容】

定义一个有序的数组, 输入一个整数, 将其插入数组中, 并确保数组仍是有序的。

【思路】

(1) 定义一个整型有序的数组 a: {5, 16, 23, 35, 46, 59, 63, 78}, 定义一个整型数组 b, 长度是 a 的长度加 1, 使用 Scanner 类输入一个整数 data。

(2) 使用 for 循环查找 data 在数组 a 中的位置并记录到变量 index 中。

(3) 把数组 a 中下标为 0~index 的元素的值赋给数组 b 中下标为 0~index 的元素; 将数组 b 的下标为 index+1 的元素的值定义为 data; 将数组 a 的剩余元素的值赋给数组 b 中下标为 index+2~b.length-1 的元素。

【代码】

```
package three;  
import java.util.Scanner;  
public class InsertDataDemo {  
    public static void main(String[] args) {  
        int a[]={5, 16, 23, 35, 46, 59, 63, 78};  
        System.out.println("插入数据前的数组为:");  
        for(int i=0; i<a.length; i++) {
```

```
        System.out.print(a[i]+" ");
    }
    System.out.println();
    int b[]=new int[a.length+1];
    Scanner s=new Scanner(System.in);
    System.out.println("请输入要插入的数据:");
    int data=s.nextInt();
    int index=-1;
    for(int i=0;i<a.length;i++) {
        if(data>=a[i]) {
            index=i;
        }
    }
    for(int i=0;i<=index;i++) {
        b[i]=a[i];
    }
    b[index+1]=data;
    for(int i=index+2;i<b.length;i++){
        b[i]=a[i-1];
    }
    System.out.println("插入数据后的数组为:");
    for(int i=0;i<b.length;i++) {
        System.out.print(b[i]+" ");
    }
}
}
```

【运行结果】

输入一个数据 25,将 25 插入数组中并确保数组仍是有序的,如图 3-5 所示。

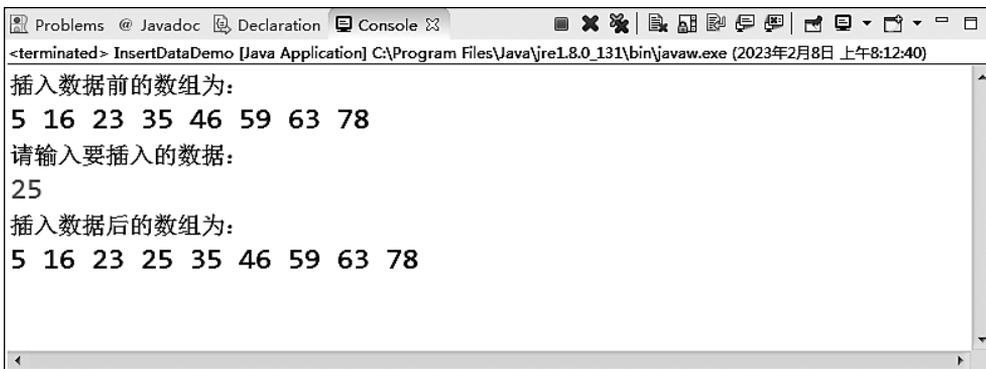


图 3-5 插入数据

实验 3-6 选择排序

【内容】

给定一个数组,使用选择排序法对数组进行从小到大的排序并输出。

【思路】

(1) 使用 Scanner 类从键盘输入 8 个整数,存放在数组中。

(2) 选择排序的过程是:对 8 个数进行选择排序需要 7 趟。第一趟从待排序的数据中找到最小的一个元素,存放在数组的第一个位置;第二趟再从剩余的待排序的数据中找到最小的一个元素,存放在数组的第二个位置;以此类推,直到第 7 趟排序从剩余的 2 个数中找到最小的元素,存放在数组的第 7 个位置,此时选择排序结束。

【代码】

```
package three;
import java.util.Scanner;
public class SelectSortDemo {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("请输入 8 个整数:");
        int a[]=new int[8];
        for(int i=0;i<8;i++) {
            a[i]=s.nextInt();
        }
        System.out.print("输入的数组为:");
        for(int i=0;i<8;i++) {
            System.out.print (a[i]+" ");
        }
        System.out.println();
        int index;
        for(int i=0;i<a.length-1;i++) {
            index=i;
            for(int j=i+1;j<a.length;j++) {
                if(a[j]<a[index]) {
                    index=j;
                }
            }
            if(index!=i) {
                int temp=a[i];
                a[i]=a[index];
                a[index]=temp;
            }
            System.out.println("第"+(i+1)+"趟选择排序的结果是:");
            for(int j=0;j<a.length;j++) {
                System.out.print(a[j]+" ");
            }
            System.out.println();
        }
        System.out.println("选择排序的最终结果是:");
        for(int i=0;i<a.length;i++) {
```

```

        System.out.print(a[i]+" ");
    }
}
}
}

