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　输出大于平均值的数

【内容】

从键盘输入 n 个数，输出这些数中大于其平均值的所有数。

【思路】

(1) 使用 Scanner 类输入一个整数 n，表示数组的长度。

(2) 使用 Scanner 类输入多个整数，作为数组中的每个元素。

(3) 计算数组的平均值，将数组中的每个元素与平均值进行比较，如果大于平均值则输出。

【代码】

```
package three;
import java.util.Scanner;
public class AboveAverageDemo {
    public static void main(String[] args) {
        int n;
        double sum=0,average;
        Scanner sc=new Scanner(System.in);
        System.out.print("请输入个数 n=");
        n=sc.nextInt();
        int a[]=new int[n];
```

```
        if(n>0) {
            for(int i=0;i<n;i++) {
                System.out.print("请输入第"+(i+1)+"个数:");
                a[i]=sc.nextInt();
                sum=sum+a[i];
            }
            average=sum/n;
            System.out.println("所有数据的平均数是:"+average);
            System.out.print("大于平均数"+average+"的数有:");
            for(int i=0;i<n;i++) {
                if(a[i]>average) {
                    System.out.print(a[i]+" ");
                }
            }
        }else {
            System.out.print("没有数据");
        }
    }
}
```

【运行结果】

如图 3-1 所示，输入一个整数 5 作为数组的长度，然后输入 5 个数据作为数组的每个元素，计算平均值后和每个元素进行比较，输出大于平均值的数据。

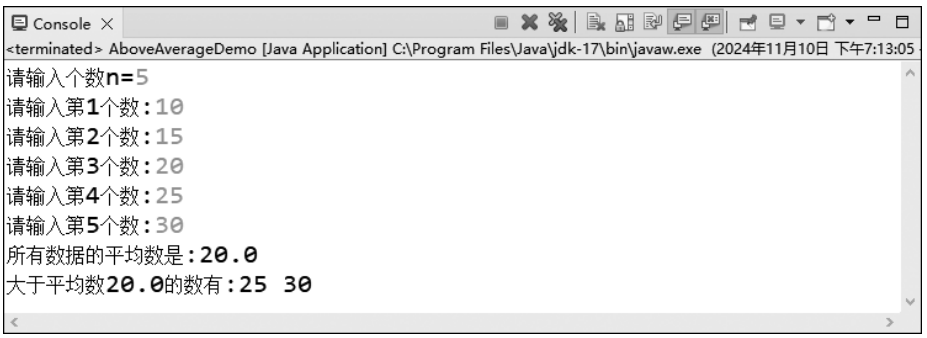

图 3-1　输出大于平均值的数

实验 3-2　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，需要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.print("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】

如图 3-2 所示，输入一个字母 F 后按 Enter 键，以 F 开始正序输出所有的大写字母。

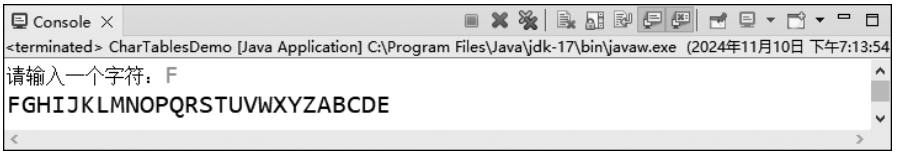

图 3-2 正序输出所有大写字母

实验 3-3　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 3-3 所示，随机生成一组双色球号码。

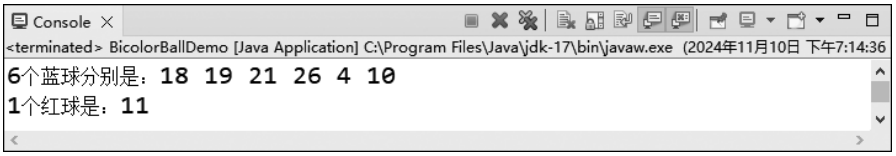

图 3-3 双色球号码

实验 3-4 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，对数组中的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.print("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 3-4 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。

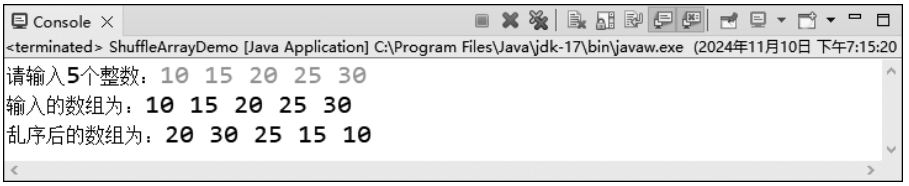

图 3-4　数组乱序输出

实验 3-5　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.print("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】

