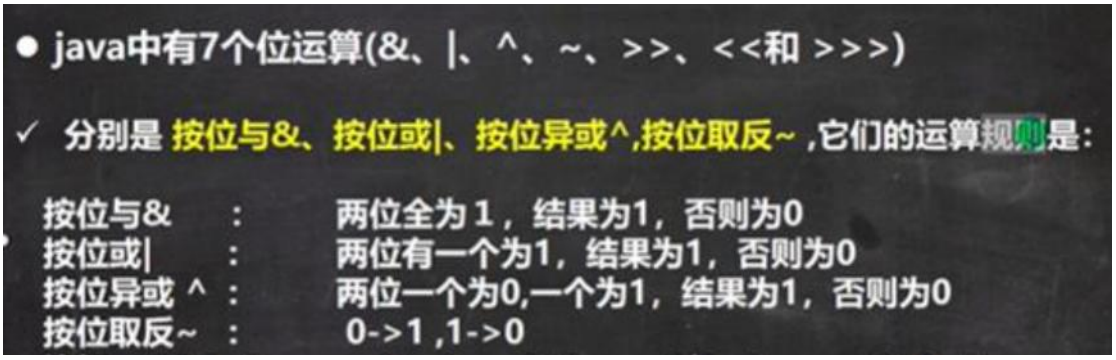
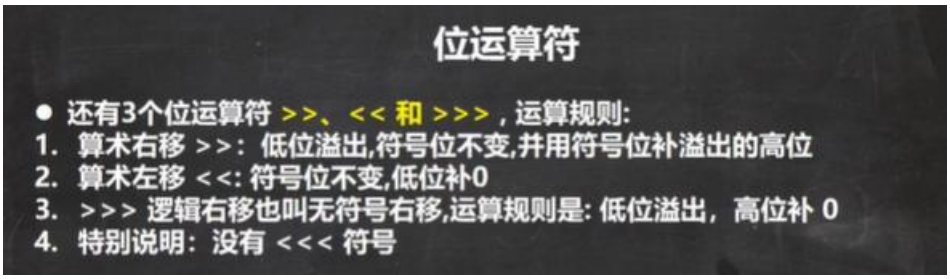




*原码，反码，补码

1.对于有符号的而言:二进制的最高位是符号位 : 0 表示正 , 1 表示 负

2.正数的原码 ,反码,补码都一样

3. 负数的反码=原码符号位不变 ,其他位取反 (0 -> 1 , 1 ->0)

