目录

[7-14(队列queue的用法,反转一个数) 2](#_Toc47380364)

[7-15(二分查找思想) 2](#_Toc47380365)

[7-16(map和stack 的用法) 2](#_Toc47380366)

[7-17(LinkedList与数组的转换) 3](#_Toc47380367)

[7-28(char和int的相互转换) 3](#_Toc47380368)

[8-3(数组复制方法,以及对p++的理解) 4](#_Toc47380369)

# 7-14(队列queue的用法,反转一个数)

队列Queue的用法:

LinkedList实现了Queue接口,所以可以将其视为队列

offer() => 队尾添加元素

poll() => 队头弹出元素

peek() => 返回第一个元素

反转一个数的牛皮思想

while (x != 0) {  
 n = n \* 10 + x % 10;  
 x = x / 10;  
}

# 7-15(二分查找思想)

字符串的用法: subString()

二分查找的思想

while (low < high) {  
 int mid = (high - low + 1) / 2 + low;  
 if (prefix(strs, mid)) {  
 low = mid;  
 } else {  
 high = mid - 1;  
 }  
}

# 7-16(map和stack 的用法)

map的containsKey() 法,判断map中是否有相应的key

栈 stack push() pop()

# 7-17(LinkedList与数组的转换)

数组转LinkedList: LinkedList linklist=new LinkedList(Arrays.asList(array));

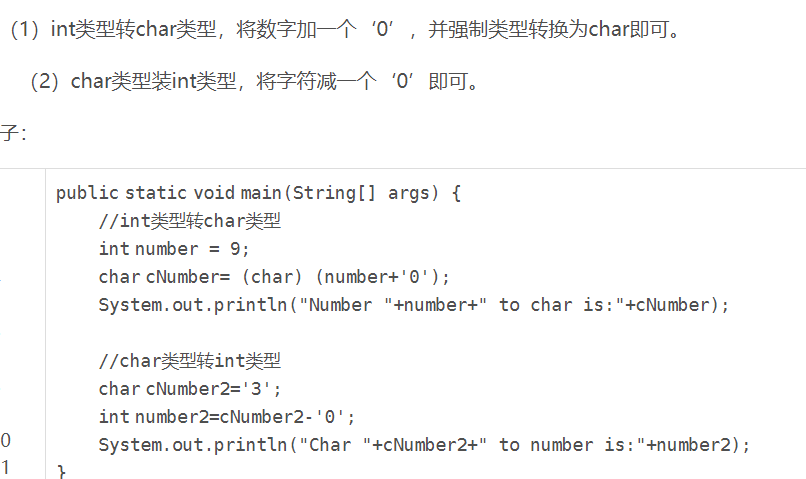
LinkedList转数组:

Integer[] arrray = (Integer[]) list.toArray(new Integer[list.size()]);

# 7-28(char和int的相互转换)

char字符有对应的ASCII码

两个char字符在做运算时会自动转换为对应的ACII码进行运算

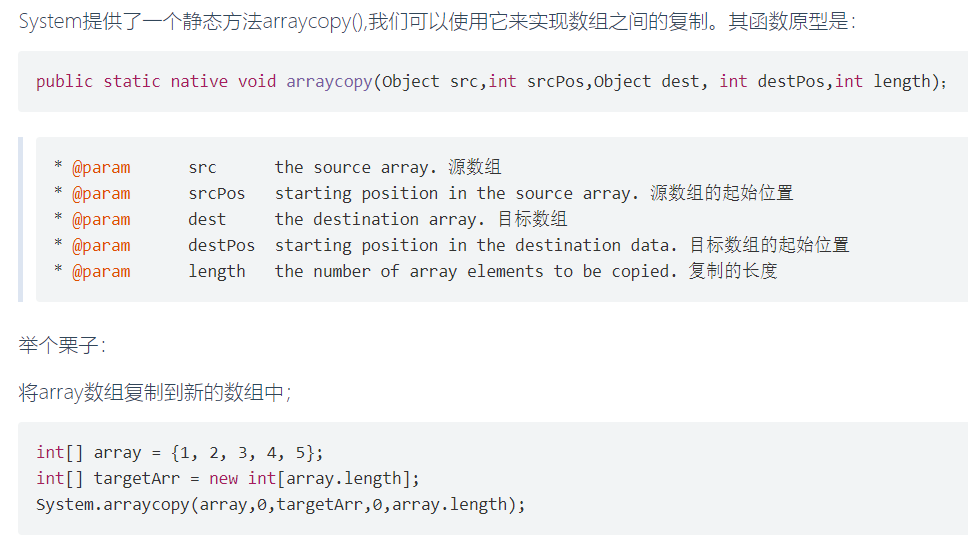


(1)是利用了强转机制, char字符与数值加减,其char字符会转换为数值再计算

(2)是利用了ASCII码

# 8-3(数组复制方法,以及对p++的理解)

System.arraycopy()



mun[p++] = mun[q++]

就相当于

mun[p]= mun[q];

p++;

q++;

# 8-27 判断二叉树是否高度平衡

使用递归计算数的高度,用到了Math.max()方法,

在判断平衡时使用到了Math.abs()来计算绝对值

# 9-9 位运算

