**数组作业**

1. **填空题**
2. 数组会在内存中开辟一块\_\_**连续**\_\_\_\_\_\_的空间，每个空间相当于之前的一个变量，称为数组的元素。数组的长度一经确定，就无法再改变。
3. 要获取一个数组的长度，可以通过\_\_\_**length**\_\_\_\_\_\_\_\_\_属性来获取，但获取的只是为数组分配的空间的数量，而不是数组中实际已经存放的元素的个数。
4. \_\_\_**声明数组**\_\_\_\_\_\_\_\_仅仅是给出了数组名字和元素的数据类型，要想真正的使用数组还必须使用new关键字为它分配内存空间。
5. 创建数组后，系统会给每一个数组元素一个默认的值，如String类型元素的默认值是\_\_\_\_**null**\_\_\_\_\_\_\_\_。
6. 在Java中有二维数组int [ ] [ ] array={{1,2,3},{4,5}} ，可以使用\_**array[0].length**得到二维数组中第二维中第一个数组的长度。
7. **选择题**

|  |  |  |
| --- | --- | --- |
| **1.** | **在Java中，以下程序段能正确为数组赋值的是（ AD ）。（选择二项）** | |
|  |  |  |
|  | **A** | int a[]={1,2,3,4}; |
|  | **B.** | int b[4]={1,2,3,4}; |
|  | **C.** | int c[];c={1,2,3,4}; |
|  | **D.** | int d[];d=new int[]{1,2,3,4}; |

|  |  |  |
| --- | --- | --- |
| **2.** | **数组元素的索引可以是（ D ）。（选择一项）** | |
|  |  |  |
|  | **A** | 整型常量 |
|  | **B.** | 整型变量 |
|  | **C.** | 整型表达式 |
|  | **D.** | 以上都可以 |

|  |  |  |
| --- | --- | --- |
| **3.** | **已知表达式int [] m={0,1,2,3,4,5,6};下面（ B ）表达式的值与数组最大下标数相等。（选择一项）** | |
|  |  |  |
|  | **A** | m.length() |
|  | **B.** | m.length-1 |
|  | **C.** | m.length()+1 |
|  | **D.** | m.length+1 |

|  |  |  |
| --- | --- | --- |
| **4.** | **在Java中，以下定义数组的语句正确的是（ CD ）。（选择二项）** | |
|  |  |  |
|  | **A** | int t[10]=new int[ ]; |
|  | **B.** | char [ ]a=”hello”; |
|  | **C.** | String [ ] s=new String [10]; |
|  | **D.** | double[ ] d [ ]=new double [4][ ]; |

|  |  |  |
| --- | --- | --- |
| **5.** | **在Java中,下面代码的输出结果为（ A ）。（选择一项）** | |
|  | **public** **static** **void** main(String[] args) {  **int**[] arrA = { 12, 22, 8, 49, 3 };  **int** k = 0; **int** len = arrA.length;  **for** (**int** i = 0; i < len; i++) {  **for** (**int** j = i + 1; j < len; j++) {  **if** (arrA[i] > arrA[j]) {  k = arrA[i];  arrA[i] = arrA[j];  arrA[j] = k;  }  }  }  **for** (**int** i = 0; i < arrA.length; i++) {  System.*out*.print(arrA[i]);  **if** (i < arrA.length - 1) {  System.*out*.print("，");  }  }  } | |
|  |  |  |
|  | **A** | 3，8，12，22，49 |
|  | **B.** | 12，22，8，49，3 |
|  | **C.** | 49，22，12，8，3 |
|  | **D.** | 编译错误 |

|  |  |  |
| --- | --- | --- |
| 6. | **分析下面的Java源程序，编译后的运行结果是（ B ）。（选择一项）** | |
|  | **import** java.util.\*;  **public** **class** Test {  **public** **static** **void** main(String[ ] args) {  **int** [ ] numbers=**new** **int**[ ]{1,2,3};  System.*out*.println(**Arrays.*binarySearch*(numbers, 2)**);  }  }  **Arrays.*binarySearch*(numbers, 2):表示取数组种为2的下标是多少** | |
|  |  | 输出 |
|  | **A** | 输出：0 |
|  | **B.** | 输出：1 |
|  | **C.** | 输出：2 |
|  | **D.** | 输出：3 |

