**顺序表基础题：**

实现课本中的顺序表模板类，在模板类中实现如下操作：

构造函数（参数为顺序表的容量）和析构函数

顺序表的输入和输出

引用型操作：Locate,getData,Search,Size，Length，IsFull，IsEmpty

加工型操作：setData，Insert，Remove

在main方法中以一串整数为例测试以上所有的操作。

**编程思路：**

1、int SeqList<T>::Search(T& x)const:

遍历顺序表，如果找到一个元素data[i]与x相等，则返回i+1；如果不能找到则返回0。

2、bool SeqList<T>::Insert(int i,T& x):

如果顺序表满了，则插入失败返回false;

如果要插入的位置i非法，即i<0或者i>last+1，则返回false；

插入时,首先last应该自增1。使用两个变量记录data[i]和data[i+1]后，可以先插入，再移动后面的所有元素。即先将data[i]的值更改为x。之后使用如下循环：

for (int j = i; j <= last; j++) {

rr = data[j + 1]; //将后一位的值记录下来

data[j + 1] = r; //将前一位的值赋给data[j+1]

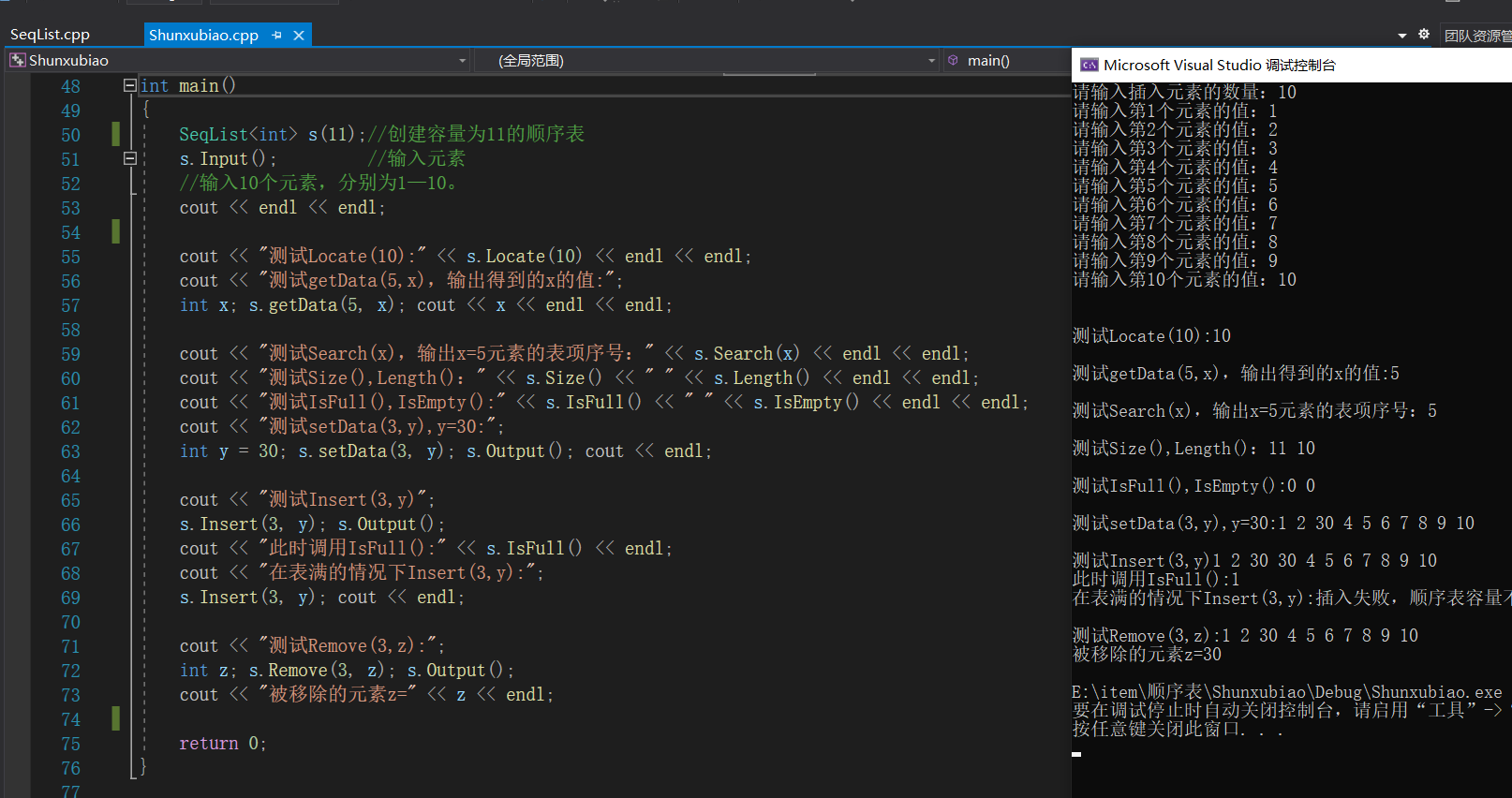
r = rr; } //更新当前值

3、bool SeqList<T>::Remove(int i,T& x):

如果顺序表是空的，则移除失败。如果i的值非法，即i<1或者i>last+1，则移除失败。

首先将需要移除的元素data[i-1]赋值给x，之后last就可以自减。在移除时，从第i个元素开始遍历，到末尾结束，每次只需要将后一项的值赋给data[i-1]。

**运行截图：**



**顺序表提高题：**

利用基础题里构建的顺序表类创建两个有序的整数顺序表对象，实现将两个有序顺序表归并成一个新的有序顺序表并输出该新有序顺序表的结果。（可以调用已定义的顺序表类的方法来实现，并注意如何将两个有序的线性表进行归并的算法）