4. 负数的补码=反 码 + 1 , 负 数 的 反 码 = 补 码 - 1

5. 0 的反码,补码都是 0

6. java 没有无符号数 , java 中所有数都有符号

7.在计算机运算的时候都是以补码的方式运算的

8.当我们看运算结果时要看原码

**位运算符 运算时化作二进制数（4 个字节 8 位） 内容看练习

>>算数右移：低位溢出符号位不变，并用符号位补溢出的高位 本质是 a>>b a/2^b

<<算数左移：符号位不变，低位补 0 本质是 a<<b a*2^b

>>>逻辑右移(无符号右移):低位溢出，高位补 0

If 语句直接练习

单支 if

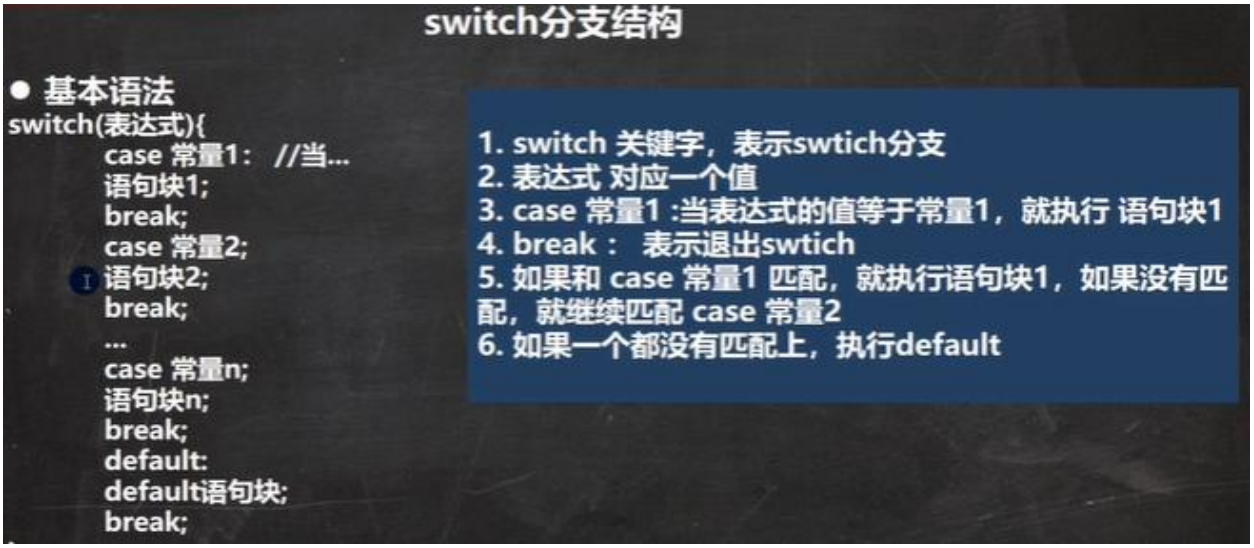
双分支 if else

多分支 if ... else if ...else if ... else

嵌套

```
// 接收字符方法：获得字符串的第一个字符
char gender = myScanner.next().charAt(0);
```

Switch



注意：

细节 1：表达式数据类型，要和 case 后的数据类型一致，或者可以自动转成可以比较的类型

细节 2：case 后面的常量不能重复

```
char c = 'b';
switch(c){
    case 'a':
    System.out.println("ok1");
    break;
    case 98: // 细节1：表达式数据类型，要和case后的数据类型一致，或者可以自动转成可以比较的类型
    System.out.println("ok2");
    break;
    case 'b'://细节2：case后面的常量不能重复
    System.out.println("ok3");
    break;
```

```
C:\Users\Administrator\Desktop\baoyan\lyc\第二周\视频中练习文件>javac SwitchDetail.java
SwitchDetail.java:12: 错误: case 标签重复
                    case 'b'://细节2：case后面的常量不能重复
                         ^
1 个错误
```

Switch 穿透现象

```
case 3:
case 4:
case 5:
    System.out.println("春季");
    break;
case 6:
case 7:
case 8:
    System.out.println("夏季");
    break;
```

