原 java初、中、高级面试题必备——数据结构与算法-数组和字符串

2019年06月25日 16:22:15 在IT中穿梭旅行 阅读数 36 编辑

数组

173.数组简介

数组是数据结构中的基本模块之一。因为**字符串**的英文由字符数组形成的，所以二者是相似的。大多数面试问题都属于这个范畴。

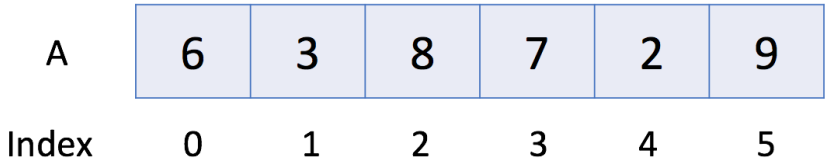
下面介绍以下问题：

1. 了解**数组**状语从句：**动态数组**之间的区别;
2. 熟悉数组和动态数组中的**基本操作**;
3. 理解**多维数组**并能够掌握**二维数组**的使用;
4. 明白**字符串**的概念以及字符串所具有的不同特性;
5. 运用能够**双指针技巧**解决实际问题。
6. **数组**状语从句：**动态数组**之间有什么**不同**?
7. 在你常用的语言中，状语从句：数组数组动态对应的**内置数据结构**的英文什么?
8. 在如何**数组**中执行初始化，数据访问，修改，迭代，排序等**基本操作**?
9. 在如何**动态数组**中执行初始化，数据访问，修改，迭代，排序，添加，删除等**基本操作**?

174.数组简介

数组是一种基本的数据结构，用于按顺序**存储元素的集合**。但是元素可以随机存取，数组因为中的每个元素都可以通过数组**索引**来识别。

。数组可以有一个或多个维度这里我们从**一维数组**。开始，它也被称为线性数组这里有一个例子：



在上面的例子中，数组A中有6个元素。也就是说，A的长度是6.我们可以使用A [0]来表示数组中的第一个元素。因此，A [0] = 6。类似地，A [1] = 3，A [2] = 8，依此类推。

数组中的操作

```
// "static void main" must be defined in a public class.
public class Main {
    public static void main(String[] args) {
        // 1. Initialize
        int[] a0 = new int[5];
        int[] a1 = {1, 2, 3};
        // 2. Get Length
        System.out.println("The size of a1 is: " + a1.length);
        // 3. Access Element
        System.out.println("The first element is: " + a1[0]);
        // 4. Iterate all Elements
        System.out.print("[Version 1] The contents of a1 are:");
        for (int i = 0; i < a1.length; ++i) {
            System.out.print(" " + a1[i]);
        }
        System.out.println();
        System.out.print("[Version 2] The contents of a1 are:");
        for (int item: a1) {
            System.out.print(" " + item);
        }
        System.out.println();
        // 5. Modify Element
        a1[0] = 4;
        // 6. Sort
        Arrays.sort(a1);
```

```
    }
}


//输出结果

/*
The size of a1 is: 3
The first element is: 1
[Version 1] The contents of a1 are: 1 2 3
[Version 2] The contents of a1 are: 1 2 3
*/
```

175.动态数组介绍

数组具有**固定的容量**，我们需要在初始化时指定数组的大小。有时它会非常不方便并可能造成浪费。

因此，大多数编程语言都提供内置的**动态数组**，它仍然是一个随机存取的列表数据结构，但**大小是可变的**.Java中使用**ArrayList。（数据结构是动态数组）**

```
// "static void main" must be defined in a public class.
public class Main {
    public static void main(String[] args) {
        // 1. initialize
        List<Integer> v0 = new ArrayList<>();
        List<Integer> v1;                       // v1 == null
        // 2. cast an array to a vector
        Integer[] a = {0, 1, 2, 3, 4};
        v1 = new ArrayList<>(Arrays.asList(a));
        // 3. make a copy
        List<Integer> v2 = v1;                  // another reference to v1
        List<Integer> v3 = new ArrayList<>(v1);     // make an actual copy of v1
        // 3. get length
        System.out.println("The size of v1 is: " + v1.size());;
        // 4. access element
        System.out.println("The first element in v1 is: " + v1.get(0));
        // 5. iterate the vector
```

```
        System.out.print("[Version 1] The contents of v1 are:");
        for (int i = 0; i < v1.size(); ++i) {
            System.out.print(" " + v1.get(i));
        }
        System.out.println();
        System.out.print("[Version 2] The contents of v1 are:");
        for (int item : v1) {
            System.out.print(" " + item);
        }
        System.out.println();
        // 6. modify element
        v2.set(0, 5);       // modify v2 will actually modify v1
        System.out.println("The first element in v1 is: " + v1.get(0));
        v3.set(0, -1);
        System.out.println("The first element in v1 is: " + v1.get(0));
        // 7. sort
        Collections.sort(v1);
        // 8. add new element at the end of the vector
        v1.add(-1);
        v1.add(1, 6);
        // 9. delete the last element
        v1.remove(v1.size() - 1);
    }
}


//输出结果
/*
The size of v1 is: 5
The first element in v1 is: 0
[Version 1] The contents of v1 are: 0 1 2 3 4
[Version 2] The contents of v1 are: 0 1 2 3 4
The first element in v1 is: 5
The first element in v1 is: 5
*/
```