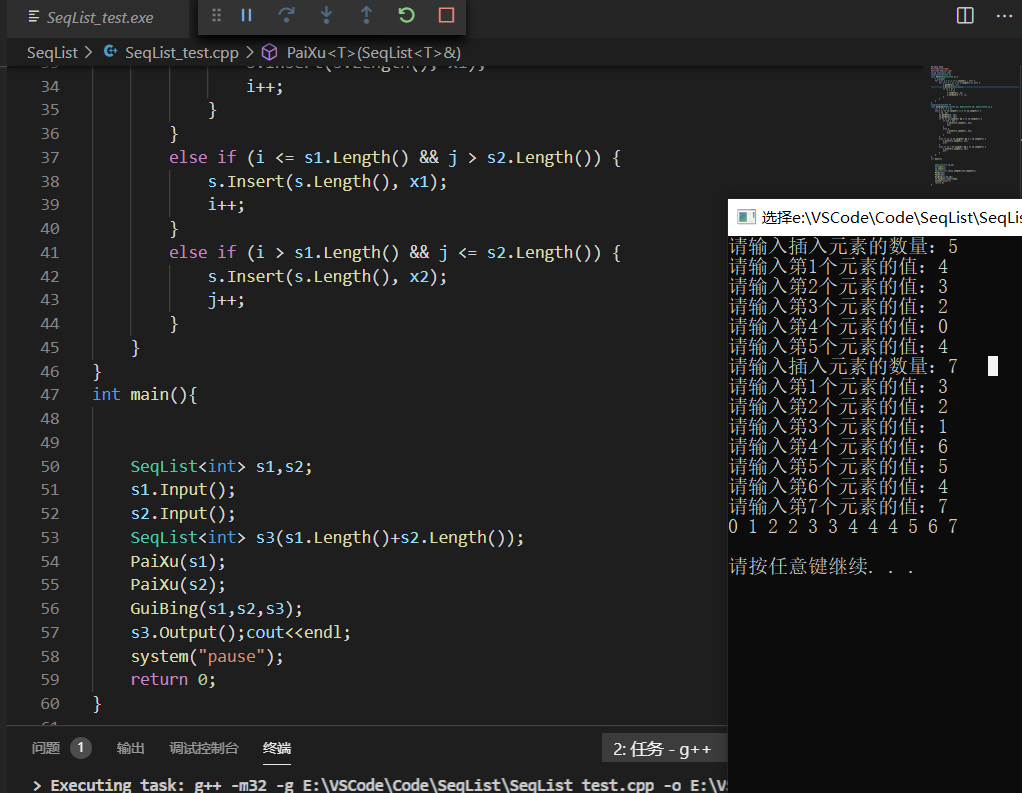
**编程思路：**

首先使用选择排序的方法编写函数PaiXu()，得到两个待归并的有序顺序表。在GuiBing()里，参数传入三个顺序表的引用。 s1、s2是两个有序顺序表，s3是归并后的新的有序表。

使用两个整型数i、j分别代表s1，s2两个表未归并的元素的位置。当两个的值均小于已知表的长度时，比较对应元素的大小，将小的元素插入insert到新表s3中，并将小的元素所在的顺序表的序号i或者j加一，进入下一次循环。

