数组

定义：长度固定内容可变的用来同一数据类型的存储数据的容器

\u----代表u16进行编码

格式：

格式一：

数据类型[] 数组名=new 数据类型[]{元素1,元素2,元素3,........};

数组名：变量名，引用名 符合标示符的定义规范

数组的长度：元素的个数

该种格式在定义数组时就给定了数组元素的值；

**int**[] ages=**new** **int**[]{12,13,56,45};

格式二：

数据类型[] 数组名={元素1,元素2,元素3,........};

该格式是格式一的简化，数组元素的值也是在数组定义时直接给定，数组的长度=元素的个数。

**int**[] ages={12,13,56,45};

格式三：

数据类型[] 数组名=new 数据类型[数组的长度];

该格式在定义数组时指定了数组的长度，但是没有直接给定元素的值

注意：

1. 格式一和格式二适用于已知数组的元素的值；也称为数组的静态初始化，格式三也成为数组的动态初始化
2. 数组名：变量名，引用名 符合标示符的定义规范
3. 格式一和格式三支持先声明再初始化，格式二不支持；

数组的操作：

数组给每个元素都定义了一个编号也成为数组的下标或者索引

数组的下标：从0开始，下标的最大值：数组的长度-1

1. 获取数组的元素

格式：数组名[下标]; 下标范围：0-数组的长度-1

数组下标越界异常



1. 修改数组元素的值

数组名[下标]=新的值;

ages[0] = 19;

1. 获取数组的长度

数组名.length;

1. 数组的遍历

方法一：通过普通for循环：注意下标的范围

方法二：增强for循环

格式：for(数据类型 变量名:数组名){

代码块;

}

将数组中的元素一个一个的拿出来依次的赋值给变量，直到数组中的所有元素都遍历完就结束了。一般用于数组或者集合

注意:在使用增强for循环时，不能修改数组的元素值

方法三：

格式：Arrays

导包：import java.util.Arrays

Arrays.toString(数组名);//将数组元素的名拼接成一个字符串[元素1，元素2，元素3，....]

数组常见的应用

1. 获取数组中元素的最大最小值和最大最小值的下标

|  |
| --- |
| //获取数组元素中的最小值  **public** **static** **void** test1(){  **int**[] ages=**new** **int**[]{15,16,13,12};  **int** min=ages[0];  **for**(**int** i=1;i<ages.length;i++){  min=min>ages[i]?ages[i]:min;  }  System.***out***.println(min);  }  //获取数组中元素最大最小值的下标  **public** **static** **void** test11(){  **int**[] ages=**new** **int**[]{15,16,13,12};  **int** index=0;  **int** min=ages[index];  **for**(**int** i=1;i<ages.length;i++){  **if**(ages[index]>ages[i]){  index=i;  }  }  System.***out***.println(index);  } |

1. 实现数组的反转

方法一:创建新的数组

|  |
| --- |
| **public** **static** **void** test2(){  **int**[] ages={12,54,256,16};  **int**[] ages1 =**new** **int**[ages.length];  **for**(**int** i=0;i<ages.length;i++){  ages1[i]=ages[ages.length-1-i];  }  System.***out***.println(Arrays.*toString*(ages1));  } |

方式二：头尾交换

|  |
| --- |
| **public** **static** **void** test3(){  **int**[] ages={15,12,16,14};  **int** start=0;  **int** end=ages.length-1;  **while**(start<end){  **int** temp;  temp=ages[start];  ages[start]=ages[end];  ages[end]=temp;  start++;  end--;  }  System.***out***.println(Arrays.*toString*(ages));  } |

1. 查找数组的元素

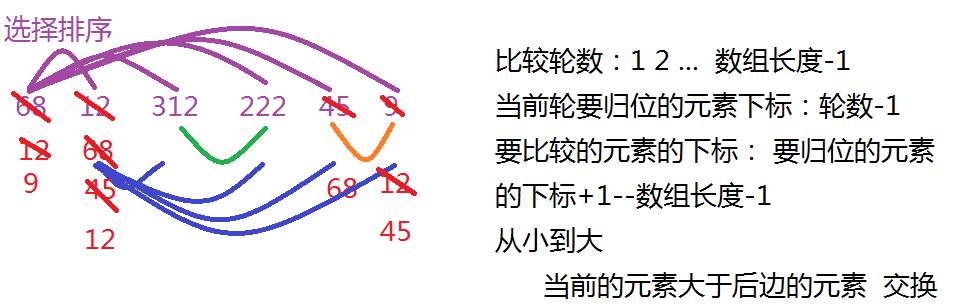
方式一：直接查找

方式二:折半查找，二分查找，有前提必须是有序排列

|  |
| --- |
| //折半查找  **public** **static** **void** test6(){  **int**[] ages={2,5,6,14,15};  **int** num=8;  **int** min=0;  **int** max=ages.length-1;  **int** mid;  **boolean** flag=**false**;  **while**(min<max){  mid=(min+max)/2;  **if**(num==ages[mid]){  System.***out***.println(num+"的下标为"+mid);  flag=**true**;  **break**;  }**else** **if**(num<ages[mid]){  max=mid-1;  }**else**{  min=mid+1;  }  }  **if**(!flag){  System.***out***.println("您输入的数没有找到！");  }  } |