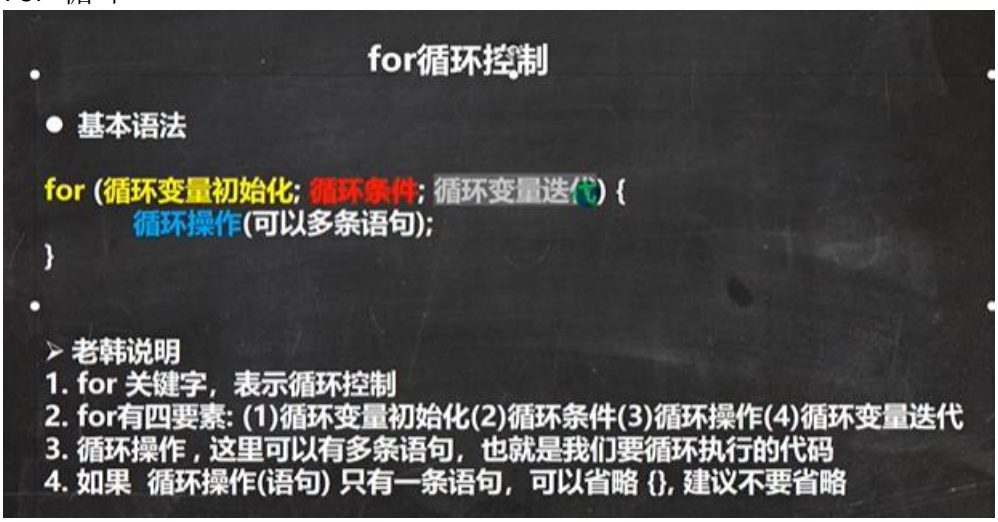
当多个常量同时判断为一种情况时可以进行穿透操作，方便。

注意：Switch 中表达式的返回值 必须是 byte short int Char String enum(枚举) 类

型

Switch 和 if 比较

For 循环



For 循环注意事项

1) 循环条件时返回一个布尔值的表达式

2）for(;循环判断条件;)种初始化和变量迭代可以写到其他地方但“；”不能省略（可扩大变量作用范围）

举例：int i=0;

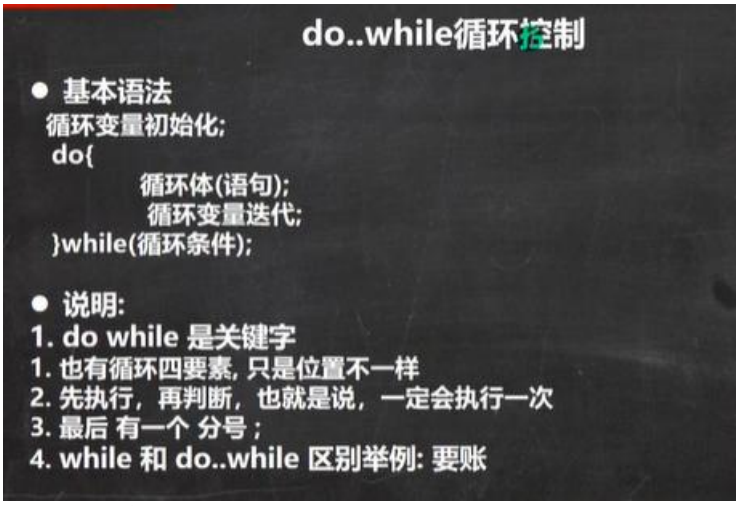
```
For(;i<10;){
System.out.println(“aaa”);
i++;//这一定要加上
}
```

