## 数组和排序算法的应用 + 面试题

### 数组的定义与使用

数组是 Java 编程中最重要的数据结构之一，也是最基本的数据结构，Java 中的常用集合 ArrayList、HashMap 等内部的实现都使用到了数组结构。数组是只能用来存储一种类型的集合，可以通过下标访问数值中的所有元素。

数组的声明方式有以下两种，如整数型数组，请参考下面代码：

* 方式一：int[] arr;
* 方式二：int arr[];

大部分情况下，我们会使用第一种方式 int[] arr; 来声明数组。

**数组初始化**

数组可使用 new int[n] 进行初始化，每个元素初始化为 0，声明了 n 个元素。也可以直接赋值，例如 new int[]{ 1,2,3…… }，具体用法可参照下面代码：

// 初始化方式一  
int[] arr = new int[5];  
// 初始化方式二  
int[] arr2 = new int[]{1, 2, 3, 4, 5};  
// 初始化方式二的延伸版，可省略 new int[] 直接赋值  
int[] arr3 = {1, 2, 3, 4, 5};

**注意** ：在 Java 中，数组初始化如果声明了数组长度，则不能直接赋值。例如，int[] arr = new int[5]{1, 2, 3, 4, 5}; 给这段初始化数组长度并赋值时，编译器会报错，编译不通过。

### 数组遍历

数组遍历的常见方式有三种：传统的 for 循环、for each 遍历、还有 JDK 8 中新增的 Lambda 表达式。具体的实现请参考以下实例。

方式一：传统 for 循环

Integer[] arr = {2, 3, 6, 7, 9};  
// 方式一：传统 for  
for (int i = 0; i < arr.length; i++) {  
 System.out.println(arr[i]);  
}

方式二：for each

Integer[] arr = {2, 3, 6, 7, 9};  
// 方式二：for each  
for (int i : arr) {  
 System.out.println(i);  
}

方式三：JDK 8 中的 Lambda 表达式

Integer[] arr = {2, 3, 6, 7, 9};  
// 方式三：jdk 8 Lambda  
Arrays.asList(arr).forEach(x -> System.out.println(x));

其中 for each 的方式，写法更简洁，也更不容易出错，不必为数组的越界而担心（大于元素的最大下标值）。

**注意** ：数组的访问是从 0 开始，而不是 1 开始，也就是第一个元素的获取是 arr[0]，而非 arr[1]。

### 数组拷贝

数组拷贝使用的是 Arrays.copyof() 方法，具体实现请参考下面代码：

int[] arr = {3, 4, 9};  
int[] arr2 = Arrays.copyOf(arr, arr.length);  
System.out.println(Arrays.toString(arr2));

程序执行结果：[3, 4, 9]

**注意** ：Arrays.copyOf(array,newLength) 第二个参数 newLength 表示声明此数组的长度，可以比拷贝的数组的长度长，多出来的元素会初始化为 0 值。

### 数组填充与合并

##### **数组填充**

即为每个元素统一赋值，使用 Arrays.fill() 进行数组填充，具体实现请参考下面代码：

int[] arr = new int[10];  
Arrays.fill(arr, 6);  
System.out.println(Arrays.toString(arr));

程序执行结果：[6, 6, 6, 6, 6, 6, 6, 6, 6, 6]

**注意** ：使用 Arrays.fill() 会覆盖原有的值，即使数组之前有赋值操作，也会被覆盖。

##### **数组合并**

使用 org.apache.commons.lang3.ArrayUtils.addAll() 方法，具体实现请参考下面代码：

int[] arr = {2, 8, 13, 11, 6, 7};  
int[] arr2 = {66, 88};  
// 合并数组  
int[] arr3 = org.apache.commons.lang3.ArrayUtils.addAll(arr, arr2);  
System.out.println(Arrays.toString(arr3));

程序执行结果：[2, 8, 13, 11, 6, 7, 66, 88]

### 排序与算法

##### **数组排序**

使用 Arrays.sort() 方法，具体实现请参考下面代码：

int[] arr = {2, 8, 13, 11, 6, 7};  
Arrays.sort(arr);  
System.out.println(Arrays.toString(arr));

程序执行结果：[2, 6, 7, 8, 11, 13]