如图 3-5 所示，输入一个数据 28 后按 Enter 键，查找该数据在数组中的位置以及出现的次数。

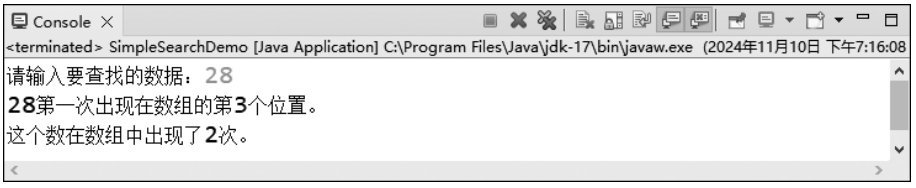

图 3-5 数组中查找数据

实验 3-6 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.print("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.print("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.print("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】

如图 3-6 所示,输入一个数据 40,将 40 插入数组中并确保数组仍是有序的。

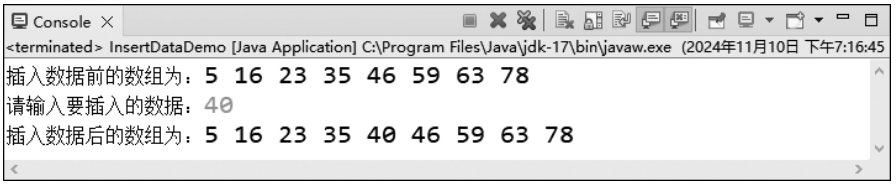

图 3-6　插入数据

实验 3-7　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 个数中找到较小的元素,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.print("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.print("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.print("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】

如图 3-7 所示，输入 8 个整数，存放到数组中，对数组进行选择排序，运行显示每趟选择排序的结果。

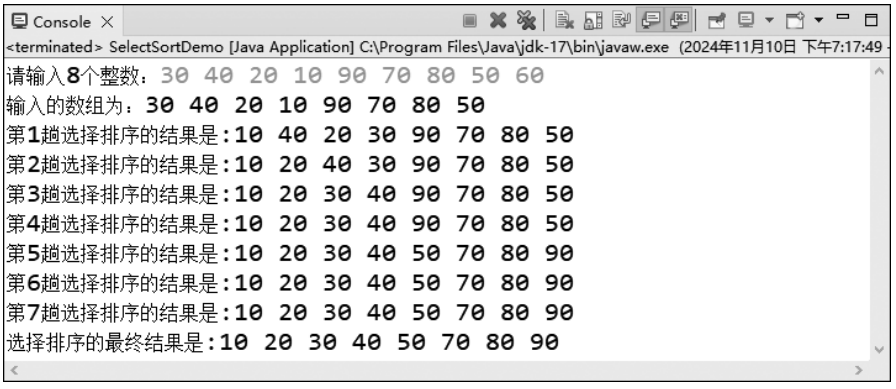

图 3-7　选择排序

实验 3-8　插入排序

【内容】

给定一个数组，使用插入排序法对数组进行从小到大的排序并输出。

【思路】

插入排序的过程为：将数组分为已排序的区间和未排序的区间，然后每次从未排序区间取出一个元素，将其插入已排序区间的合适位置中，使得数据插入后仍然保持已排序的区间有序，使用循环重复这个过程，直到未排序空间为空，整个序列有序为止。

【代码】

```
package three;
import java.util.Scanner;
public class InsertSortTest {
    public static void main(String[] args) {
        int a[]={15,5,25,18,35,46,38};
        System.out.print("数组顺序为:");
        for(int i=0;i<a.length;i++){
            System.out.print(a[i]+" ");
        }
        for(int i=1;i<a.length;i++) {
            int temp=a[i];
            int j=i-1;
            while(j>=0&&a[j]>temp) {
                a[j+1]=a[j];
                j--;
            }
```

```
            a[j+1]=temp;
        }
        System.out.print("\n 插入排序后的数组顺序为:");
        for(int i=0;i<a.length;i++){
            System.out.print(a[i]+" ");
        }
    }
}
```