3）循环初始值可以有多条初始化语句，但要求类型一样中间用逗号隔开，循环变量迭代也可以有多条变量迭代语句，中间用逗号隔开

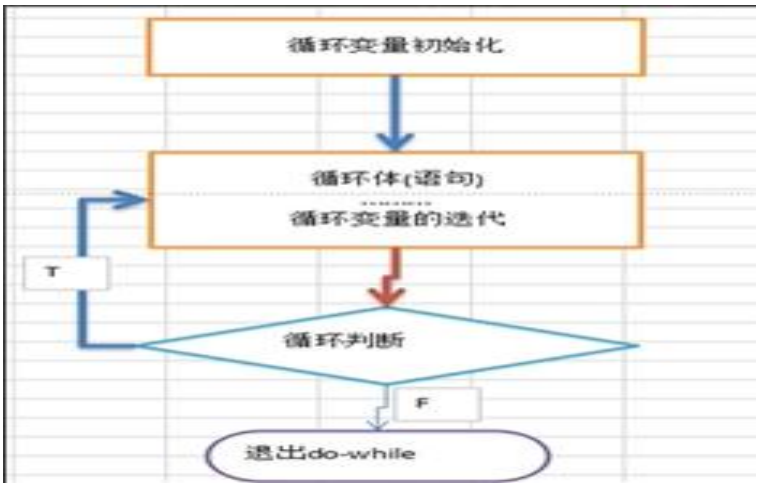
While 用法

具体看练习程序

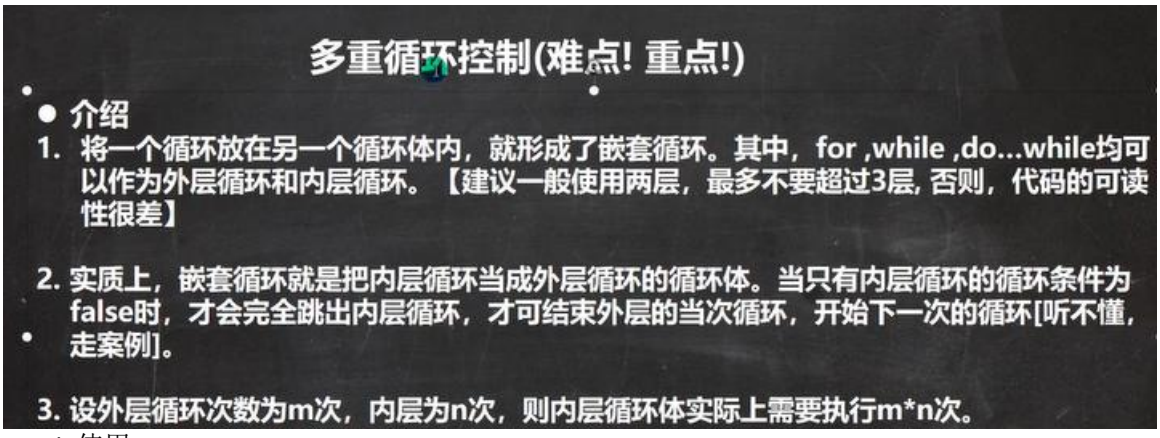
Do while　相比 while 在显示菜单时用



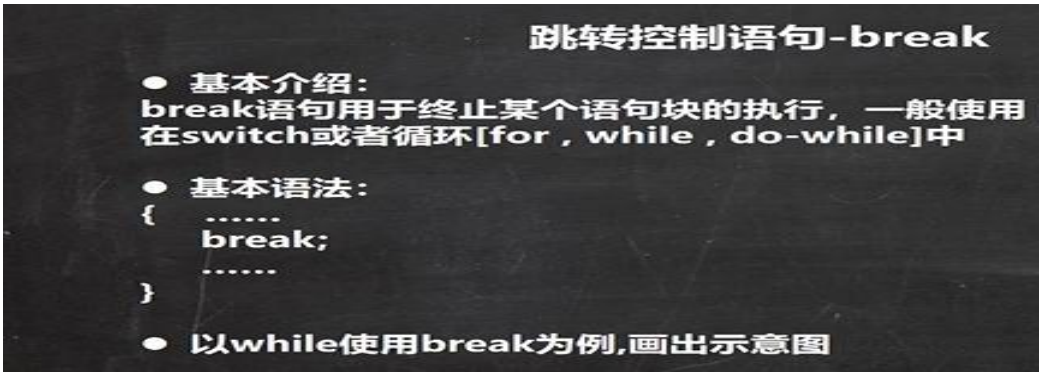
Do while 流程图

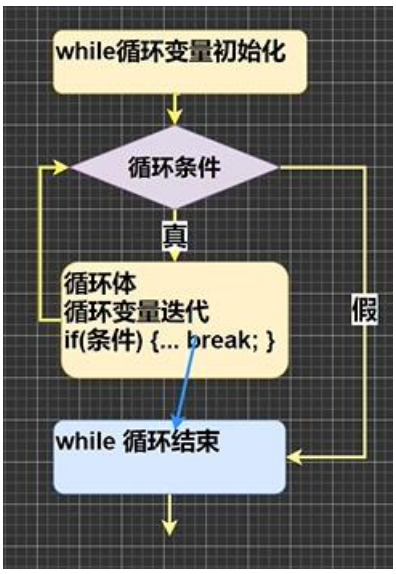


多重循环



Break 使用

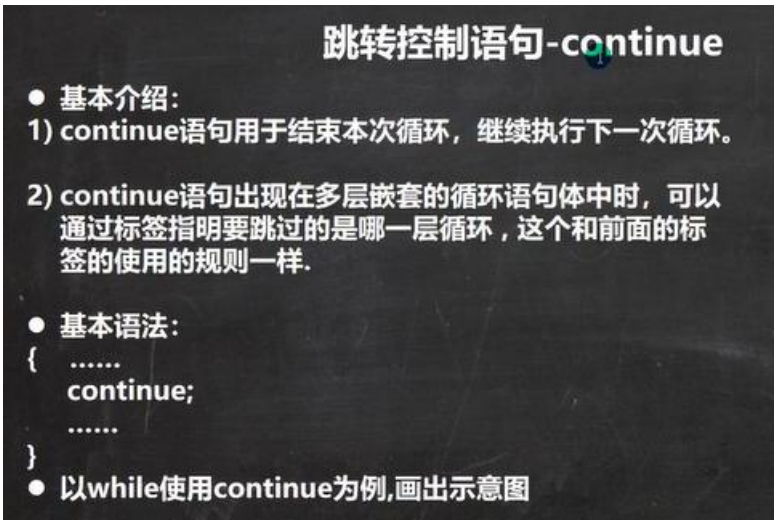




当 if 条件为真 执行 break 跳转到循环结束这一步

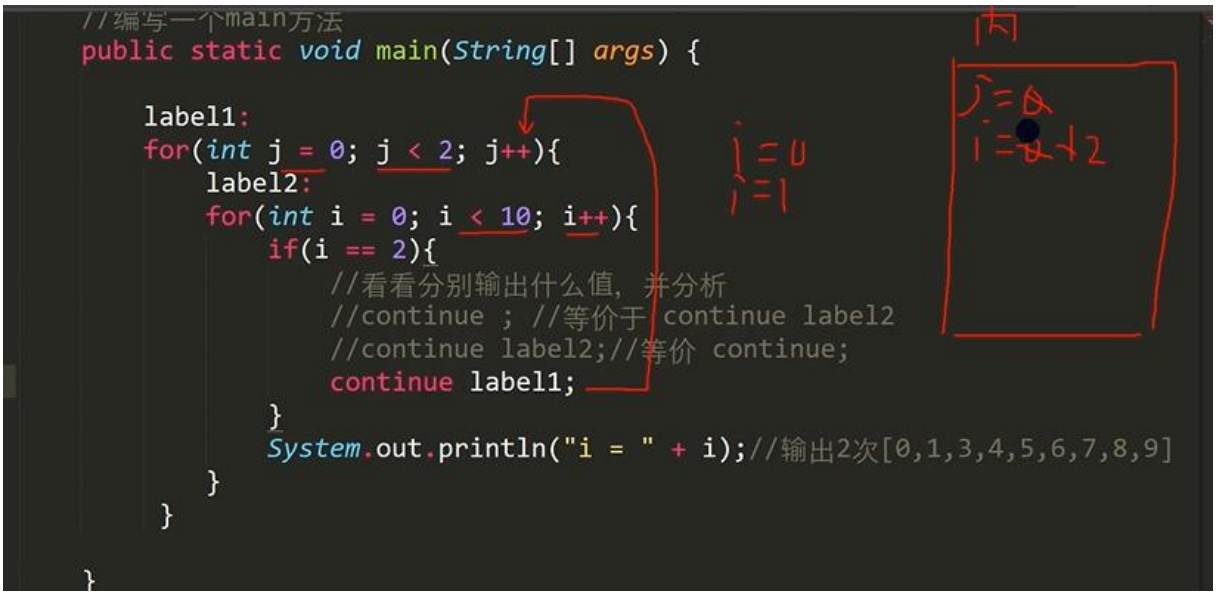
