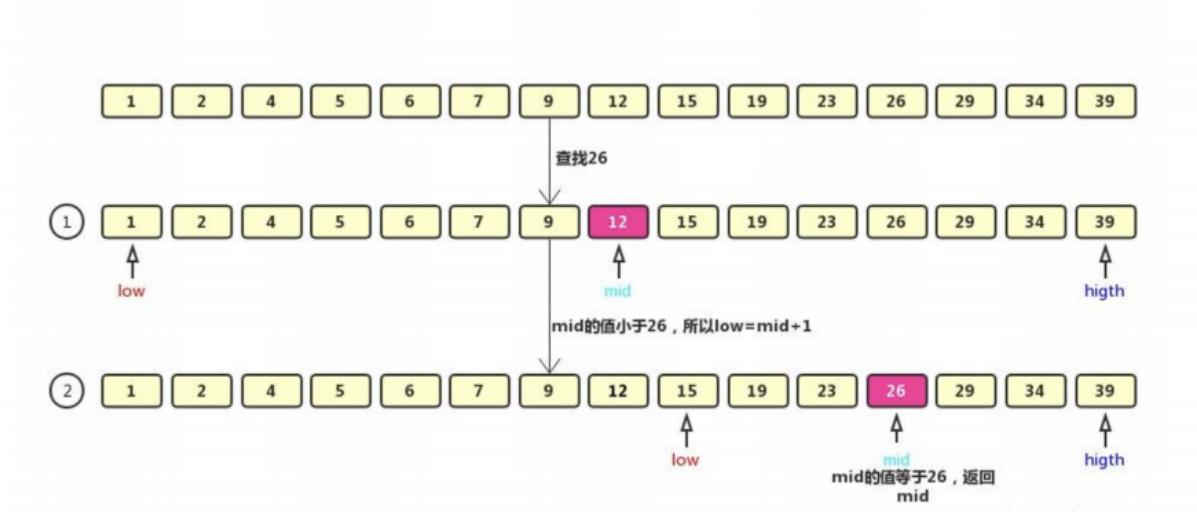
Java基础(7)[1]

二分查找

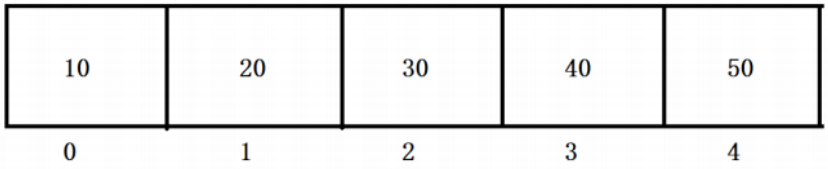
二分查找（Binary Search）算法，也叫折半查找算法。二分查找的思想非常简单，有点类似分治的思想。二分查找针对的是一个有序的数据集合，每次都通过跟区间的中间元素对比，将待查找的区间缩小为之前的一半，直到找到要查找的元素，或者区间被缩小为 0。



随机排名

分析一下随机排名的思路

> 1.在程序中录入数据存储起来 ---> 使用动态初始化数组的方式。
>
> 2.依次遍历数组中的每个数据。
>
> 3.每遍历到一个数据，都随机一个索引值出来，让当前数据与该索引位置处的数据进行交换。



核心思路：遍历过程中，每获取一个元素，同时随机产生一个索引，让当前索引位置的元素和随机索引位置的元素互换

当前索引：i = 0;
随机索引：index = 3;
arr[i]和arr[index]换位置

当前索引：i = 1;
随机索引：index = 2
arr[i]和arr[index]换位置

```java
public class Test3 {
public static void main(String[] args) {
      // 目标：完成随机排名
      // 1、定义一个动态初始化的数组用于存储5名员工的工号
      int[] codes = new int[5];
      // 2、提示用户录入5名员工的工号。
      Scanner sc = new Scanner(System.in);
      for (int i = 0; i < codes.length; i++) {
          // i = 0 1 2 3 4
          System.out.println("请您输入第" + (i + 1) +"个员工的工号：");
          int code = sc.nextInt();
          codes[i] = code;
      }
      // 3、打乱数组中的元素顺序。
      // [12, 33, 54, 26, 8]
      // i index
      Random r = new Random();
      for (int i = 0; i < codes.length; i++) {
         // codes[i]
         // 每遍历到一个数据，都随机一个数组索引范围内的值。
         //然后让当前遍历的数据与该索引位置处的值交换。
         int index = r.nextInt(codes.length); // 0 - 4
         // 定义一个临时变量记住index位置处的值
         int temp = codes[index];
         // 把i位置处的值赋值给index位置处
         codes[index] = codes[i];
         // 把index位置原来的值赋值给i位置处
         codes[i] = temp;
      }
      // 4、遍历数组中的工号输出即可
      for (int i = 0; i < codes.length; i++) {
         System.out.print(codes[i] + " ");
      }
 }
```

```
35  }
```

多维数组

Java 中的多维数组是一种数组的数组，即一个数组的元素也是一个数组。Java 中的多维数组可以包含任意数量的维度。在处理多维数组时，需要注意数组下标的范围，以避免数组越界异常。同时，还可以使用循环嵌套来遍历多维数组中的所有元素。

