**数组作业答案**

1. **填空题**
   1. 连续
   2. length
   3. 声明数组
   4. null
   5. array[0].length
2. **选择题**
   1. AD
   2. D
   3. B
   4. CD
   5. A
   6. B
   7. AC
   8. B
   9. AD
3. **判断题**
4. √
5. ×
6. √
7. √
8. ×
9. √
10. √
11. ×
12. **简答题**

答案略

1. **编码题**
2. 定义一个长度为10 的一维字符串数组，在每一个元素存放一个单词；然后运行时从命令行输入一个单词，程序判断数组是否包含有这个单词，包含这个单词就打印出“Yes”，不包含就打印出“No”。

**public** **class** Test {

// 思路：遍历数组时当有值与输入字符串相同时使用一个boolean记录状态

**public** **static** **void** main(String[] args) {

String[] strArr = **new** String[10];

strArr[0] = "java"; strArr[1] = "html";

strArr[2] = "sql"; strArr[3] = "android";

strArr[4] = "javascript"; strArr[5] = "ruby";

strArr[6] = "c";

strArr[7] = "c++";

strArr[8] = ".net";

strArr[9] = "ios";

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("请输入需要查找的计算机编程语言:");

String inputStr = scanner.next();

**boolean** flag = **false**;

**for** (String string : strArr) {

**if** (string.equals(inputStr)) {

flag = **true**;

}

}

**if** (flag) {

System.***out***.println("YES");

} **else** {

System.***out***.println("NO");

}

}

}

1. 利用Java的Math类的random()方法，编写函数得到0到n之间的随机数，n是参数。  
   提示： int num=(int)(n\*Math.random());  
   并找出产生50个这样的随机数中最大的、最小的数，并统计其中大于等于60的有多少个。

**public** **class** Test {

**public** **static** **void** main(String[] args) {

//创建一个长度为50的，每个元素在0-100之间的数组

**int**[] arr = *getArray*(100);

//遍历数组

*printArr*(arr);

//获取最大值和最小值

**int** min = *getMin*(arr);

**int** max = *getMax*(arr);

System.***out***.println("数组中最大值：" + max + "\t数组中最小值：" + min);

//统计其中大于等于60的元素个数

**int** count = *count*(arr,60);

System.***out***.println("数组中大于或等于60的数有:" + count + " 个");

}

//遍历数组

**private** **static** **void** printArr(**int**[] arr) {

System.***out***.println("输出数组元素");

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

**if**(i%10 ==0){

System.***out***.println();

}

System.***out***.print(arr[i]+"\t");

}

}

//获取数组最大值

**private** **static** **int** getMax(**int**[] arr) {

**int** max = arr[0];

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

**if**(max<arr[i]){

max = arr[i];

}

}

**return** max;

}

//获取数组最小值

**private** **static** **int** getMin(**int**[] arr) {

**int** min = arr[0];

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

**if**(min>arr[i]){

min = arr[i];

}

}

**return** min;

}

// 创建数组 并且初始化50个数据

**public** **static** **int**[] getArray(**int** n) {

**int** arr[] = **new** **int**[50];

**for** (**int** i = 0; i < 50; i++) {

arr[i] = (**int**) (n \* Math.*random*());

}

**return** arr;

}

// 统计数组中大于等于60的数量的方法 思路：使用一个int值计数

**public** **static** **int** count(**int**[] arr,**int** elem) {

**int** count = 0;

**for** (**int** i : arr) {

**if** (i >= elem) {

count++;

}

}

**return** count;

}

}

1. 数组逆序操作：定义长度为10的数组，将数组元素对调，并输出对调前后的结果。

**public** **class** Test{

**public** **static** **void** main(String[] args) {

//定义一个数组，并进行静态初始化。

**int**[] arr = {12,98,50,34,76,87,90,34,67,80};

//逆序前

System.***out***.println("逆序前：");

*printArr*(arr);

//逆序后

System.***out***.println("逆序后：");

*reverseArr*(arr);

*printArr*(arr);

}

**public** **static** **void** reverseArr(**int**[] arr) {

**for**(**int** x=0; x<arr.length/2; x++) {

**int** temp = arr[x];

arr[x] = arr[arr.length-1-x];

arr[arr.length-1-x] = temp;