Break 提前结束循环 并未退出程序

跳转控制语句 continue



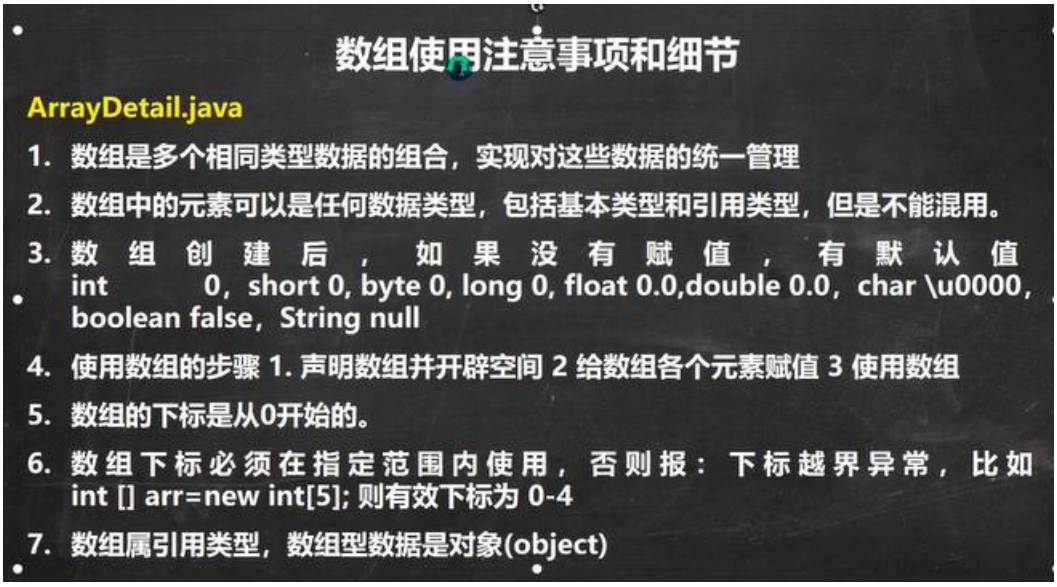
其中 continue 默认跳转到的是最近得循环

可以指定跳转到哪个循环 就是在 continue 后面加上 标签号



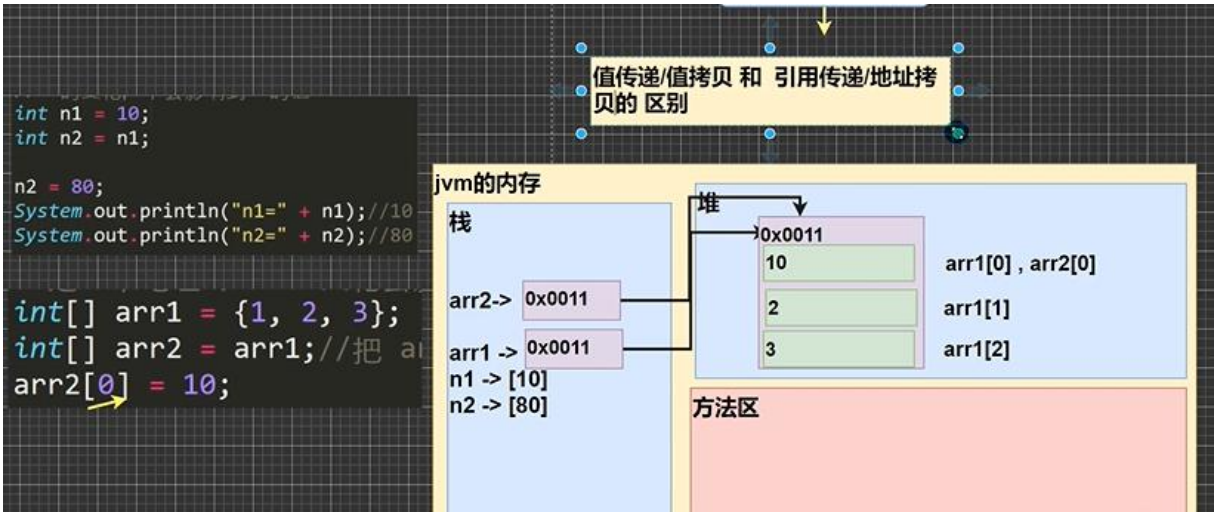
跳转控制语句 return

数组细节



数组赋值一般情况下是引用传递，赋的值是地址

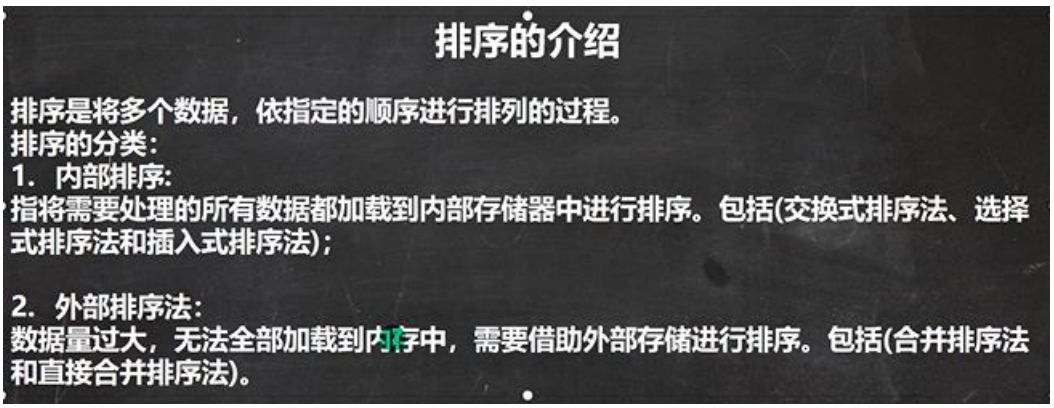
值传递与引用传递得区别



数组反转 2 逆序赋值法

空间变换讲解如图

```java
int[] arr = {11, 22, 33, 44, 55, 66};
//使用逆序赋值方式
//老韩思路
//1. 先创建一个新的数组 arr2 ,大小 arr.length