176.二维数组简介

类似于一维数组，二维数组也是由元素的序列组成。但是这些元素可以排列在矩形网格中而不是直线上。

```
// "static void main" must be defined in a public class.
public class Main {
    private static void printArray(int[][] a) {
        for (int i = 0; i < a.length; ++i) {
            System.out.println(a[i]);
        }
        for (int i = 0; i < a.length; ++i) {
            for (int j = 0; a[i] != null && j < a[i].length; ++j) {
                System.out.print(a[i][j] + " ");
            }
            System.out.println();
        }
    }
    public static void main(String[] args) {
        System.out.println("Example I:");
        int[][] a = new int[2][5];
        printArray(a);
        System.out.println("Example II:");
        int[][] b = new int[2][];
        printArray(b);
        System.out.println("Example III:");
        b[0] = new int[3];
        b[1] = new int[5];
        printArray(b);
    }
}
```


字符串

177.字符串介绍

实际上字符串的英文一个 **unicode 字符**数组。

在java中，我们**可能无法使用** “==”来比较两个字符串。当我们使用“==”时，它实际上会比较这两个对象是否是同一个对象。

字符串被初始化，你就无法改变它的内容。字符串是不可变的

如下列例子：

```
// "static void main" must be defined in a public class.
public class Main {
    public static void main(String[] args) {
        // initialize
        String s1 = "Hello World";
        System.out.println("s1 is \"" + s1 + "\"");
        String s2 = s1;
        System.out.println("s2 is another reference to s1.");
        String s3 = new String(s1);
        System.out.println("s3 is a copy of s1.");
        // compare using '=='
        System.out.println("Compared by '==':");
        // true since string is immutable and s1 is binded to "Hello World"
        System.out.println("s1 and \"Hello World\": " + (s1 == "Hello World"));
        // true since s1 and s2 is the reference of the same object
        System.out.println("s1 and s2: " + (s1 == s2));
        // false since s3 is refered to another new object
        System.out.println("s1 and s3: " + (s1 == s3));
        // compare using 'equals'
        System.out.println("Compared by 'equals':");
        System.out.println("s1 and \"Hello World\": " + s1.equals("Hello World"));
        System.out.println("s1 and s2: " + s1.equals(s2));
        System.out.println("s1 and s3: " + s1.equals(s3));
        // compare using 'compareTo'
        System.out.println("Compared by 'compareTo':");
        System.out.println("s1 and \"Hello World\": " + (s1.compareTo("Hello World") == 0));
        System.out.println("s1 and s2: " + (s1.compareTo(s2) == 0));
        System.out.println("s1 and s3: " + (s1.compareTo(s3) == 0));
```

```
29      }30  }
```

字符串其他操作

```
1  public class Main {
2      public static void main(String[] args) {
3          String s1 = "Hello World";
4          // 1. concatenate
5          s1 += "!";
6          System.out.println(s1);
7          // 2. find
8          System.out.println("The position of first 'o' is: " + s1.indexOf('o'));
9          System.out.println("The position of last 'o' is: " + s1.lastIndexOf('o'));
10         // 3. get substring
11         System.out.println(s1.substring(6, 11));
12     }
13 }
14
15 /*
16 Hello World!
17 The position of first 'o' is: 4
18 The position of last 'o' is: 7
19 World
20 */
```


例如，如果字符串的长度是 N，那么查找操作和子字符串操作的 时间复杂度是O(N)。

178.不可变字符串 - 问题和解决方案

字符串是不可变的，则会带来一些问题。下面我们提供解决方案。

修改操作

显然，不可变字符串无法被修改。哪怕你只是想修改其中的一个字符，也必须创建一个新的字符串。

字符串连接。 让我们来看一个在for循环中重复进行字符串连接的例子：

```
// "static void main" must be defined in a public class.
public class Main {
    public static void main(String[] args) {
        String s = "";
        int n = 10000;
        for (int i = 0; i < n; i++) {
            s += "hello";
        }
    }
}
```

在Java中，由于字符串是**不可变的**，因此在连接时首先为新字符串分配足够的空间，复制旧字符串中的内容并附加到字字串。

因此，总时间复杂度将是：

$$
\begin{aligned}
&5 + 5\times2 + 5\times3 + \ldots + 5\times n \\
&= 5\times (1 + 2 + 3 + \ldots + n) \\
&= 5\times n\times (n + 1) / 2,
\end{aligned}
$$

也就是。 **O(n2)**

解决方案

1.如果你确实希望你的字符串是可变的，则可以将其转换为字符数组。

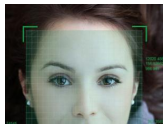
```
public class Main {
    public static void main(String[] args) {
        String s = "Hello World";
        char[] str = s.toCharArray();
        str[5] = ',';
        System.out.println(str);
    }
}
```

```
9
10 //输出结果   hello,world
```

2.如果你经常必须连接字符串，最好使用一些其他的数据结构，如 StringBuilder 。以下代码以 O(n) 的复杂度运行。

```
1  public class Main {
2      public static void main(String[] args) {
3          int n = 10000;
4          StringBuilder str = new StringBuilder();
5          for (int i = 0; i < n; i++) {
6              str.append("hello");
7          }
8          String s = str.toString();
9      }
10 }
```

下一章二分查找：https://blog.csdn.net/weixin_38201936/article/details/93743464



人脸识别主要算法原理

人脸识别