数组

一、什么是数组

数组：是一种容器，可以用来存储同种数据类型的多个值

- 数组容器在存储数据的时候，需要结合隐式转换考虑。
- 例如：int类型的数组容器 （boolean byte short int double）
- 例如：double类型的数组容器（byte short int long float double）
- 建议：容器的类型，和存储的数据类型保持一致

二、数组的静态初始化

数组的静态初始化

- 初始化：就是在内存中，为数组容器开辟空间，并将数据存入容器中的过程
- 完整格式：数据类型[] 数组名 = new 数据类型[] { 元素1，元素2，元素3... };
- 范例：int[] array = new int[]{ 11，22，33 };
- 范例：double[] array2 = new double[] { 11.1，22.2，33.3};
- 简化格式：数据类型[] 数组名 = { 元素1，元素2，元素3... };
- 范例：int[] array = { 11，22，33 };
- 范例：double[] array2 = { 11.1，22.2，33.3};

三、数组的动态初始化

- **数组动态初始化**

  动态初始化：初始化时只指定数组长度，由系统为数组分配初始值。

  格式：数据类型[] 数组名 = new 数据类型[数组长度];

  范例：int[] arr = new int[3];

> *数组默认初始化值的规律*
> - *整数类型:默认初始化值 0*
> - *小数类型:默认初始化值 0.0*
> - *字符类型:默认初始化值 '\u0000' 空格*
> - *布尔类型:默认初始化值 false*
> - *引用数据类型: 默认初始化值 null（引用数据类型包括：类、接口类型、数组类型、枚举类型、注解类型，字符串型）*

四、数组的静态初始化和动态初始化的区别

数组动态初始化和静态初始化的区别

动态初始化：手动指定数组长度，由系统给出默认初始化值。

只明确元素个数，不明确具体数值，推荐使用动态初始化

- 举例：使用数组容器来存储键盘录入的5个整数。
- int[] arr = { ? ? ? ? ? };
- int[] arr = new int[5];

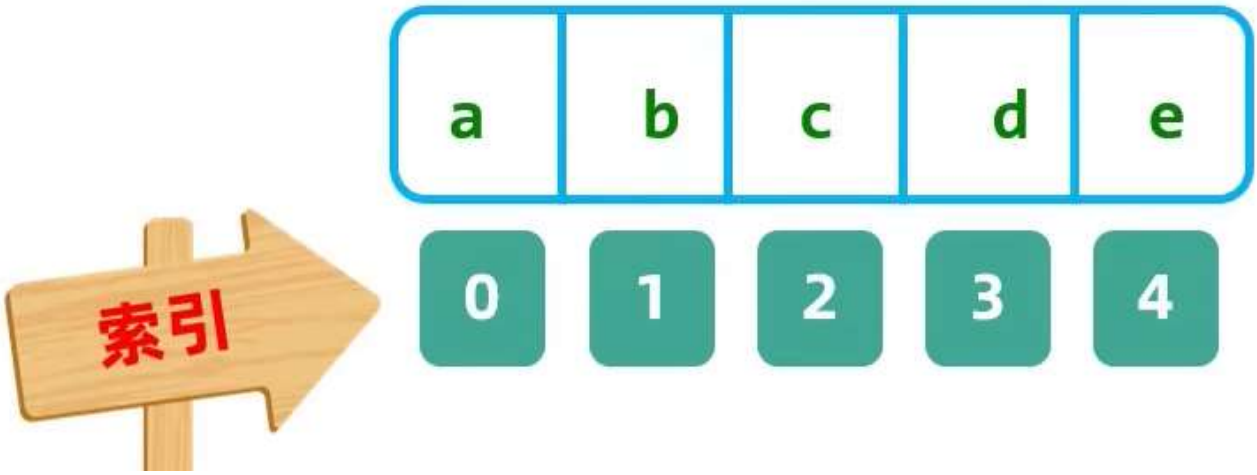
静态初始化：手动指定数组元素，系统会根据元素个数，计算出数组的长度。

需求中已经明确了要操作的具体数据，直接静态初始化即可。

- 举例：将全班的学生成绩存入数组中 11，22，33
- int[] arr = {11, 22, 33};

五、索引

- 数组元素访问
- 数组元素访问的格式为：数组名[索引];
- 索引：也叫做下标，角标；
- 索引特点：从0开始，逐个+1增长，连续不间断
- （我们可以通过索引把数组中的元素拿出来用，也可以把元素添加到数组当中去）