```

【运行结果】

输入 8 个整数,存放到数组中,对数组进行选择排序,显示每趟选择排序的结果,如图 3-6 所示。

```

<terminated> SelectSortDemo [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (2023年2月8日 上午8:13:48)
请输入8个整数:
10 30 60 70 50 40 80 20
输入的数组为: 10 30 60 70 50 40 80 20
第1趟选择排序的结果是:
10 30 60 70 50 40 80 20
第2趟选择排序的结果是:
10 20 60 70 50 40 80 30
第3趟选择排序的结果是:
10 20 30 70 50 40 80 60
第4趟选择排序的结果是:
10 20 30 40 50 70 80 60
第5趟选择排序的结果是:
10 20 30 40 50 70 80 60
第6趟选择排序的结果是:
10 20 30 40 50 60 80 70
第7趟选择排序的结果是:
10 20 30 40 50 60 70 80
选择排序的最终结果是:
10 20 30 40 50 60 70 80

```

图 3-6 选择排序

3.2 二维数组

二维数组可以看成维数为 2 的数组,常用来表示表格或矩形。二维数组的声明、初始化与一维数组类似。

实验 3-7 杨辉三角形

【内容】

根据用户输入的整数,输出对应指定行数的杨辉三角形。例如,输入的是 5,则输出如图 3-7 所示的 5 行杨辉三角形。

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

```

图 3-7 5 行杨辉三角形

【思路】

(1) 杨辉三角形是多行多列的数据,采用二维数组来完成。杨辉三角形的规律是每行数字的第一列和最后一列的数字都是 1,从第三行开始,除去第一列和最后一列都为数字 1 以外,其余每列的数字都等于它上方和左上方两个数字之和。

(2) 使用 Scanner 类输入一个整数 n,创建 $n \times n$ 的二维数组。

(3) 为数组中的元素赋值。用 i 和 j 表示数组元素的行下标和列下标,赋值的操作分为两步:第一步先将每行的第一个元素和二维数组的对角线元素赋值为 1;第二步再对每行的第一个元素和对角线之间的元素赋值,这部分元素的行下标 $i=2 \sim n-1$,列下标 $j=1 \sim i-1$,此区域内的每个元素等于其上方和左上方的元素之和;数组中其他的元素可以不必赋值。

(4) 输出时只需输出二维数组的左下三角部分即可。

【代码】

