数组概念：

所谓数组，是有序的元素序列，在内存中是顺序存储结构， 若将有限个类型相同的变量的集合命名，那么这个名称为数组名。组成数组的各个变量称为数组的分量，也称为数组的元素，有时也称为下标变量。用于区分数组的各个元素的数字编号称为下标。数组是在程序设计中，为了处理方便， 把具有相同类型的若干元素按无序的形式组织起来的一种形式。 [1] 这些无序排列的同类数据元素的集合称为数组。

相关操作：

1、插入和删除：由于数组的内存地址的连续性，当元素被插入或者删除的时候，数组的后续元素都需要移动。如果插入到最后一个位置，或者删除最后一个位置，则时间复杂度为o(1)，如果插入或者删除是在第一个位置,,则后面的其他元素都需要移动，则时间复杂度为o(n),平均的时间复杂度为o((n-1)/2),即o(n);

2、查找：对于顺序存储结构来说，要实现查找的操作，只要将下标对应的返回值返回即可，时间复杂度为o(1)

二、java数组

:数组定义格式：数据类型[] 数组名 = new 数据类型[数组的长度];

数组的初始化动态初始化

- A:什么是数组的初始化 * 就是为数组开辟连续的内存空间，并为每个数组元素赋予值
- B:如何对数组进行初始化 * a:动态初始化 只指定长度，由系统给出初始化值 * int[] arr = new int[5]; * b:静态初始化 给出初始化值，由系统决定长度
- C:动态初始化的格式： * 数据类型[] 数组名 = new 数据类型[数组长度]; 数组的内存图解1一个数组)
- A:画图演示 * 一个数组： 数组的地址赋值给arr，而不是整个实体进行赋值 数组的内存图解2二个数组
- A:画图演示
- 二个不同的数组 数组的初始化静态初始化
- A:静态初始化的格式： * 格式：数据类型[] 数组名 = new 数据类型[]{元素1,元素2,…}; * 简化格式： * 数据类型[] 数组名 = {元素1,元素2,…};
- B:案例演示 * 对数组的解释 * 输出数组名称和数组元素

数组操作的两个常见小问题越界和空指针：

- A:案例演示 * a:ArrayIndexOutOfBoundsException:数组索引越界异常 * 原因：你访问了不存在的索引。 * b:NullPointerException:空指针异常 * 原因：数组已经不在指向堆内存了。而你还用数组名去访问元素。 * int[] arr = {1,2,3}; * arr = null; * System.out.println(arr[0]);

三、数组的常见操作及算法：

1、数组的操作1遍历

- A:案例演示 * 数组遍历：就是依次输出数组中的每一个元素。 * 数组的属性:arr.length数组的长度 * 数组的最大索引:arr.length - 1; public static void print(int[] arr) { if(arr==null || arr.length()==0) return; for (int i = 0;i < arr.length ;i++) { System.out.print(arr[i] + " "); } } 时间复杂度：o(n) 测试数据：null，数组长度为0，长度>0 2、数组的操作2获取最值
- A:案例演示 * 数组获取最值(获取数组中的最大值最小值 public static int getMax(int[] arr) { int max = arr[0]; for

(int i = 1;i < arr.length ;i++) { //从数组的第二个元素开始遍历 if (max < arr[i]) { //如果max记录的值小于的数组中的元素 max = arr[i]; //max记录住较大的 } } return max; } 时间复杂度，取平均值为o(n) 3、数组的操作3反转：

- A:案例演示 * 数组元素反转(就是把元素对调) 方法一：这种方法不创建新的数组，时间复杂度为0(n); public static void reverseArray(int[] arr) { for (int i = 0;i < arr.length / 2 ; i++) { //arr[0]和arr[arr.length-1-0]交换 //arr[1]和arr[arr.length-1-1]交换 //arr[2]和arr[arr.lentth-1-2] //... int temp = arr[i]; arr[i] = arr[arr.length-1-i]; arr[arr.length-1-i] = temp; } }

方法二：创建新的数组,时间复杂度哦o(n)

public static void reverse1(int[] arrays) {
if (arrays == null || arrays.length == 0) {
return null;
}
int[] reverseArray = new int[arrays.length];
for (int i = arrays.length - 1; i >= 0; i--) {
reverseArray[i] = arrays[arrays.length - 1-i];
}

```
}
```

4、 * 数组查表法(根据键盘录入索引,查找对应星期)

```
            public static char getWeek(int week) {
                  char[] arr = {' ','一','二','三','四','五','六','日'};            //定义了一张星期表
                  return arr[week];                                  //通过索引获取表中的元素
            }
```

5、数组的操作5基本查找

- A:案例演示 * 数组元素查找(查找指定元素第一次在数组中出现的索引) public static int getIndex(int[] arr,int value) { if(arr==null|| arr.length==0) return -1; //注意边界条件 for (int i = 0;i < arr.length ;i++) { //数组的遍历 if (arr[i] == value) { //如果数组中的元素与查找的元素匹配 return i; } } return -1; }

测试数据：arr=null, arr=new int[5], arr=new int[]{1,2,3,4,5} ;