##### **数组逆序**

使用 org.apache.commons.lang3.ArrayUtils.reverse(arr) 方法，具体实现请参考下面代码：

int[] arr = {2, 8, 13, 11, 6, 7};  
int[] arr = {2, 8, 13, 11, 6, 7};  
// 数组正序（排序）  
Arrays.sort(arr);  
// 数组逆序  
org.apache.commons.lang3.ArrayUtils.reverse(arr);  
System.out.println(Arrays.toString(arr));

程序执行结果：[13, 11, 8, 7, 6, 2]

**注意** ：org.apache.commons.lang3.ArrayUtils.reverse() 是数组逆序，并不是数组倒序，也就是说 ArrayUtils.reverse() 只会把数组原顺序颠倒输出，并不会自然排序后再倒序输出。

##### **冒泡排序**

依次比较相邻的两个数，把较大的值放后面，执行整个循环之后，数组就从小到大进行排列了。具体实现请参考下面代码：

int[] arr = {2, 8, 13, 11, 6, 7};  
System.out.println("排序前：" + Arrays.toString(arr));  
for (int i = 0; i < arr.length; i++) {  
 // 因为冒泡是把每轮循环中较大的数飘到后面，所以是 arr.length-i-1  
 for (int j = 0; j < arr.length - i - 1; j++) {  
 if (arr[j] > arr[j + 1]) {  
 // 元素交换  
 int temp = arr[j + 1];  
 arr[j + 1] = arr[j];  
 arr[j] = temp;  
 }  
 }  
}  
System.out.println("排序后：" + Arrays.toString(arr));

程序执行结果：

排序前：[2, 8, 13, 11, 6, 7]  
排序后：[2, 6, 7, 8, 11, 13]

##### **选择排序**

每次从待排序的数据元素中选出最小（或最大）的一个元素，顺序放在已排好序的数列的最后，直到全部待排序的数据元素排完。具体实现请参考下面代码：

int[] arr = {2, 8, 13, 11, 6, 7};  
System.out.println("排序前：" + Arrays.toString(arr));  
for (int i = 0; i < arr.length; i++) {  
 int lowerIndex = i;  
 for (int j = i + 1; j < arr.length; j++) {  
 // 找出最小的一个索引  
 if (arr[j] < arr[lowerIndex]) {  
 lowerIndex = j;  
 }  
 }  
 // 交换  
 int temp = arr[i];  
 arr[i] = arr[lowerIndex];  
 arr[lowerIndex] = temp;  
}  
System.out.println("排序后：" + Arrays.toString(arr));

程序执行结果：

排序前：[2, 8, 13, 11, 6, 7]  
排序后：[2, 6, 7, 8, 11, 13]

关于更多的排序算法，后面会有专门的章节进行介绍。

### 元素查找

查找数组是否包含某个值，使用 Arrays.binarySearch() 方法查询。 Arrays.binarySearch() 是利用二分法查询某个值，如果查到包含某值会返回该值的下标，如果没有查到则返回负值。

int[] arr = {1, 3, 4, 5};  
// Arrays.binarySearch() 使用二分法查询某值  
int index = Arrays.binarySearch(arr, 5);  
System.out.println(index);

**注意** ：使用 Arrays.binarySearch 之前一定要先调用 Arrays.sort() 对数组进行排序，否则返回的结果有误。

### 多维数组

我们之前使用的数组可以称之为一维数组，而多维数组可以理解为数组的数组，可以用二维数组来举例，二维数组也是一种特殊的多维数组。

比如我们声明一个二维数组：int[][] arr = new int[2][4];

这就相当于我们创建了一个两行四列的表，它的使用、赋值与取值，请查看下面代码示例：

// 声明二维数组  
int[][] arr = new int[2][4];  
//循环二维数组  
for (int i = 0; i < arr.length; i++) {  
 for (int j = 0; j < arr[0].length; j++) {  
 // 二维数组赋值  
 arr[i][j] = j;  
 }  
}  
// 二维数组取值  
System.out.println(arr[0][1]);  
// 打印二维数组  
System.out.println(Arrays.toString(arr[0]));  
System.out.println(Arrays.toString(arr[1]));

以上程序执行的结果是：

1  
[0, 1, 2, 3]  
[0, 1, 2, 3]