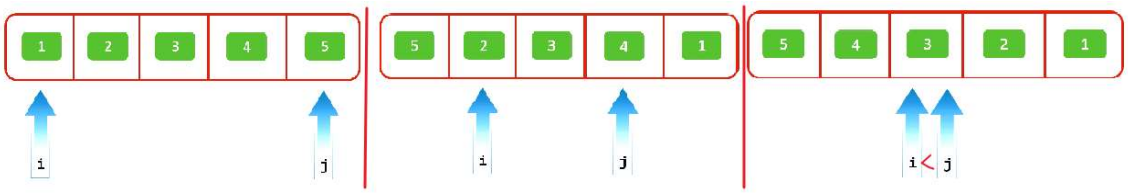



```
int[] arr = {1,2,3,4,5};
//获取数组中的第一个元素
//其实就是0索引上对应的元素
// int number = arr[0];
// System.out.println(number);//1
//获取数组中1索引上对应的数据，并直接打印出来
// System.out.println(arr[1]);//2


//2.把数据存储到数组当中
//格式:    数组名[索引] = 具体数据/变量;
//细节：一旦覆盖之后，原来的数据就不存在了。
arr[0] = 100;

System.out.println(arr[0]);//100
```

六、数组角标越界异常

- 访问了数组不存在索引，就会引发数组角标越界异常
- 避免：知道索引的范围

七、数组常见操作

- 1.求最值
  - ```java
    @Test
    public void arrayMaxTest() {
        //思路：先赋值一个最大值。
        int[] num={40,55,15,28,66,97,16,92,99};
        int max=num[0];
        for (int i = 1; i < num.length; i++) {
            if(num[i]>max){
                max=num[i];
            }
        }
        System.out.println(max);
    }
    ```
- 2.求和
  - ```java
    //动态初始化一个数组，数组长度是10
    int[] num= new int[10];
    //定义一个变量存放所有数据的和
    int sum=0;
    //填充数组
    for (int i = 0; i < num.length; i++) {
        num[i]= (int) (Math.random()*100+1);
        System.out.print(num[i]+",");
        sum=sum+num[i];
    }
    ```
- 3.交换数据
  - 


```java
//方法二:将数组中的 0角标位置，与最大角标位置呼唤。
    // 思路：在循环中，定义两个变量 i 和 j 用来记录数组两边的 角标
     // 让其 一个自增，一个自减
    // 当 左边 i 角标 不小于 右边j 角标的话，就说明交换到中间位置了，就可以停止了
//记住：对数组的操作，就是对角标的操作

int[] arr = {1,2,3,4,5};
//定义一个临时变量，用来记录值
int item=0;
for (int i = 0,j = arr.length-1; i < j; i++,j--) {
    item=arr[i];
    arr[i]=arr[j];
    arr[j]=item;
}
for (int i : arr) {
    System.out.println(i);
}
```

- **4.打乱数据**

```java
/**
 * 打乱数组中的数据
 *
 * 主要思路是，每次便利，都使用当前索引上的值，
 * 与随机索引去进行数据互换。最终便利完全后就是随机打乱的数组了
 */

@Test
public void arrayRandomTest() {
    //年龄数组
    int[] age={55,63,18,29,78,90,150,66};
    //定义一个临时变量
    int item=0;
    // 生成一个随机数，要求范围在0-数组的长度之间
    Random random = new Random();
    for (int i = 0; i < age.length; i++) {
        int sub = random.nextInt(age.length); //范围[0,8)
        item=age[i];
        age[i]=age[sub];
        age[sub]=item;
        System.out.println(sub);
    }
    for (int i : age) {
        System.out.print(i+" ");
    }

}
```

- **5.冒泡排序**

```java
/**
 * 数组排序
 * 将数组中的数据，按照从大到小的顺序进行排序
 * 使用冒泡排序法
 *  第位与第n+1位比较，大的放在前面，循环
 * {2,4,5,1,66,32,88}
 */
@Test
public void maopaoTest() {

    int[] a={2,4,5,1,66,32,88};
    int item=0;
    boolean flag = false;//用于优化冒泡排序，判断是否进行过交换
    for (int j = 0; j < a.length-1; j++) {
        for (int i = 0; i < a.length-1-j; i++) {
            //三角交换
            if(a[i]<a[i+1]){
                item=a[i];
                a[i]=a[i+1];
                a[i+1]=item;
                flag = true;
            }
        }
        System.out.println("第"+(j+1)+"趟"+ Arrays.toString(a));
        //如果没有进入三角交换则证明数组已经有序，直接退出循环即可
        //如果进入了三角交换，把flag赋值为false，来判断下一次循环是否进入三角交换
        if (flag == false){
            break;
        }else {
            flag = false;
        }
    }

    for (int i : a) {
        System.out.print(i+" ");
    }

}
```

- **6.数组拷贝**

```java
/**
 * @param arr 要被拷贝的数组
 * @param from 拷贝开始的 索引
 * @param to   拷贝结束的 索引
 * @return 返回拷贝后的数组
 */
public static int[] copyArray(int[] arr,int from, int to){

    //[from,to)
     //动态初始化 新数组
    int[] newArray=new int[to-from];

    //伪造索引思想
    int index=0;

     for (int i = from; i < to; i++) {
         newArray[index]=arr[i];
         index++;
     }
     return newArray;
}
```