|  |  |  |
| --- | --- | --- |
| **7.** | **以下选项中关于Java中方法的可变参数的说法正确的是（ AC ）。（选择二项）** | |
|  |  |  |
|  | **A** | 可变参数是JDK1.5增加的内容，用来表示方法的形参 |
|  | **B.** | 一个方法可以没有可变参数，可以有1个或者多个可变参数 |
|  | **C.** | 可变参数可以被当作数组参数来处理 |
|  | **D.** | 可变参数对应的实参可以1个，2个，多个，但不能0个，还可以是一个数组 |

|  |  |  |
| --- | --- | --- |
| **8.** | **分析下面的Java程序，编译运行后的输出结果是（ B ）。（选择一项）** | |
|  | **public** **class** Example {  String str = **new** String("good");  **char**[] ch = { 'a', 'b', 'c' };  **public** **static** **void** main(String args[]) {  Example ex = **new** Example( );  ex.change(ex.str, ex.ch);  System.*out*.print(ex.str + "and");  System.*out*.print(ex.ch);  }  **public** **void** change(String str1, **char** ch1[]) {  str1 = "test ok";  ch1[0] = 'g';  }  } | |
|  |  |  |
|  | **A** | goodandabc |
|  | **B.** | goodandgbc |
|  | **C.** | test okandabc |
|  | **D.** | test okandgbc |

|  |  |  |
| --- | --- | --- |
| **9.** | **以下选项中能够正确创建一个数组的是（ AD ）。（选择二项）** | |
|  |  |  |
|  | **A.** | float []f[] = new float[6][6]; |
|  | **B.** | float f[][] = new float[][]; |
|  | **C.** | float [6][]f = new float[6][6]; |
|  | **D.** | float [][]f = new float[6][]; |

1. **判断题**
2. 数组可以声明为任何数据类型，包括任何基本数据类型和引用数据类型。（ **〇** ）
3. 数组的长度是确定的，数组一旦被创建，它的大小就是不可以改变的。但是其元素类型可以是不同类型，允许出现混合类型。（ **×** ）
4. 声明数组并分配空间后，数组的每个元素将会赋予初始值。（ **〇** ）
5. 创建数组后，系统会给每个数组元素一个默认值，如double型元素的默认值是0.0。（ **〇** ）
6. 数组的主要优点是按照索引查找某个元素效率高，同时按照元素值查询某个元素效率也很高，但是添加和删除元素需要大量移动元素，效率低下。（ **×** ）
7. 数组的某个元素被传递给一个方法并被该方法修改，当被调用方法执行完毕时，这个元素中含有修改过的数值。（**〇** ）
8. Java允许创建不规则数组，即Java多维数组中各行的列数可以不同。（ **〇** ）
9. 对于数组int[][] t={{1,2,3},{4,5,6}}来说，t.length等于3，t[0].length等于2（ **×** ）

这里的t[0].length就是指第一个花括号里元素个数，t[1].length指第二个花括号里元素的个数

1. **简答题**
2. 数组的特点。
3. 数组的优缺点
4. 冒泡排序的算法。
5. **编码题**
6. 数组查找操作：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”。

方法1：

String[] str = new String[]{"a","鸭梨","橘子","风扇","床","电视","桌子","手机","电脑","沙发"};

Scanner input = new Scanner(System.in);

System.out.println("请输入一个物品：");

String content = input.nextLine();

for(int i=0;i<10;i++){

if(content.equals(str[i])){

System.out.println("yes");

return;

}

}

System.out.println("no");

方法2：

**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) {

//本质上和for(int i=0;i<length;i++)没太大区别，但for(String string : strArr)中的arr一定是字符串数组，每次循环取一个字符串出来赋值给string，直到strArr数组长度-1结束

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

flag = **true**;

}

}

**if** (flag) {

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

} **else** {

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

}

}

}

1. 获取数组最大值和最小值操作：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。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>=60的有多少个。  
   提示：使用 int num=(int)(n\*Math.random());获取随机数

**public class** Number {  
 **public static void** main(String args[]) {  
 **int**[] array = *getArr*(20,50) ;  
 *printArr*(array);  
 **int** max = *getMax*(array);  
 **int** min = *getMin*(array);  
 **int** count = *getCount*(array,30);  
 System.***out***.println(**"\n数组里最大数为："**+max);  
 System.***out***.println(**"数组里最小数为："**+min);  
 System.***out***.println(**"数组里超过60的数为："**+count+**"个"**);  
 }  
  
 **public 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"**);  
 }  
 }  
  
 **public static int**[] getArr(**int** l,**int** n){  
 **int**[] arr = **new int**[l];  
 **for**(**int** i=0;i<l;i++){  
 arr[i] = (**int**)(Math.*random*()\*n);  
 }**return** arr;  
 }  
  