【运行结果】

如图 3-8 所示，使用插入排序对数组进行升序排序。

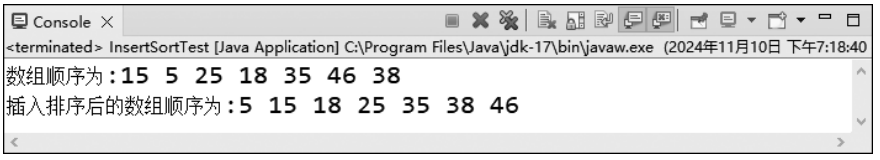

图 3-8 插入排序

实验 3-9 快速排序

【内容】

给定一个数组，使用快速排序法对数组进行从小到大的排序并输出。

【思路】

快速排序的过程为：首先设定一个分界值，通过该分界值将数组分成左右两部分。将大于或等于分界值的数据集中到数组右边，小于分界值的数据集中到数组的左边。此时，左边部分中各元素都小于分界值，而右边部分中各元素都大于或等于分界值。然后，左边和右边的数据可以独立排序。对于左侧的数组数据，又可以取一个分界值，将该部分数据分成左右两部分，同样在左边放置较小值，右边放置较大值。右侧的数组数据也可以做类似处理。使用递归重复上述过程，通过递归将左侧部分排好序后，再递归排好右侧部分的顺序。当左、右两个部分各数据排序完成后，整个数组也是有序的了。

【代码】

```
package three;
public class QuickSortDemo{
    public static void quickSort(int[] arr, int low, int high){
        if(low<high){
            int pivotIndex=partition(arr,low,high);
            quickSort(arr,low,pivotIndex-1);
            quickSort(arr,pivotIndex+1,high);
        }
    }
    private static int partition(int[] arr,int low,int high){
```

```
        int pivot=arr[high];
        int i=(low-1);
        for (int j=low;j<high;j++){
            if (arr[j]<pivot){
                i++;
                int temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
        }
        int temp=arr[i+1];
        arr[i+1]=arr[high];
        arr[high]=temp;
        return i+1;
    }
    public static void main(String[] args){
        int[] arr={10,5,15,0,20,25};
        quickSort(arr,0,arr.length-1);
        for (int val:arr) {
            System.out.print(val+" ");
        }
    }
}
```

【运行结果】

如图 3-9 所示，使用快速排序对数组进行升序排序。

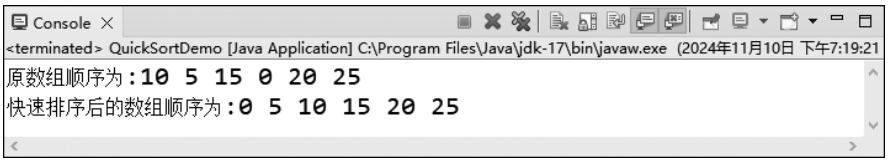

图 3-9　快速排序

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 3-10　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 3-10 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 3-10　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×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～n－1,列下标 j=1～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了。

(4) 输出时只需输出二维数组的左下三角部分内容即可。

【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.print("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】