二维数组

Java中定义和操作多维数组的语法与一维数组类似。在实际应用中，三维及其以上的数组使用很少，主要使用二维数组。使用二维数组同一维数组的步骤，（1）定义数组、（2）为数组元素分配内存、（3）数组元素初始化、（4）使用数组。下面主要以二维数组为例进行讲解。

H5 定义二维数组

定义二维数组的语法规则如下：

`数据类型[][] 数组名;`

或者

`数据类型 数组名[][];`

语法解析：

[][] `[][]`表示二维数组，前面的[ ]表示第一维，后面的[ ]表示第二维。

[][] `[][]`放在数组名的前面或后面都是正确的。

H5 分配内存

```
1  int[][] arr = new int[3][4];
```

H5 数组元素初始化

```
1  // 二维数组初始化
2  int[][] arr = new int[3][4]; // 动态初始化
3  arr[0][0] = 1;
4  int[][] arr1 = new int[][]{ // 静态初始化
5      {1, 2, 3},
6      {2, 3},
7      {3, 4, 5, 4}
8  };
```

H5 二维数组的迭代

```
for (int i = 0; i < arr1.length; i++) {
    System.out.println(arr1[i]);
    for (int i1 = 0; i1 < arr1[i].length; i1++) {
        System.out.println(arr1[i][i1]);
    }
}
for (int[] t : arr1) {
    for (int a : t) {
        System.out.println(a);
    }
}
```

数组工具类

Arrays类

JDK中提供了一个专门用于操作数组的工具类，即Arrays类，位于java.util包中。该类提供了一些列方法来操作数组，如排序、复制、比较、填充等，用户直接调用这些方法即可，不需要自己编码实现，降低了开发难度。

Arrays类的常用方法

方法	返回类型	说明
equals(array1,array2)	boolean	比较两个数组是否相等
sort(array)	void	对数组array的元素进行排序
toString(array)	String	将一个数组array转换成一个字
copyOf(array,length)	与array数据类型一致	把数组array复制成一个长度为length的新数组
binarySearch(array,val)	int	查询元素值val在数组array中的下标
compare(array1,array2)	int	按字典顺序比较数组，前面的数组大，返回大于0的值，反之返回小于0的值
copyOfRange(arr,start,end)	与array数据类型一致	将指定数组的指定范围复制到新数组中。
fill(arr,start,end, val)	void	将指定的值分配给指定数组的指定范围的每个元素。
mismatch(array1,array2)	int	查找并返回两个数组之间第一个不匹配的索引，如果未找到不匹配，则返回 -1。
mismatch(array1,start1, end1,array2,start2,end2)	int	查找并返回指定范围内两个数组之间第一个不匹配的相对索引，如果未找到不匹配，则返回 -1。

Arrays类的应用

比较两个数组是否相等

Arrays类的equals()方法用于比较两个数组是否相等。只有当两个数组长度相等，对应位置的元素也一一相等时，该方法返回true，否则返回false。

对数组元素进行升序排序

Arrays类的sort()方法对数组的元素进行升序排序。

将数组转换成字符串

Arrays类中提供了专门输出数组内容的toString()方法。该方法用于将一个数组转换成一个字符串。它按顺序把多个数组元素连在一起，多个数组元素之间使用英文逗号和空格隔开。利用这种方法可以很清楚地观察到各个数组元素的值。

将数组所有元素赋值为相同的值

Arrays类的fill(array,val)方法用于把数组array的所有元素都赋值为val。

将数组赋值成一个长度为设定值的新数组

Arrays类的copyOf()方法把数组复制成一个长度为设定值的新数组。

> Arrays类的copyOf(array,length)方法可以进行数组复制，把原数据复制成一
>
> 个新数组，其中length是新数组的长度。如果length小于原数组的长度，则新
>
> 数组就是原数组的前面length个元素；如果length大于原数组的长度，则新数
>
> 组前面的元素就是原数组的所有元素，后面的元素是按照数组类型补充的默
>
> 认值，如整数补充0，浮点数补充0.0等。

System.arraycopy() 方法从指定的源数组复制一个数组，从指定位置开始，到目标数组的指定位置。该方法声明如下：

```
1  public static void arrcopy(Object src, int srcPos, Object dest,
   intdestPos, int length)
```

参数解析：

- src：这是源数组。
- srcPos：这是源数组中的起始位置。
- dest：这是目标数组。
- destPos：这是目标数据中的起始位置。
- length: 这是要复制的数组元素的数量。

数组组件的子序列从 src 引用的源数组复制到 dest 引用的目标数组。复制的组件数等于 length 参数。源数组中位置srcPos到srcPos + length - 1的元素被复制到目标数组的destPos 到 destPos + length - 1的位置。

查询元素在数组中的下标

Arrays类的binarySearch(Object[],Object key)方法用于查询数组元素在数组中的下标。调用该方法时要求数组中的元素已经按升序排列。如果key在数组中，则返回搜索值的索引；如果key不在数组中，返回值-1或“-”（插入点）。插入点的值有如下四种情况。

[1] 搜索值是数组元素，从0开始计数，得搜索值的索引值；

[2] 搜索值不是数组元素，且在数组范围内，从1开始计数，得“ - 插入点索引值”；

[3] 搜索值不是数组元素，且大于数组内元素，索引值为 – (length + 1);

[4] 搜索值不是数组元素，且小于数组内元素，索引值为 – 1。

1. Java学习第七天 ↩