```
package three;
import java.util.Scanner;
public class YangHuiTriangleDemo {
    public static void main(String[] args) {
        Scanner scn=new Scanner(System.in);
        System.out.println("请输入一个正整数:");
        int n=scn.nextInt();
        int arr[][]=new int[n][n];
        int i,j;
        for(i=0;i<n;i++){
            arr[i][0]=1;
            arr[i][i]=1;
        }
        for(i=2;i<n;i++){
            for(j=1;j<i;j++){
                arr[i][j]=arr[i-1][j-1]+arr[i-1][j];
            }
        }
        for( i=0;i<arr.length;i++){
            for( j=0;j<=i;j++){
                System.out.printf("%-3d",arr[i][j]);
            }
            System.out.println();
        }
    }
}
```

【运行结果】

输入一个整数 6,显示 6 行杨辉三角形,如图 3-8 所示。

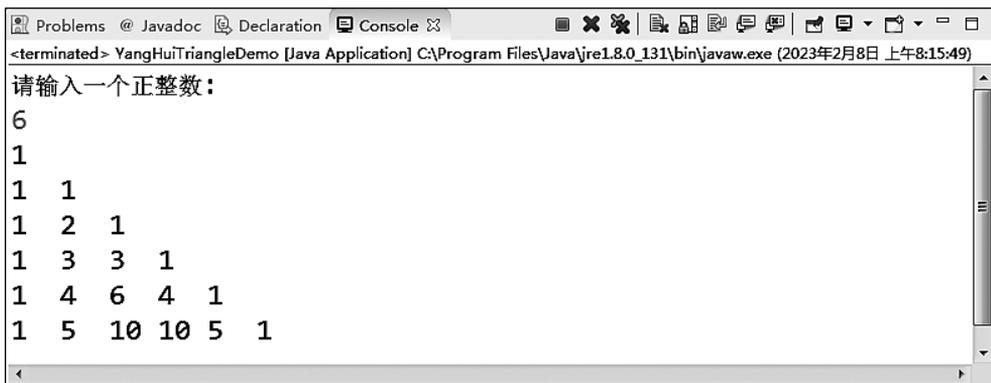


图 3-8 6 行杨辉三角形

实验 3-8 求矩阵的鞍点

【内容】

在矩阵中,一个元素在所在行中是最大值,在所在列中是最小值,则该元素称为这个矩阵的鞍点。

【思路】

(1) 定义一个 4×4 的二维数组,使用 `Math` 类中 `random()` 方法随机生成的数为二维数组中的每个元素赋值。

(2) 定义一个长度为 4 的一维数组 `temp1`,用于存储二维数组每行的最大值。

(3) 定义一个长度为 4 的一维数组 `temp2`,用于存储二维数组每列的最小值。

(4) 将 `temp1` 中的每个元素和 `temp2` 中的每个元素进行比较,如果相等,则这个值为矩阵的鞍点,获得这个元素的行值和列值;如果不相等,则矩阵没有鞍点。

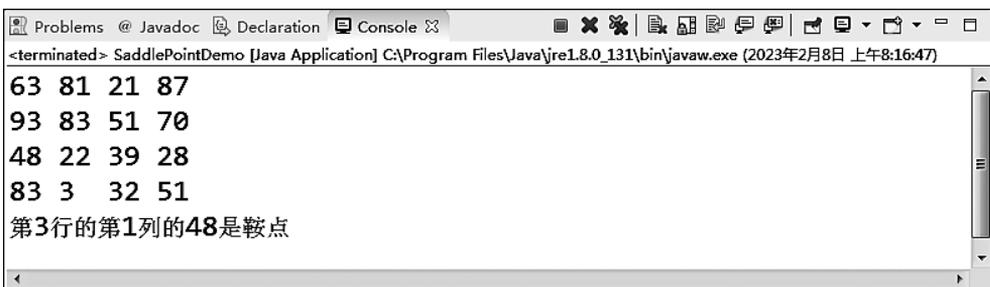
【代码】

```
package three;
public class SaddlePointDemo {
    public static void main(String[] args) {
        int arr[][]=new int[4][4];
        boolean flag=false;
        int row=0;
        int column=0;
        int value=0;
        for(int i=0;i<arr.length;i++){
            for(int j=0;j<arr[i].length;j++){
                arr[i][j]=(int)(Math.random()*100);
                System.out.printf("%-3d",arr[i][j]);
            }
            System.out.println();
        }
        int temp1[]=new int[4];
```

```
int temp2[]=new int[4];
for(int i=0;i<4;i++){
    int max=arr[i][0];
    for(int j=0;j<4;j++){
        if(arr[i][j]>max)
            max=arr[i][j];
    }
    temp1[i]=max;
}
for(int j=0;j<4;j++){
    int min=arr[0][j];
    for(int i=0;i<4;i++){
        if(arr[i][j]<min)
            min=arr[i][j];
    }
    temp2[j]=min;
}
for(int j=0;j<4;j++){
    for(int i=0;i<4;i++){
        if(temp1[i]==temp2[j]){
            flag=true;
            row=i+1;
            column=j+1;
            value=temp1[i];
        }
    }
}
if(flag){
    System.out.println("第"+row+"行的第"+column+"列的"+value+"是鞍点");
}else{
    System.out.println("没有鞍点");
}
}
```

【运行结果】

随机生成一个 4×4 的二维数组,获得该二维数组的鞍点,如图 3-9 所示。



```
<terminated> SaddlePointDemo [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (2023年2月8日 上午8:16:47)
63 81 21 87
93 83 51 70
48 22 39 28
83 3 32 51
第3行的第1列的48是鞍点
```

图 3-9 矩阵的鞍点

自测题

1. 输入一个小写字母,以该字母为第一个字母,逆序输出所有的小写字母。例如:输入 e 后,输出 edcbazyxwvutsrqponmlkghgf。如果输入的是其他字符,则显示输入错误。
2. 使用“插入排序”对数组进行排序。
3. 数组消重:已定义一个数组 a,新生成一个数组 b,b 由 a 中所有不重复的元素组成。
4. 随机生成一个二维数组,求二维数组元素中的最大值及位置。