java_5

本节内容：
数组，排序，循环打印，冒泡排序

数组(数据结构之一)

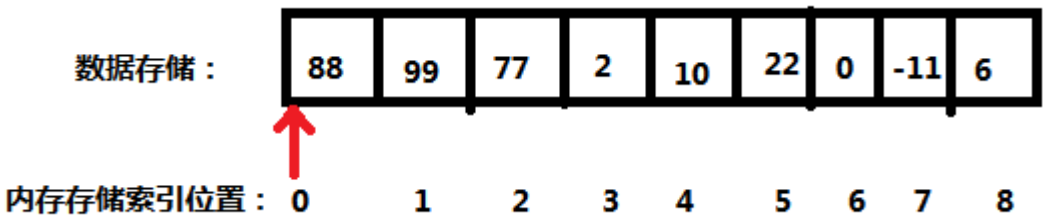
如军人训练，是整齐，有序，高效，报数快（相当于查找快速），每个士兵都有自己的固定位置，固定的编号，能够高效的执行上官下达的一个个命令，与之很相似的java中的一**个数据结构**则是**数组**，那么数组自然也拥有整齐，**有序**，**高效**之方式。

数组是java重要的**数据结构之一**，在Java 语言中提供的数组是用来存储**固定大小**的**同类型**元素，是一种特殊的数组，里面的元素按一定的**顺序**排列（或顺序存储），优点是查询效率快，删除和插入元素效率慢，因此，对于有序数组更适合用于**查询**，很好的实现逻辑上的**顺序表**。

数组在内存中的顺序存储样子：内存是由一个个连续的内存单元组成，每一个内存单元都有自己固定的编号位置地址，在这些内存单元中，有些被其它存储放置的数据占用，也有些是处于空闲位置，等待放置存储数据。

数组中每一个元素，存储于内存单元中（一个个的小格子区块），并且元素之间是紧密排列，既不能打乱元素的存储顺序，也不能跳过某个存储单元进行存储，因此必须是**有序进行存储**。

数组存储是从下标**0索引**位置开始，如图：



数组是将**同一种类型**数据存储在一起。

声明一个数组变量，如 numbers[100] 来代替直接声明 100 个独立变量 number0，number1，....，number99。（开发使用当中声明变量不上接放置长度，是为了防止取值时**数组越界**而报错，因此可以不用设置固定变量，代码好拓展）。

数组变量声明方式

声明数组变量的语法及代码方式：

```
String[] strs;
int[] ints;
double[] dous;
float[] flos;
......
```

创建数组

Java使用**new**操作符来创建不同类型的数组，语法如下：

```
String[] strs = new String[3];
int[] ints = new int[3];
double[] dous = new double[3];
float[] flos = new float[3];
```

步聚：

```
1).使用String[] strs创建了一个数组
2).把新创建的数组的引用赋值给变量 strs
```

数组变量的声明，和创建数组可以用一条语句完成，如：

```
String[] strs = new String[3];
......
```

还可以使用{}的方式创建数组:

```
String[] strs2 = {"江苏省","广西省","湖南省"};
```

数组的元素是通过索引访问，数组索引从 0 开始，索引值从 0 到 strs2.length-1。

创建一个省份数组

声明一个数组变量为 pros，创建固定大小为 5 的String 类型元素的数组，并且把它的引用赋值给 pros变量。

1).省份数组显示值代码编写：

```
public class ArrsDemo {

    public static void main(String[] args) {
        showProvince();
    }

    /**
     * 获取省份数组值
     */
    public static void showProvince() {
        //创建一个省数组，大小固定为5
        String[] pros = new String[5];
        pros[0] ="江苏省";
        pros[1] ="广西省";
        pros[2] ="湖南省";
        pros[3] ="江西省";
        pros[4] ="河南省";

        //循环输出值
```

```
        for (int i = 0; i < pros.length; i++) {
            System.out.println(pros[i]);//通过索引获取数组中的每个值
        }
    }

}
```