 **public static int** getMax(**int**[] arr1){  
 **int** max = arr1[0];  
 **for**(**int** i=0;i<arr1.**length**;i++){  
 **if**(arr1[i]>max){  
 max = arr1[i];  
 }  
 }**return** max;  
 }  
  
 **public static int** getMin(**int**[] arr2){  
 **int** min = arr2[0];  
 **for**(**int** i=0;i<arr2.**length**;i++){  
 **if**(arr2[i]<min){  
 min = arr2[i];  
 }  
 }**return** min;  
 }  
  
 **public static int** getCount(**int**[] arr3,**int** element){  
 **int** count = 0;  
 **for**(**int** i=0;i<arr3.**length**;i++){  
 **if**(arr3[i]>element){  
 count++;  
 }  
 }**return** count;  
 }  
}

**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。

思路：把0索引和arr.length-1的元素交换，把1索引和arr.length-2的元素交换…..

只要交换到arr.length/2的时候即可。

**public class** Number {  
 **public static void** main(String args[]) {  
 **int**[] arr = {1,2,3,4,5,6,7,8,9,10};  
 System.***out***.println(**"原数组顺序为："**);  
 *printArr*(arr);  
 **int**[] arr1 = *arrGet*(arr);  
 System.***out***.println(**"\n交换完数组顺序为："**);  
 *printArr*(arr1);  
 }  
  
 **public static int**[] arrGet(**int**[] arr){  
 **for**(**int** i = 0;i<arr.**length**/2;i++){  
 **int** temp = arr[i];  
 arr[i]=arr[arr.**length**-1-i];  
 arr[arr.**length**-1-i]=temp;  
 }  
 **return** arr;  
 }  
  
 **public static void** printArr(**int**[] arr){  
 **for**(**int** i=0;i<arr.**length**;i++){  
 System.***out***.print(arr[i]+**"\t"**);  
 }  
 }  
  
}

**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. 合并数组操作：现有如下一个数组：   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** Number {  
 **public static void** main(String args[]) {  
 **int** oldArr[] = {1,2,5,8,0,4,0,1};  
 **int**[] newArr=*newArr*(oldArr);  
 *arrPrint*(newArr);  
 }  
  
 **public static void** arrPrint(**int**[] arr){  
 System.***out***.print(**"{"**);  
 **for**(**int** i =0;i<arr.**length**;i++){  
 System.***out***.print(arr[i]+**"\t"**);  
 }  
 System.***out***.print(**"}"**);  
 }  
  
  
 **public static int**[] newArr(**int**[] arr){  
 **int** count = 0;  
 **for**(**int** i =0;i<arr.**length**;i++){  
 **if**(arr[i]!=0){  
 count++;  
 }  
 }  
 **int**[] newArr = **new int**[count];  
 **int** j = 0;  
 **for**(**int** i=0;i<arr.**length**;i++){  
 **if**(arr[i]!=0){  
 newArr[j++]=arr[i];  
 }  
 }**return** newArr;  
 }  
}

**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. 二分法查找操作：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。

分析：二分法查找的前提是数组有序。

假如有一组数为3，12，24，36，55，68，75，88要查给定的值24.可设三个变量front，mid，end分别指向数据的上界，中间和下界，mid=（front+end）/2.

1. 开始令front=0（指向3），end=7（指向88），则mid=3（指向36）。因为mid>x，故应在前半段中查找。
2. 令新的end=mid-1=2，而front=0不变，则新的mid=1。此时x>mid，故确定应在后半段中查找。
3. 令新的front=mid+1=2，而end=2不变，则新mid=2，此时a[mid]=x，查找成功。
4. 如要查找的数不是数列中的数，例如x=25，当第三次判断时，x>a[mid]，按以上规律，令front=mid+1，即front=3，出现front>end的情况，表示查找不成功。

**public class** Number {  
 **public static void** main(String[] args) {  
 **int**[] array = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
 System.***out***.println(*searchLoop*(array, 9));  
 }  
 **public static int** searchLoop(**int**[] arr,**int** findNumber){  
 **int** start = 0;  
 **int** end = arr.**length**-1;  
 **while**(start<=end){  
 **int** middle = (start+end)/2;  
 **if**(findNumber<arr[middle]){  
 end = middle-1;  
 }**else if**(findNumber>arr[middle]){  
 start = middle+1;  
 }**else**{  
 **return** middle;  
 }  
 }**return** -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. 二维数组遍历求和操作：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** Number {  
 **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** Number {  
 **public static void** main(String[] args) {  
 **int**[] arr = *arrayCreate*(50);  
 arr = *arrayNew*(arr);  
 *arrayPrint*(arr);  
 }  
  