### 数组类型转换

##### **字符串转数组**

使用 split 分隔字符串就形成了数组，请参考以下代码：

String str = "laowang,stone,wanglei";  
String[] arr = str.split(","); // 字符串转数组  
System.out.println(arr[0]);

##### **数组转字符串**

使用 Arrays.toString() 方法，请参考以下代码：

String[] arr = {"laowang", "stone", "wanglei"};  
String str = Arrays.toString(arr);  
System.out.println(str);

若要查看更多数组转字符串的方式，请查看本文面试部分的介绍。

##### **数组转集合**

使用 Arrays.asList() 方法，请参考以下代码：

String[] strArr = {"cat", "dog"};  
List list = Arrays.asList(strArr);  
System.out.println(list);

##### **集合转数组**

使用 List.toArrray() 方法，请参考以下代码：

List<String> list = new ArrayList<String>();  
list.add("cat");  
list.add("dog");  
// 集合转换为数组  
String[] arr = list.toArray(new String[list.size()]);  
System.out.println(Arrays.toString(arr));

### 相关面试题

#### 1. 数组和集合有什么区别？

答：数组和集合的区别如下：

* 集合可以存储任意类型的对象数据，数组只能存储同一种数据类型的数据；
* 集合的长度是会发生变化的，数组的长度是固定的；
* 集合相比数组功能更强大，数组相比集合效率更高。

#### 2. 以下代码访问数组元素打印的结果是多少？

int[] arr = new int[5] {1, 2, 3, 4, 5};  
System.out.println(arr[4]);

答：程序编译报错，在 Java 中初始化数组时，如果直接给数组赋值，不能声明数组长度；如果声明了数组长度，则不能赋值给数组，否则编译器报错。

正确的写法如下：

int[] arr = new int[]{1, 2, 3, 4, 5};  
System.out.println(arr[4]);

输出的结果为：5，访问元素从 0 开始。

#### 3. 执行以下代码会输出什么结果？

public static void main(String[] args) {  
 int[] arr = {2, 3, 4, 8};  
 change(arr);  
 System.out.println(arr[2]);  
}  
private static void change(int[] arr) {  
 for (int i = 0; i < arr.length; i++) {  
 if (i % 2 == 0) {  
 arr[i] \*= i;  
 }  
 }  
}

答：输出的结果是 8。

题目解析：在 Java 中数组本质是引用类型，因此在调用方法中修改数组，就是对原数组本身的修改。

#### 4. 以下程序打印的结果是多少？

int[] intArr = new int[3];  
String[] StrArr = new String[3];  
System.out.println(intArr[1]);  
System.out.println(StrArr[1]);

答：以上程序打印的结果是：0 和 null。

题目解析：new int[3] 相当于声明了数组的长度为 3，每个元素初始化为 0，而 new String[3] 相当于声明了数组的长度为 3，每个元素初始化为 null。

#### 5. 数组转换字符串有哪些方式？

答：数组转换字符串，有以下几种方式。

方式一：遍历拼接，完整代码如下：

String[] arr = {"laowang", "stone", "wanglei"};  
StringBuffer sb = new StringBuffer();  
for (int i = 0; i < arr.length; i++) {  
 sb.append(arr[i]);  
 if (i != arr.length - 1)  
 sb.append(",");  
}  
System.out.println(sb.toString());

方式二：Arrays.toString() 转换，完整代码如下：

String[] arr = {"laowang", "stone", "wanglei"};  
String str2 = Arrays.toString(arr);  
System.out.println(str2);

方式三：StringUtils.join() 转换，完整代码如下：

String[] arr = {"laowang", "stone", "wanglei"};  
String str3 = StringUtils.join(Arrays.asList(arr), ","); // 使用英文逗号分隔  
System.out.println(str3);

#### 6. 数组遍历有哪几种方式？

答：常见的数组遍历有以下三种方式。

* 传统 for 循环，如 for (int i = 0; i < arr.length; i++) { //…… }
* for each 循环，如 for (int i : arr) { //…… }
* jdk 8 Lambda 方式，如 Integer[] arr = {2, 3, 6, 7, 9}; Arrays.\_asList\_(arr).forEach(x -> System.\_out\_.println(x));