当一个表的所有元素均归并到新表后，只需要将另外一个表的元素依次插入到新表中。

**运行截图：**



**链表基础题**

实现课本中的带附加头结点的单链表模板类，完成如下功能：

定义链表节点的结构体类型

构造函数和析构函数

单链表的输入输出

引用型操作：getData,Locate ,Search,Length，IsEmpty, getHead

加工型操作：setData，Insert，Remove

在main方法中以一串整数为例测试以上所有的操作。

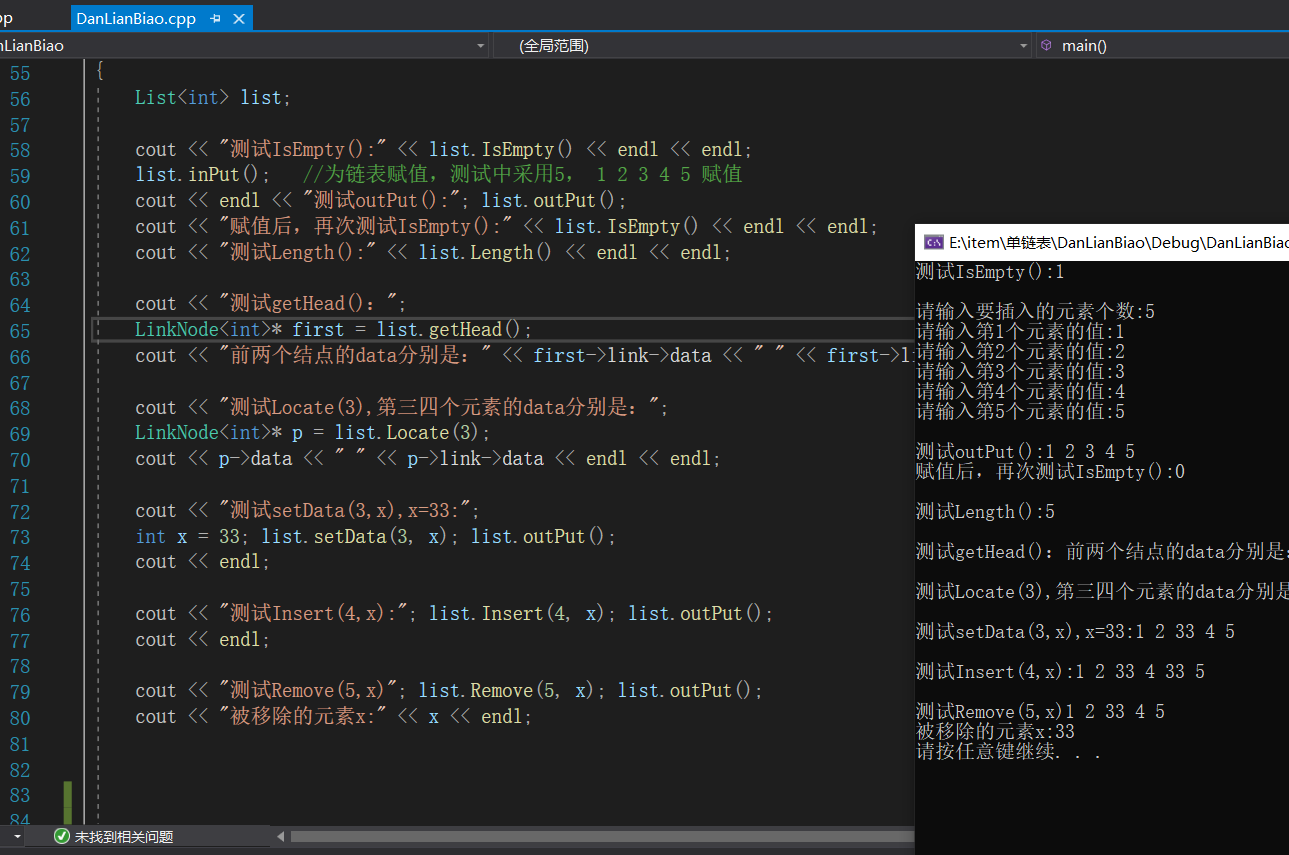
**编程思路:**

LinkNode<T>\* List<T>::Search(T& x)const:使用一个指向结构体的指针指向链表头节点。后通过该指针遍历整个链表，当p->data等于x的时候，返回p指向的地址。

bool List<T>::Insert(int i,T& x):考虑到健壮性，如果表中没有第i个元素，即this->Locate(i)是false的，那么直接返回false；当表中存在第i个元素时，通过locate函数定位到第i个元素的地址，用指针inode指向它。创建一个新的结点，让该结点的link指向inode->link，最后，inode->link要指向新节点newnode。

bool List<T>::Remove(int i,T& x):同样的，考虑到健壮性，如果第i个元素不存在，则返回false；若存在，使用指针inode指向Locate(i)，即指向被删除的结点。将该节点的data值赋给x返回。将第i-1个结点的link指向被删除结点的后继，即inode->link；最后删除inode。

**运行截图：**



**链表提高题**

利用基础题里构建的单链表类创建两个有序的整数链表对象，实现将两个有序链表归并成一个新的有序链表并输出该新有序链表的结果。（可以调用已定义的链表类的方法来实现，并注意如何将两个有序的线性表进行归并的算法）