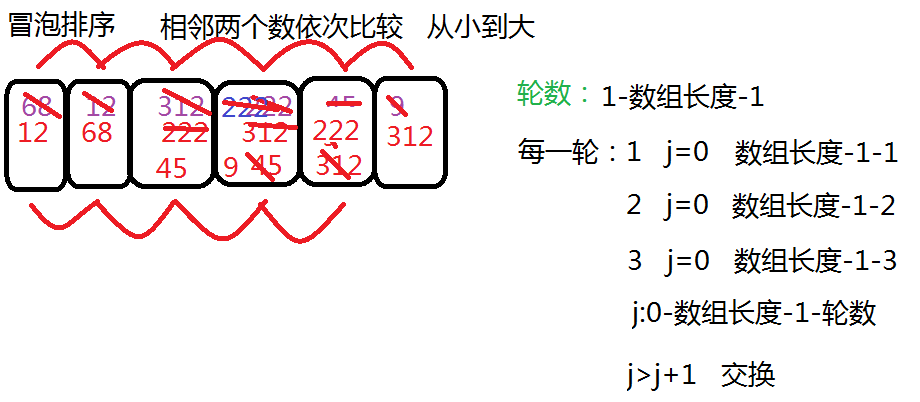
1. 数组排序

方式一：选择排序



|  |
| --- |
| //选择排序  **public** **static** **void** test4(){  **int**[] ages={45,12,36,26,33,34};  **for**(**int** i=1;i<ages.length-1;i++){//轮数  **for**(**int** j=i;j<ages.length-1;j++){  **if**(ages[i-1]>ages[j]){  **int** temp;  temp=ages[i-1];  ages[i-1]=ages[j];  ages[j]=temp;  }  }  }  System.***out***.println(Arrays.*toString*(ages));  } |