#### 7. 以下数组比较的结果分别是什么？

String[] strArr = {"dog", "cat", "pig", "bird"};  
String[] strArr2 = {"dog", "cat", "pig", "bird"};  
System.out.println(Arrays.equals(strArr, strArr2));  
System.out.println(strArr.equals(strArr2));  
System.out.println(strArr == strArr2);

答：上面代码执行的结果，分别为：true、false、false。

题目解析：strArr == strArr2 为引用比较，因此结果一定是 false，而数组本身的比较也就是 strArr.equals(strArr2) 为 false 的原因是因为数组没有重写 equals 方法，因此也是引用比较。数组 equals 源码实现如下：

public boolean equals(Object obj) {  
 return (this == obj);  
}

而 Arrays.equals 的结果之所以是 true 是因为 Arrays.equals 重写了 equals 方法。源代码实现如下：

public static boolean equals(Object[] a, Object[] a2) {  
 if (a==a2)  
 return true;  
 if (a==null || a2==null)  
 return false;  
 int length = a.length;  
 if (a2.length != length)  
 return false;  
 for (int i=0; i<length; i++) {  
 Object o1 = a[i];  
 Object o2 = a2[i];  
 if (!(o1==null ? o2==null : o1.equals(o2)))  
 return false;  
 }  
 return true;  
 }

#### 8. 以下程序使用 Arrays.binarySearch 返回的结果是 true 还是 false？

String[] arr = {"dog", "cat", "pig", "bird"};  
int result = Arrays.binarySearch(arr, "bird");  
System.out.println(result == -1);

答：返回的结果是：true。

题目解析：使用 Arrays.binarySearch 之前一定要先调用 Arrays.sort() 对数组进行排序，否则返回的结果有误，本数组返回的结果是 ﹣1，是因为没有使用排序的结果，正确的使用请查看以下代码：

String[] arr = {"dog", "cat", "pig", "bird"};  
Arrays.sort(arr);  
int result = Arrays.binarySearch(arr, "bird");  
System.out.println(result == -1);

#### 9. Arrays 对象有哪些常用的方法？

答：Arrays 常用方法如下：

* Arrays.copyOf() 数组拷贝
* Arrays.asList() 数组转为 List 集合
* Arrays.fill() 数组赋值
* Arrays.sort() 数组排序
* Arrays.toString() 数组转字符串
* Arrays.binarySearch() 二分法查询元素
* Arrays.equals() 比较两个数组的值

#### 10. 查询字符串数组中是否包含某个值有几种方法？

答：常见查询数组中是否包含某个值有以下两种方式：

* 方式一：Arrays.asList(array).contains(“key”);
* 方式二：Arrays.binarySearch(array, “key”);

具体的实现代码如下：

String[] arr = {"doc", "pig", "cat"};  
// 方式一：Arrays.asList(array).contains  
boolean bool = Arrays.asList(arr).contains("cat");  
System.out.println(bool);  
// 方式二：Arrays.binarySearch  
Arrays.sort(arr);  
boolean bool2 = Arrays.binarySearch(arr, "cat") > -1;  
System.out.println(bool2);

#### 11. 如何修改数组的第三个到第五个元素的值为 6？

答：本题考察的知识点显然不是使用 for 循环修改那么简单，而是考察对 Arrays.fill() 方法的掌握，以下提供了两种实现方式可供参考。

方式一：for 循环方式

int[] arrInt = new int[10];  
for (int i = 0; i < arrInt.length; i++) {  
 if (i >= 2 && i < 5) {  
 arrInt[i] = 6;  
 }  
}

方式二：Arrays.fill() 方式

int[] arrInt = new int[10];  
Arrays.fill(arrInt, 2, 5, 6);

### 总结

在 Java 中数组本质是引用类型，数组只能用来存储固定大小的同类型元素。在 Java 中很多集合的内部都是依赖数组实现的，如 ArrayList 和 HashMap 等。数组的冒泡排序和选择排序也是面试常考的内容，很多公司会要求面试者手写冒泡排序。本文也介绍了数组、字符串和集合之间的相互转换，只有掌握好这些技能才能开发出更好的 Java 程序。

[点击此处下载本文源码](https://github.com/vipstone/java-%20interview/tree/master/interview-code/src/main/java/com/interview)

## 更多资源下载交流请加微信：Morstrong,加入永久会员,网盘更新更快捷！

# 本资源由微信公众号：光明顶一号，提供支持