1. 定义一个数组，将数组正序输出和倒序输出

代码：

public void test1(int[] a){

//定义一个数组，将数组正序输出和倒序输出

for(int i = 0; i<a.length; i++){

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

}

System.out.println();//换行 倒序输出

for(int i = a.length-1; i>=0; i--){

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

}

}

1. 求一组数的最大值和最小值以及平均值

代码：

public void test2(int[] a){

//求一组数的最大值和最小值以及平均值

//最大值

int max = a[0];

for(int i = 1; i<a.length;i++){

if(a[i] > max){

max = a[i];

}

}

System.out.println("最大值为 "+max);

//最小值

int min = a[0];

for(int i = 1; i<a.length;i++){

if(a[i]<min){

min = a[i];

}

}

System.out.println("最小值为 "+min);

//平均值

double sum = 0;

for(int i = 0; i<a.length;i++){

sum+=a[i];

}

double p = sum/a.length;

System.out.println("平均值为 "+p);

}

1. 产生4位随机验证码（借助API完成）

从该字符串中产生：

String str="abcdefghigklmnopqrstuvwxyzABCDEFGHIGKLMNOPQRSTUVWXYZ0123456789";

生成指定范围的随机数：

Random r = new Random();

r.nextInt(10);

字符串截取的方法：

str.subString(1,2);

代码：

public void test3(){

/\*产生4位随机验证码（借助API完成）

从该字符串中产生：

String str="abcdefghigklmnopqrstuvwxyzABCDEFGHIGKLMNOPQRSTUVWXYZ0123456789";

生成指定范围的随机数：

Random r = new Random();

r.nextInt(10); 随机生成一个0到参数（不包括参数）的一个随机数

字符串截取的方法：

str.subString(1,2); 截取str的字符串从索引1开始 到索引2（不包含）\*/

Random r = new Random();

String str="abcdefghigklmnopqrstuvwxyzABCDEFGHIGKLMNOPQRSTUVWXYZ0123456789";

for(int i =0 ;i< 4;i++){

int ran = r.nextInt(62);

String s = str.substring(ran,ran+1);

System.out.print(s);

}

}

1. 写出数组拷贝的几种方法(最少三种)。

代码：

public void test4(int[] a){

//写出数组拷贝的几种方法(最少三种)

int[] b = new int[10]; //1

for(int i = 0;i<a.length;i++){

b[i] = a[i];

}

System.out.println(Arrays.toString(b));

int[] c = new int[10]; //2

System.arraycopy(a,0,c,0,a.length);

System.out.println(Arrays.toString(c));

int[] d =Arrays.copyOf(a,10);

System.out.println(Arrays.toString(d));

}

1. 产生100个[1,6]的整数，并统计每个整数出现的频率

代码：

public void test5(){

//产生100个[1,6]的整数，并统计每个整数出现的频率

int[] arr = new int[6];

System.out.println("生成的数字为 ：");

for(int i = 0;i < 100; i++){

int a = (int)(Math.random()\*6+1);

System.out.print(a+" ");

arr[a-1]++;

}

System.out.println();

for(int i = 0; i < arr.length; i++){

if(arr[i] != 0){

System.out.println("数字"+(i+1)+"的出现次数为"+arr[i]+"次");

}

}

}

1. 定义一个数组 {9,3,1,7,8},将该数组从小到大进行排序

a）选择排序:

在要排序的一组数中，选出最小的一个数与

第一个位置的数交换；

然后在剩下的数当中再找最小的与第二个位置的数交换，

如此循环到倒数第二个数和最后一个数比较为止。

Selection.java

{9,3,1,7,8}

0 第一轮:1,3,9,7,8

1 第二轮:1,3,9,7,8

2 第三轮:1,3,7,9,8

3 第四轮:1,3,7,8,9 array.length-1轮

public static arr

代码：

public class Test{

public static void main(String[] args){

int[] array={10,17,45,8,75,24,44,11,7};

System.out.println("排序前的数组为："+Arrays.toString(array));

for(int i=0;i<array.length-1;i++){

int index=i;//记录假最小值的下标

for(int j=i+1;j<array.length;j++){

if(array[index]>array[j]){

index=j;

}

}

//交换array[i]与array[index]元素位置

int temp=array[i];

array[i]=array[index];

array[index]=temp;

System.out.println("第"+(i+1)+"趟后数组为："+Arrays.toString(array));

}

}

}

b）冒泡排序:

在要排序的一组数中，对当前还未排好序的

范围内的全部数，自上而下对相邻的两个数

依次进行比较和调整，

让较大的数往下沉，较小的往上冒。

即：每当两相邻的数比较后发现它们的排序

与排序要求相反时，就将它们互换。

Bubble.java

代码：

public void sort(int[] a) {

//-1减少比较趟数

for (int i = 0; i < a.length -1; i++) {//趟数

// -1 ： 防止 下标越界 -i 减少比较趟数

for (int j = 0; j < a.length-1-i; j++) {//两两比较

if(a[j] > a[j+1]) {//左边的数大于右边的数

//交换

int temp = a[j];//中间变量 保存

a[j] = a[j+1];//将右边的数移动到左边

a[j+1] = temp;//将左边的数移动到右边

}

}

System.out.println("第"+(i+1)+"趟："+Arrays.toString(a));

}

//扩展：插入排序 选择排序

}

c）插入排序:

排序过程的某一中间时刻，R被划分成两个子区间R[1．．i-1]（已排好序的有序区）和

R[i．．n]（当前未排序的部分，可称无序区）。

插入排序的基本操作是将当前无序区的第1个记录R[i]插人到有序区R[1．．i-1]中适当的

位置上，使R[1．．i]变为新的有序区。因为这种方法每次使有序区增加1个记录，通常称增量法。

插入排序与打扑克时整理手上的牌非常类似。摸来的第1张牌无须整理，此后每次从桌上的

牌(无序区)中摸最上面的1张并插入左手的牌(有序区)中正确的位置上。为了找到这个正确

的位置，须自左向右(或自右向左)将摸来的牌与左手中已有的牌逐一比较。

Insertion.java

代码：

import java.util.Arrays;

public class Test{

public static void main(String[] args){

int[] array = {34,8,64,51,32,21};

System.out.println("插入排序数组为："+Arrays.toString(array));

for(int i=1;i<array.length;i++) {

int temp = array[i];//插入元素的值

int index=i;//插入元素的下标值

//array[i]与 它前面的元素(a[0]-a[i-1])进行比较。

for(int j=i-1;j>=0;j--) {

//当a[i]比前一个元素小，将前一个元素向后移动。

if (temp < array[j]) {

array[j + 1] = array[j];

index--;//插入元素下标值减小

}

}

//最后将a[0]-a[i-1]之间移动后空缺的位置array[index]插入a[i]

array[index] = temp;

System.out.println("第"+(i+1)+"趟后数组为："+Arrays.toString(array));

}

}

}