**编程思路：**

首先利用使用冒泡排序将已知链表从小到大排序。在归并算法GuiBing()中，需要三个引用参数，前两个为待归并的有序链表，最后一个为归并后的有序链表。

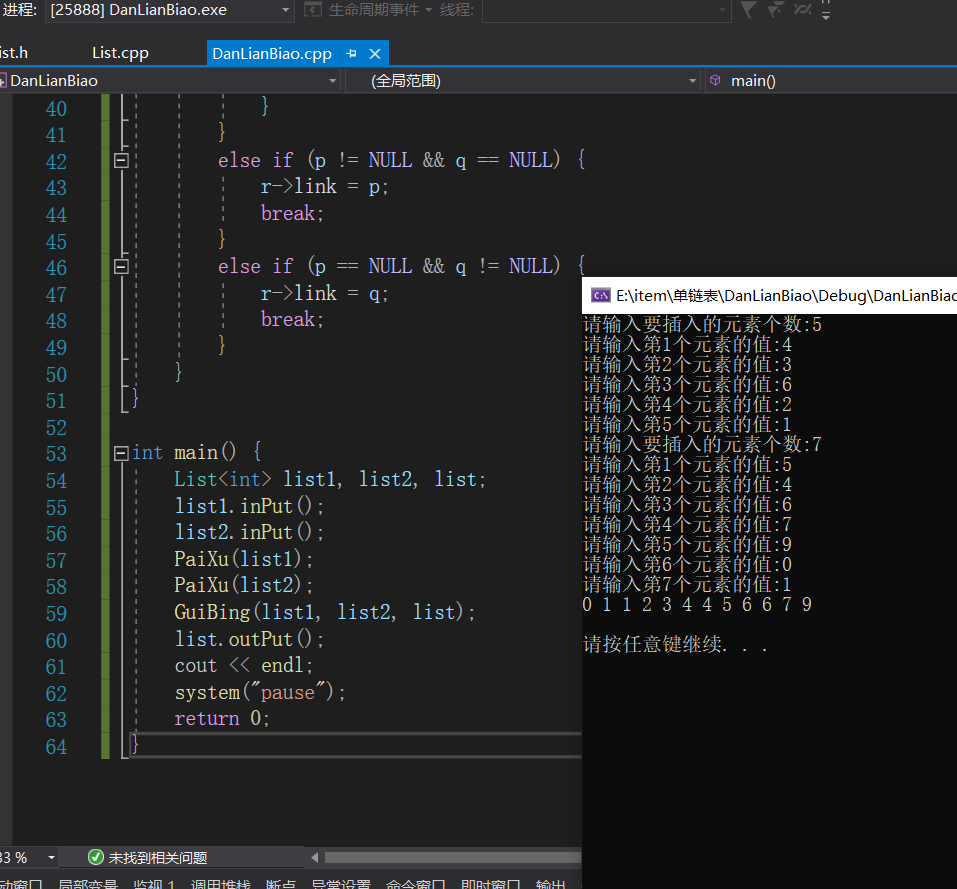
定义两个指针p,q，分别指向list1，list2的头节点的下一个结点，即第一个元素。定义一个指针r指向list的头节点。

通过p、q遍历list1，list2。当两者都不指向NULL时，比较两者的data域大小，并用r指向data小的那个结点（假设为q），之后q需要指向其下一个结点，r也需要指向其下一个结点。

当p、q只有一个不为NULL时，只需要让r->link指向p或q当前结点即可（该节点以及之后所有结点均是有序的）。

在完成本题后，突然想到，既然已经写出了排序算法，那么可以将两个非有序链表合并后，再排序。即直接将list1的最后一个结点的link域指向list2的第二个节点，再对list1链表排序即可。

**运行截图：**



**加强题**

编写一个求解Josephus问题的函数。用整数序列1, 2, 3, ……, n表示顺序围坐在圆桌周围的人。然后使用n = 9, s = 1, m = 5，以及n = 9, s = 1, m = 0，或者n = 9, s = 1, m = 10作为输入数据，检查你的程序的正确性和健壮性。最后分析所完成算法的时间复杂度。定义JosephusCircle类，其中含完成初始化、报数出圈成员函数、输出显示等方法。（可以选做其中之一，存储结构可以用循环链表或数组）

**编程思路:**

对于题干中m、n、s参数的意义理解混乱了。代码中，m表示圆桌周围坐了m个人，n表示报数到n的人出局，s表示从第s个人开始报数。

定义JosephusCircle类。在类的初始化中，使用自定义的单链表类，创建m个结点表示m个参与的人员，结点的data域表示该成员的编号。由于循环链表解决本题较为方便，在初始化中将第m个结点的link域指向第一个结点，即完成了循环链表的构建。

在Run()函数中，使用两个指针，指针p指向开始报数的人，即指向结点s；指针last指向开始报数的前一个人，即结点s-1（方便在\*p出局时删除\*p）。开始通过指针p遍历循环链表，当循环n的整数次时，将该结点删除，形成长度为m-1的循环链表。最终，结束循环的条件是p=p->link，即链表中只剩下一个结点。此时使用类的私有成员指针success指向p。

最后在output()函数中，输出success指针的data域，即成功的人的编号。

**运行截图：**