}

}

//遍历数组

**public** **static** **void** printArr(**int**[] arr) {

System.***out***.print("[");

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

**if**(x == arr.length-1) { //这是最后一个元素

System.***out***.println(arr[x]+"]");

}**else** {

System.***out***.print(arr[x]+", ");

}

}

}

}

1. 现在有如下的一个数组：   int oldArr[]={1,3,4,5,0,0,6,6,0,5,4,7,6,7,0,5}   要求将以上数组中值为0的项去掉，将不为0的值存入一个新的数组，生成的新数组为： int newArr [] ={1,3,4,5,6,6,5,4,7,6,7,5}

思路： 确定出不为0的个数，这样可以开辟新数组；从旧的数组之中，取出内容，并将其赋给新开辟的数组。

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int** oldArr[] = { 1, 3, 4, 5, 0, 0, 6, 6, 0, 5, 4, 7, 6, 7, 0, 5 };

**int** newArr[] = *selectArr*(oldArr);

**for** (**int** i : newArr) {

System.***out***.print(i+" ");

}

}

// 去掉数组中值为0的元素的方法

**public** **static** **int**[] selectArr(**int**[] arr) {

// 1.计算数组中元素不为0的个数

**int** count = 0;

**for** (**int** i : arr) {

**if** (i != 0) {

count++;

}

}

// 2.创建一个新数组,长度为count

**int** newArr[] = **new** **int**[count];

// 3.复制不为0的元素到新数组中

**int** size = 0;

**for** (**int** i : arr) {

**if** (i != 0) {

newArr[size++] = i;

}

}

**return** newArr;

}

}

1. 使用二分法查找有序数组中元素。找到返回索引，不存在输出-1。

**public** **class** BinarySearch {

**public** **static** **void** main(String[] args) {

**int**[] array = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

System.*out*.println(*searchLoop*(array, 9));

}

/\*\*

\* 循环二分查找，返回第一次出现该值的位置

\* **@param** array 已排序的数组

\* **@param** findValue 需要找的值

\* **@return** 值在数组中的位置，从0开始。找不到返回-1

\*/

**public** **static** **int** searchLoop(**int**[] array, **int** findValue) {

// 如果数组为空，直接返回-1，即查找失败

**if** (array == **null**) {

**return** -1;

}

// 起始位置

**int** start = 0;

// 结束位置

**int** end = array.length - 1;

**while** (start <= end) {

// 中间位置

**int** middle = (start + end) / 2;

// 中值

**int** middleValue = array[middle];

**if** (findValue == middleValue) {

// 等于中值直接返回

**return** middle;

} **else** **if** (findValue < middleValue) {

// 小于中值时在中值前面找

end = middle - 1;

} **else** {

// 大于中值在中值后面找

start = middle + 1;

}

}

// 返回-1，即查找失败

**return** -1;

}

}

1. 用二重循环求出二维数组b所有元素的和：  
    int[][] b={{11},{21,22},{31,32,33}}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int** sum=0;

**int**[][] b={{11},{21,22},{31,32,33}};

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

**for**(**int** j=0;j<b[i].length;j++){

sum += b[i][j];

}

}

System.***out***.println(sum);

}

}

1. **可选题**
2. 生成一百个随机数，放入数组，然后排序输出。

**public** **class** Test {

**public** **static** **void** main(String[] args) {

//1.生成100个随机数的数组

**int** []arr = *createArr*();

//2.从小到大排序

arr = *sortArr*(arr);

//3.打印

*printArr*(arr);

}

//生成100个随机数的方法

**public** **static** **int**[] createArr(){

**int** []arr = **new** **int**[100];

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

arr[i]= (**int**) (100\*Math.*random*());

}

**return** arr;

}

//对数组进行排序的方法

**public** **static** **int**[] sortArr(**int** arr[]){

**int** temp;

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

**for** (**int** j = 0; j <= i; j++) {

**if** (arr[i] < arr[j]) {

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

**return** arr;

}

//数组遍历输出打印的方法

**public** **static** **void** printArr(**int** []arr){

**for**(**int** i:arr){

System.***out***.println(i);