//2. 逆序遍历 arr ,将 每个元素拷贝到 arr2的元素中(顺序拷贝)
//3. 建议增加一个循环变量 j -> 0 -> 5
int[] arr2 = new int[arr.length];
//逆序遍历 arr
for(int i = arr.length - 1, j = 0; i >= 0; i--, j++) {
    arr2[j] = arr[i];
}
//4. 当for循环结束, arr2就是一个逆序的数组 {66, 55, 44,33, 22, 11}
//5. 让 arr 指向 arr2数据空间, 此时 arr原来的数据空间就没有变量引用
//   会被当做垃圾, 销毁
arr = arr2;
System.out.println("====arr的元素情况=====");
```



冒泡排序对于五个数得数组来说

分析冒泡排序

数组 [24,69,80,57,13]

第1轮排序: 目标把最大数放在最后

第1次比较[24,69,80,57,13]

第2次比较[24,69,80,57,13]

第3次比较[24,69,57,80,13]

第4次比较[24,69,57,13,80]

第2轮排序: 目标把第二大数放在倒数第二位置

第1次比较[24,69,57,13,80]

第2次比较[24,57,69,13,80]

第3次比较[24,57,13,69,80]

第3轮排序: 目标把第3大数放在倒数第3位置

第1次比较[24,57,13,69,80]

总结冒泡排序特点

1. 我们一共有5个元素
2. 一共进行了 4轮排序, 可以看成是外层循环
3. 每1轮排序可以确定一个数的位置，比如第1轮排序确定最大数,第2轮排序， 确定第2大的数位置，依次类推
4. 当进行比较时， 如果前面的数大于后面的数，就交换
5. 每轮比较在减少 4->3->2->1

顺序查找就是从头到尾遍历