2).输出结果：

```
江苏省
广西省
湖南省
江西省
河南省
```

*：输出流程用**断点方式**演示。

数组循环输出

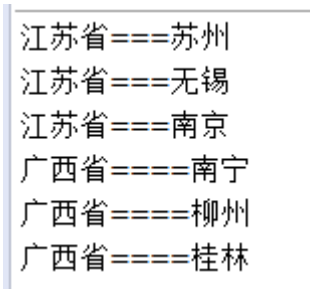
数组的元素类型和数组的大小都是确定的，所以当处理数组元素时候，可以使用基本循环或者 For-Each（增强 for）循环。

1).代码编写：

```
/**
     * for多重循环获取数组省市值
     */
    public static void showProCity() {
        //创建一个省数组，两个省分，对应城市三个
        String[] pros ={"江苏省","广西省"};
        String[] city1 ={"苏州","无锡","南京"};
        String[] city2 ={"南宁","柳州","桂林"};

        for (int i = 0; i < pros.length; i++) {
            if(pros[i].equals("江苏省")){//如果是"江苏省"则取出这个省份下的城市
                for (int j = 0; j < city1.length; j++) {
                    System.out.println(pros[i]+"==="+city1[j]);
                }
            }
            if(pros[i].equals("广西省")){
                for (int z = 0; z < city2.length; z++) {
                    System.out.println(pros[i]+"===="+city2[z]);
                }
            }
        }
    }
```

2).运行结果：



For-Each 循环

使用 For-Each 循环或者**加强**型(**增强**型)循环，能在不使用下标的情况下**遍历**数组值[代码变得简洁]。

语法格式：

```
for(type element: array)
{
    System.out.println(element);
}
```

1).代码编写：

```
public static void showForEachPro() {
        String[] pros ={"江苏省","广西省","湖南省","江西省","湖北省"};
        for(String pro : pros){//循环
            System.out.println(pro);
        }
}
```

2).运行结果：

数组作为函数的参数

数组可以作为参数传递给方法。

```
public class ArrsDemo {

    public static void main(String[] args) {
        int[] ints = {101,100,88,99};
```

```
        //参数传递给方法
        showInts(ints);
        //或是使用这样showInts(new int[]{101,100,88,99});
    }

    public static void showInts(int[] ints) {
        for (int i = 0; i < ints.length; i++) {
            System.out.println("输出的数字为："+ints[i]);
        }
    }
}
```

输出结果：

```
输出的数字为：101
输出的数字为：100
输出的数字为：88
输出的数字为：99
```

数组作为函数的返回值

```
public class ArrsDemo {

    public static void main(String[] args) {
        String[] newPros = showStrs();
        StringBuffer appendPro = new StringBuffer();//对字符串进行修改使用类，Buffer翻译为缓存区或可以理解为临时存放区
        for (String pro : newPros) {
            appendPro.append(pro +",");
        }

        System.out.println("原始的：=="+appendPro);
        System.out.println("截取：=="+appendPro.substring(0,appendPro.length()-1));

        //分割返回数组再取值
        String[] strSplit =
appendPro.toString().substring(0,appendPro.length()-1).split(",");
        for (String split : strSplit) {
            System.out.println(split);
        }
    }

    public static String[] showStrs() {
        String[] pros ={"江苏省","广西省","湖南省","江西省","湖北省"};
        return pros;
    }
}
```

实例中 pros 数组作为函数的**返回值**。

注：长度固定之下，假如插入值超过数组长度，会出现什么状况？数据是不是会撑爆掉，然后出现的错误提示（数组**越界**异常），因此如出现越界可以对数组做**扩容**（ 旧数组长度*2 ），然后使用 System.arraycopy(array,0,arraynew,0,array.length);

```
/**
 * 当超过长度时，数组进行扩容
 * @author Administrator
 *
 */
public class ArrayTest {
    public static void main(String[] args) {
        int[] ints = new int[2];
        int[] newInts = new int[ints.length*2];
        ints[0] = 9;
        ints[1] = 8;
        //ints[2] = 7;//此处插入数组报数组越界异常
        //旧数组复制到新数组
        System.arraycopy(ints, 0, newInts, 0, ints.length);
        ints = newInts;
        //ints数组长度已经是4，因此插入数据没有问题
        ints[2] = 7;
        for (int i = 0; i < ints.length; i++) {
            System.out.println(ints[i]);//输出时最后一位没有放置数据自然为空
        }
    }

}
```