}

}

}

1. 题目：输入某年某月某日，判断这一天是这一年的第几天？

分析：以3月5日为例，应该先把前两个月的加起来，然后再加上5天即本年的第几天，特殊情况，闰年且输入月份大于3时需考虑多加一天。可以定义数组存储1-12月各个月的天数。

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Scanner scanner1 = **new** Scanner(System.***in***);

System.***out***.print("输出今年是第几年");

**int** year = scanner1.nextInt();

System.***out***.print("输出今年是第几月");

**int** month = scanner1.nextInt();

System.***out***.print("输出今年是第几日");

**int** day = scanner1.nextInt();

**int** daynum = 0; // 天数

//一年365天

**int**[] month1 = **new** **int**[] { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,30, 31 };

//闰年366天

**if** (year % 400 == 0 && year % 100 != 0 || year % 100 == 0) {

month1 = **new** **int**[] { 31, 29, 31, 30, 31, 30, 31, 31, 30, 31,30, 31 }; // 闰年里的每个月的天数

}

//判断这一天是第几天

**for** (**int** index = 1; index <= month; index++) {

**if** (index == 1) {

daynum = day;

} **else** {

daynum += month1[index - 2];

}

}

System.***out***.println("这一天是这一年的第" + daynum + "天！");

}

}

1. 使用二分法查找有序数组中元素。找到返回索引，不存在输出-1。使用递归实现

**public** **class** BinarySearch {

**public** **static** **void** main(String[] args) {

**int**[] array = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

System.*out*.println(*searchRecursive*(array, 0, array.length - 1, 20));

}

/\*\*

\* 执行递归二分查找，返回第一次出现该值的位置 \*

\* **@param** array 已排序的数组

\* **@param** start 开始位置

\* **@param** end 结束位置

\* **@param** findValue 需要找的值

\* **@return** 值在数组中的位置，从0开始。找不到返回-1

\*/

**public** **static** **int** searchRecursive(**int**[] array, **int** start, **int** end,

**int** findValue) {

// 如果数组为空，直接返回-1，即查找失败

**if** (array == **null**) {

**return** -1;

}

**if** (start <= end) {

// 中间位置

**int** middle = (start + end) / 1;

// 中值

**int** middleValue = array[middle];

**if** (findValue == middleValue) {

// 等于中值直接返回

**return** middle;

} **else** **if** (findValue < middleValue) {

// 小于中值时在中值前面找

**return** *searchRecursive*(array, start, middle - 1, findValue);

} **else** {

// 大于中值在中值后面找

**return** *searchRecursive*(array, middle + 1, end, findValue);

}

} **else** {

// 返回-1，即查找失败

**return** -1;

}

}

}

1. 现在给出两个数组：

 数组A：“1，7，9，11，13，15，17，19：；

 数组b：“2，4，6，8，10”

 两个数组合并为数组c，按升序排列。

要求：使用Arrays类的方法快速实现。

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**int** data1[] = **new** **int**[] { 1, 7, 9, 11, 13, 17, 19 };

**int** data2[] = **new** **int**[] { 2, 4, 6, 8, 10 };

**int** newArr[] = *concat*(data1, data2);

*print*(newArr);

Arrays.*sort*(newArr);

*print*(newArr);

}

**public** **static** **int**[] concat(**int** src1[], **int** src2[]) {

**int** len = src1.length + src2.length; // 新数组的大小

**int** arr[] = **new** **int**[len]; // 新数组

// int count = 0;

// for (int i : src1) {

// arr[count++] = i;

// }

// for (int i : src2) {

// arr[count++] = i;

// }

// 还可以通过系统的方法拷贝数组

System.*arraycopy*(src1, 0, arr, 0, src1.length); // 拷贝第一个数组

System.*arraycopy*(src2,0,arr,src1.length, src2.length); //拷贝第二个数组

**return** arr;

}

**public** **static** **void** print(**int**[] temp) {

**for** (**int** x = 0; x < temp.length; x++) {

System.***out***.print(temp[x] + "、");

}

System.***out***.println();

}

}