 **public static int**[] arrayCreate(**int** number){  
 **int**[] array = **new int**[number+1];  
 **for**(**int** i=0;i<number;i++){  
 array[i]=(**int**)(100\*Math.*random*());  
 }  
 **return** array;  
 }  
  
 **public static int**[] arrayNew(**int**[] array){  
 **for**(**int** i=0;i<array.**length**;i++){  
 **for**(**int** j=0;j<=i;j++){  
 **if**(array[i]<array[j]){  
 **int** temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
 }  
 }  
 **return** array;  
 }  
  
  
 **public static void** arrayPrint(**int**[] array){  
 **for**(**int** i=1;i<array.**length**;i++){  
 **if**(i%10==0){  
 System.***out***.print(array[i]+**"\n"**);  
 }**else** {  
 System.***out***.print(array[i]+**"\t"**);  
 }  
 }  
 }  
}

**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，先把前两个月的加起来，然后再加上5天即本年第几天，特殊情况，闰年且输入月份大于3需考虑多加一天。可定义数组存储1-12月各月天数。

**import** java.util.Scanner;  
  
**public class** Number {  
 **public static void** main(String[] args) {  
 Scanner input = **new** Scanner(System.***in***);  
 System.***out***.println(**"请输入年份："**);  
 **int** year = input.nextInt();  
 System.***out***.println(**"请输入月份："**);  
 **int** month = input.nextInt();  
 System.***out***.println(**"请输入日："**);  
 **int** day = input.nextInt();  
 **int** daySum = 0;  
 **int**[] monthDay = {31,28,31,30,31,30,31,31,30,31,30,31};  
 **for**(**int** i=0;i<month-1;i++){  
 daySum+=monthDay[i];  
 }  
 daySum+=day;  
 **if**(year%4==0 & year%100!=0 & month>3){  
 daySum+=1;  
 }  
 System.***out***.println(daySum);  
 }  
}

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** Number {  
 **public static void** main(String[] args) {  
 **int**[] arr = {1,5,7,9,14,22,37,66,68,80};  
 System.***out***.println(*numberReseach*(arr,0,arr.**length**-1,70));  
 }  
  
 **public static int** numberReseach(**int**[] arr,**int** start,**int** end,**int** findValue){  
  
 **if**(start<=end){  
 **int** middle = (start+end)/2;  
 **if**(findValue<arr[middle]){  
 **return** *numberReseach*(arr,start,middle-1,findValue);  
 }**else if**(findValue>arr[middle]){  
 **return** *numberReseach*(arr,middle+1,end,findValue);  
 }**else**{  
 **return** middle;  
 }  
 }**else**{  
 **return** -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(*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) / 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

**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. 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。

**import** java.util.Arrays;  
**public class** Number {  
 **public static void** main(String[] args) {  
 **int**[] A = {1,7,9,11,13,15,17,19};  
 **int**[] B = {2,4,6,8,10};  
 **int**[] C = *arrayNew*(A,B);  
 *arrayPrint*(C);  
 Arrays.*sort*(C);  
 *arrayPrint*(C);  
 }  
  
 **public static int**[] arrayNew(**int**[] a,**int**[] b){  
 **int** length = a.**length**+b.**length**;  
 **int**[] c = **new int**[length];  
 **int** count = 0;  
 **for**(**int** i=0;i<a.**length**;i++){  
 c[count++]=a[i];  
 }  
 **for**(**int** j=0;j<b.**length**;j++){  
 c[count++]=b[j];  
 }  
 **return** c;  
 }  
  
 **public static void** arrayPrint(**int**[] arr){  
 **for**(**int** i=0;i<arr.**length**;i++){  
 System.***out***.print(arr[i]+**"\t"**);  
 }  
 System.***out***.println();  
 }  
}

**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

public static void arraycopy(Object src,int srcPos,Object dest,int destPos,int length)

src:源数组；   
srcPos:源数组要复制的起始位置；   
dest:目的数组；   
destPos:目的数组放置的起始位置；   
length:复制的长度。   
注意：src and dest都必须是同类型或者可以进行转换类型的数组．

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();

}

}