如图 3-11 所示,输入一个正整数 7,显示 7 行杨辉三角形。

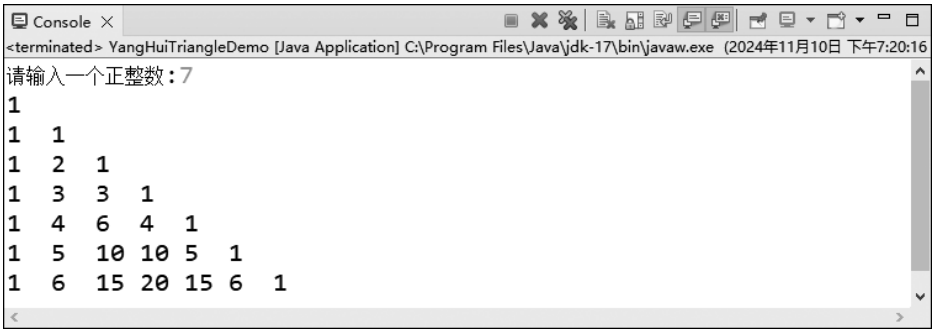

图 3-11　杨辉三角形

实验 3-11　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】

如图 3-12 所示，随机生成一个 4×4 的二维数组，获得该二维数组的鞍点。

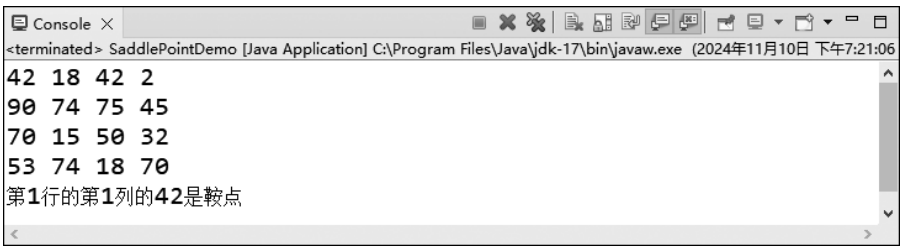

图 3-12　矩阵的鞍点

实验 3-12　对角线元素之和

【内容】

求一个 3 阶方阵的对角线上各元素之和。

【思路】

使用 Scanner 类输入一个 3 阶方阵的 9 个数据，存放在一个二维数组里，计算二维数组行号和列号相等位置的元素之和。

【代码】

```
package three;
import java.util.Scanner;
public class DiagonalsSum {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int[][] nums=new int[3][3];
        int i,j,sum=0;
        System.out.println("请输入一个 3 阶的方阵数据:");
        for(i=0;i<3;i++) {
            for(j=0;j<3;j++) {
                nums[i][j]=sc.nextInt();
            }
        }
        for(i=0;i<3;i++) {
            sum=sum+nums[i][i];
        }
        System.out.print("对角线的和是:"+sum);
    }
}
```

【运行结果】

如图 3-13 所示，输入一个 3 阶的方阵，输出对角线元素之和。

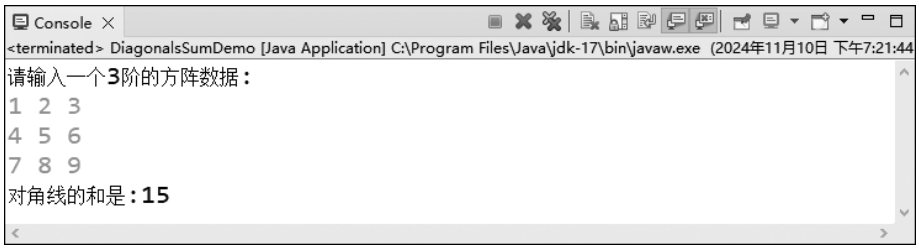

图 3-13　对角线元素之和

实验 3-13　二维数组的最值

【内容】

求一个给定二维数组的最大值和最小值以及所在的位置。

【思路】

(1) 给定一个二维数组,定义两个变量 max,min 分别存储最大值和最小值;定义两个变量 maxrow,maxcolumn 存储最大值所在的位置;定义两个变量 minrow,mincolumn 存储最小值所在的位置。

(2) 使用循环求得二维数组的最大值、最小值以及最大值和最小值所在的位置。

【代码】

```
package three;
public class MaxMinSearchDemo {
    public static void main(String[] args) {
        int a[][]= {{5,3,1},{2,4,6},{9,8,7}};
        int max=a[0][0];
        int min=a[0][0];
        int maxcolumn=0;
        int maxrow=0;
        int mincolumn=0;
        int minrow=0;
        for(int i=0;i<a.length;i++) {
            for(int j=0;j<a[i].length;j++) {
                if(a[i][j]>max) {
                    max=a[i][j];
                    maxrow=i;
                    maxcolumn=j;
                }
                if(a[i][j]<min) {
                    min=a[i][j];
                    minrow=i;
                    mincolumn=j;
                }
            }
        }
         System.out.println("最大值"+max+"在第"+(maxrow+1)+"行,在第"+
(maxcolumn+1)+"列");
         System.out.println("最小值"+min+"在第"+(minrow+1)+"行,在第"+
(mincolumn+1)+"列");
    }
}
```

【运行结果】

如图 3-14 所示,显示这个二维数组的最大值、最小值以及所在的位置。

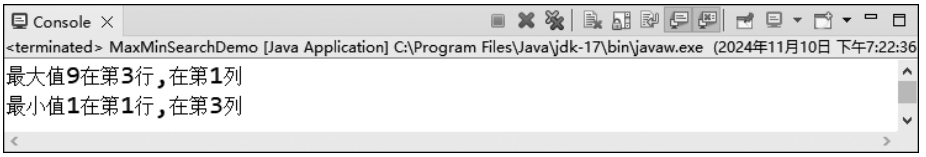

图 3-14　二维数组的最大值和最小值

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 edcbazyxwvutsrqponmlkgihgf。如果输入的是其他字符，显示输入错误。

2. 输入一个长度为 10 的整型数组，将其中的最小值与第一个数交换位置，将其中的最大值与最后一个数交换位置，并输出结果。

3. 数组消重：已定义一个数组 a，新生成一个数组 b，b 是由 a 中所有不重复的元素组成的。

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。

5. 将一维数组转换为二维数组。

6. 归并排序。