Arrays 类

java.util.Arrays 类能方便地操作数组，提供所有方法都是静态的，直接调用即可。

功能(查看源码了解)：

```
1).查看数组长度
2).对数组排序：通过 sort 方法,按升序
3).比较数组：通过 equals 方法比较数组中元素值是否相等
4).遍历数组
5).数组中是否包含某一个值
```

代码编写：

```
  public class ArrayDemo {

    public static void main(String[] args) {
        //showlen();
        /*showSore();*/
        //showEquals();
        String loginName = " 罗祖民 ";
        boolean f=runComparaEq(loginName);
```

```
        System.out.println(f?"有此人":"无此人");
    }

    //数组长度
    public static void showlen() {
        int[] ints ={1,44,6,7,99,0,33};
        System.out.println("数组的长度："+ints.length);
    }

    //array  sore排序
    public static void showSore() {
        int[] ints ={1,44,6,7,99,0,33};
        //遍历数组
        for (int i : ints) {
            System.out.println("没有排序前的输出数字："+i);
        }
        Arrays.sort(ints);//排序  底层有一个sort的方法帮我们编排序
        for (int i : ints) {
            System.out.println("Arrays排序后的输出数字："+i);
        }

        //输出是降序
        System.out.println();
        for (int i = ints.length - 1; i >= 0; i--) {
            System.out.print(ints[i] + "\t");
        }
    }

    //比较登录
    /**
     * 数组是否存在一个值
     */
    public static boolean runComparaEq(String loginName) {
        String[] names = {"李欢","罗祖民","朱明爽"};
        boolean flag = false;
        for (String n : names) {
            if(n.equals(loginName.trim())){//trim()方法：前后去空格
                //System.out.println(loginName);
                flag = true;
            }
        }
        return flag;
    }

    public static void showEquals() {
        String[] pros ={"江苏省","湖北省","广西省"};
        /*String[] prosEq ={"湖北省"};
        for (String pro : pros) {
            if(pro.equals("湖北省")){
                 System.out.println(pro);
            }
        }*/
```

```
        System.out.println(Arrays.equals(pros, prosEq));//false
    }

}
```

冒泡排序

```
package com.fy.demo;

import java.util.Arrays;
/**
 * 冒泡排序:最主是两两数字比较，位置交换(交换相领排序或叫上浮)
 * @author Administrator
 * @date   2020年4月22日
 *
 */
public class DemoBubbleSort5 {

    public static void main(String[] args) {
        int[] newSortNums=bubbleSortArrays();
        for (int i = 0; i < newSortNums.length; i++) {
            System.out.print(newSortNums[i]+"\t");
        }
    }

    public static int[] bubbleSortArrays(){
        int[] arrays = {88,20,50,100,66,-2};//@15db9742对象地址


        int in = 0;
        int n = 0;
        int fn = 0;
        for(int i = 0;i<arrays.length-1;i++){//外循环控制多少轮结束
            boolean flag = true;//顺序是正确
            //针对arrays进行排序
            for(int j= 0;j<arrays.length-1-i;j++){//把比出来的最大的数字挑下去不做比较
                if(arrays[j]>arrays[j+1]){//两数比较，当大于时，两数位置进行转换
                //if(arrays[j]<arrays[j+1]){
                    //把第一个数字先存在一个临时的位置变量中
                    int temp = arrays[j];
                    arrays[j] = arrays[j+1];//交换元素值
                    arrays[j+1] = temp;
                    flag = false;
                }
                in++;
                System.out.println(Arrays.toString(arrays));
            }
            System.out.println("**********************");
            if(flag){//顺序已经是正确，不需要再做比较
                fn++;
                break;
            }
```

```
            }

            n++;

            System.out.println("内层循环输出次数:"+in);
            System.out.println("外层循环输出次数:"+n);
            System.out.println("比较顺序break次数:"+fn);

            return arrays;
        }
    }
```

*：查看api:https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html