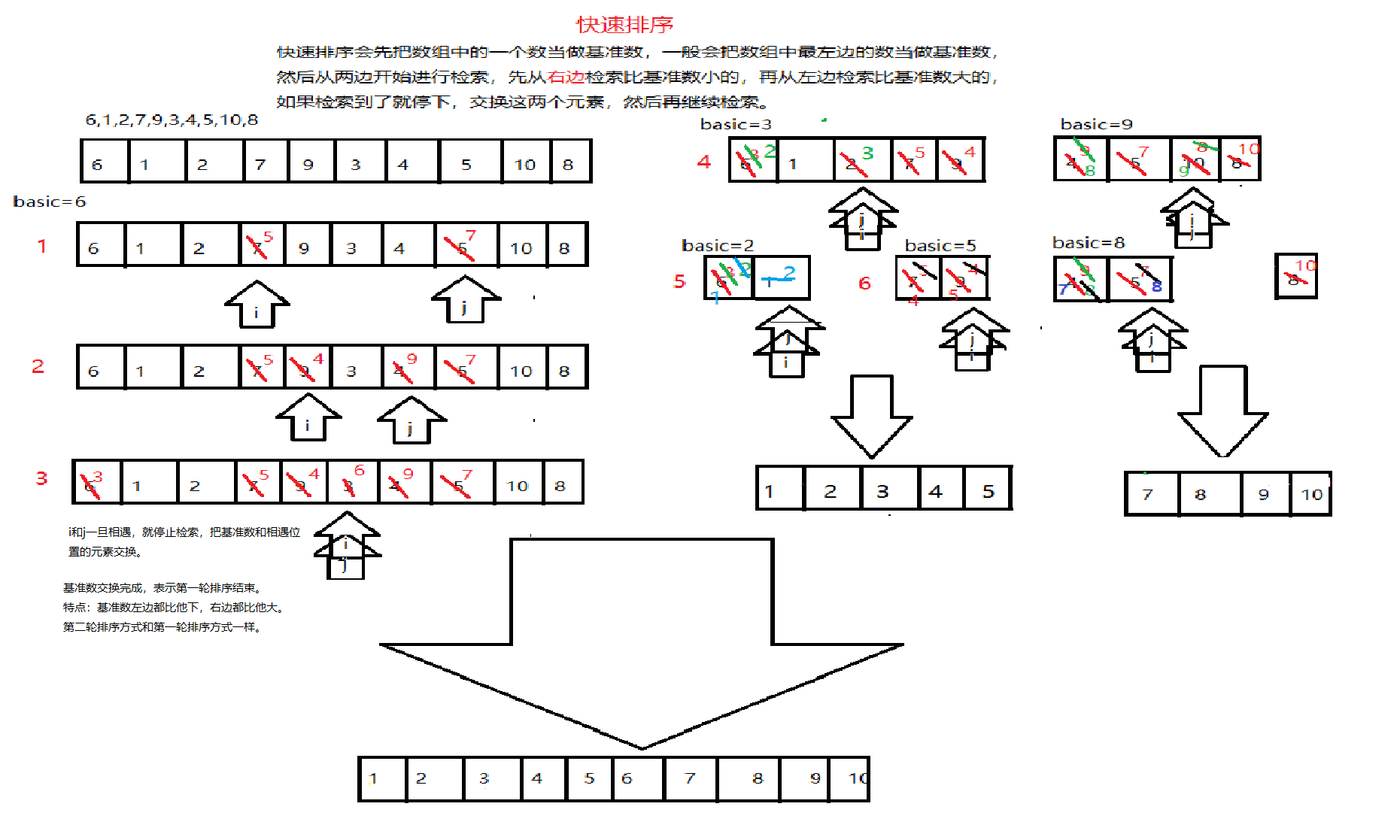
方式二：冒泡排序:相邻的两个数依次相比



|  |
| --- |
| //冒泡排序  **public** **static** **void** test5(){  **int**[] ages={14,13,18,174,12};  **int** len=ages.length-1;  **for**(**int** i=1;i<=len;i++){  **for**(**int** j=0;j<=len-i;j++){  **if**(ages[j]>ages[j+1]){  **int** temp;  temp=ages[j];  ages[j]=ages[j+1];  ages[j+1]=temp;  }  }  }  System.***out***.println(Arrays.*toString*(ages));  } |

方式三：导包；Arrays.sort(数组名);//默认对数组进行升序排序

方式四：快速排序



|  |
| --- |
| **public class** ArraySortDemo {  **public static void** main(String[] args) {  **int**[] arr=**new int**[]{1,3,2,4,34};  *quickSort*(arr,0,arr.**length**-1);  **for** (**int** i : arr) {  System.***out***.println(i);  }  }  *//定义方法，用来进行快速排序* **public static void** quickSort(**int**[] arr,**int** left,**int** right){  *//进行判断，如果左边索引大于右边索引是不合法的，直接使用return结束这个方法。* **if**(left>right){  **return** ;  }  *//定义变量保存基准数* **int** base=arr[left];  *//定义变量i指向向量左边。* **int** i=left;  *//定义变量j执行变量右边* **int** j=right;  *//当i和j不相遇的时候，在循环中进行检索* **while** (i!=j){  *//j向右开始检索，检索到比base小，就停下  //如果检索到比基准数大或者相等就继续检索。* **while** (arr[j]>=base&&j>i){  j--;  }  *//i向左开始检索，检索到比基准数大就停下。  //如果检索到比基准数小或者相等就继续检索。* **while** (arr[i]<=base&&j>i){  i++;  }  *//当代码走到这里，代表i和j都停下了  //当i和j都停下之后，交换i和j* **int** tmp=arr[i];  arr[i]=arr[j];  arr[j]=tmp;  }  *//如果while条件不成立，就代表i和j相遇了  //当i和j相等时交换base和i的位置，并且结束此次的方法，进行下一次* arr[left]=arr[i];  arr[i]=base;  *quickSort*(arr,left,i-1);  *quickSort*(arr,i+1, right);  } } |

1. 数组的复制 扩容

复制：

方法一：使用普通for循环进行复制

方法二：System.arraycopy([Object](mk:@MSITStore:D:\01-开班学习工具\02-课程内软件安装\01-JDK\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html) src,int srcPos,[Object](mk:@MSITStore:D:\01-开班学习工具\02-课程内软件安装\01-JDK\JDK_API_1_6_zh_CN.CHM::/java/lang/Object.html) dest,int destPos,int length)

src:原数组名

srcPos：原数组从哪个下标开始

dest：目标数组名

destPos：目标数从哪个下标开始

length:要复制的元素的个数

方法三:Arrays 导包

copyOf(Object[] original, int newLength)

original：原数组的数组名

newLength：要复制元素的个数

扩容：

Arrays.copyOf(Object[] original,int newLength);

original：原数组的数组名

newLength：数组扩容之后的新的长度

以上方法的底层通过System.arraycopy 实现的

|  |
| --- |
| **If(**newLength>ages.length**){**  **int[] ages1 = new int[**newLength**];**  System.**arraycopy(ages,0,ages1,0,**ages.length**);**  **}else if(**newLength<ages.length**){**  **int[] ages1 = new int[**newLength**];**  System.**arraycopy(ages,0,ages1,0,**newLength**);**  **}** |

注意：数组扩容时底层其实已经实现了一个新的数组。