作业1

两种存储结构分别放在了不同的命名空间以隔离。

顺序存储结构（SeqList）`namespace SeqLis`

存储结构

```
struct List
{
    Datatype data[MAXN];
    int Length;
    List()
    {
        Length = 0;
    }
};
```

函数说明

`void Insert(DataType x, int P, List &L)` 对于线性表 `L`，在位置 `P` **后**插入元素 `x`。（线性表长度有上限`const int MAXN = 1e5;`，超过此长度会提示错误）
`List InputNewList(int Length)` 输入一个长为 `Length` 新的线性表，并返回它。
`void print(List L)` 输出线性表 `L`。

1. 删除给定元素的算法。
   `void Del(int P, List &L)` 对于线性表 `L`，删除位置为 `P` 的元素。
2. 对于已排好序的线性表，删除所有重复元素的算法。
   `int IsSorted(List &L)` 判断线性表 `L` 的顺序，0:乱序 1:小到大 2:大到小 3:无法确定。
   `void DelRepeatedEleSorted(List &L)` 对于已排好序的线性表 `L`，删除所有重复元素。
3. 线性表“逆置”算法。
   `void Reverse(List &L)` 逆置线性表 `L`。
4. 线性表循环左移/右移 k 位的算法。
   `void Move2Left(int k, List &L)` 线性表 `L` 循环左移 `k` 位。
   `void Move2Right(int k, List &L)` 线性表 `L` 循环右移 `k` 位。
5. 合并两个已排好序的线性表的算法。
   `List MergeListSorted(Lis32t &L1, List &L2` 合并两个已排好序的线性表`L1` 和 `L2`。

链式存储结构（LinkList）`namespace LinkLis`

存储结构

```
struct List
{
    Datatype data;
    List *Next;
    List()
    {
        this->Next = NULL;
    }
};
typedef List * Ele;
```

函数说明

`Ele NewEle()` 新建线性表元素。
`int GetLength(Ele Head)` 返回以 `Head` 为头的线性表的长度。
`void Insert(DataType x, int P, Ele &Head)` 对于以 `Head` 为头的线性表，在位置 `P` **后**插入元素 `x` 。（线性表长度有上限`const int MAXN = 1e5;`，超过此长度会提示错误）
`Ele InputNewList(int Length)` 输入一个长为 `Length` 新的线性表，并返回它。
`void print(Ele Head)` 输出以 `Head` 为头的线性表。

1. 删除给定元素的算法。
   `void Del(int P, Ele &Head)` 对于以 `Head` 为头的线性表，删除位置为 `P` 的元素。
2. 对于已排好序的线性表，删除所有重复元素的算法。
   `int IsSorted(Ele &Head)` 判断以 `Head` 为头的线性表的顺序，0:乱序 1:小到大 2:大到小 3:无法确定。
   `void DelRepeatedEleSorted(Ele &Head)` 对于已排好序的以 `Head` 为头的线性表，删除所有重复元素。
3. 线性表“逆置”算法。
   `void Reverse(Ele &Head)` 逆置以 `Head` 为头的线性表。
4. 线性表循环左移/右移 k 位的算法。
   `void Move2Left(int k, Ele &Head)` 以 `Head` 为头的线性表循环左移 `k` 位。
   `void Move2Right(int k, Ele &Head)` 以 `Head` 为头的线性表循环右移 `k` 位。
5. 合并两个已排好序的线性表的算法。
   `List MergeListSorted(ELe &H1, Ele &H2)` 合并两个已排好序的以 `Head1` `Head2` 为头的线性表。

自测

测试说明

由于本程序未加入`system("pause")`，建议在CMD/Terminal中测试。

为方便测试，提供测试函数（`void Test_SeqLis()`，`void Test_LinkLis()`）、测试数据（文件夹下`Homework1_In.txt`）及测试结果（文件夹下`Homework1_Out.txt`）。

测试函数都定义一个链表 `Test_List` 用作测试，除“合并两个已排好序的线性表的算法”外，其余的函数都对 `Test_List` 做操作。

测试函数中，一开始输入操作次数 `Option_Num` ，之后输入操作代码，具体如下表所示：

操作代码	函数	作用
0	Insert	插入一个元素
1	Del	删除一个元素
2	DelRepeatedEleSorted	对于已排好序的线性表，删除所有重复元素
3	Reverse	线性表逆置
4	Move2Left	线性表循环左移 k 位的算法
5	Move2Right	线性表循环右移 k 位的算法
6	MergeListSorted	合并两个已排好序的线性表
7	InputNewList	输入一个线性表

输入每个操作代码后，按照相应格式输入数据，完成操作。

为方便监视线性表内容，每次操作后都会调用 `print` 输出线性表内容，并用 `SeqListEnd!` 或 `LinkListEnd!` 结尾。

测试过程

```
/********************
前半部分为对SeqList的测试，后半部分为对LinkList的测试。
********************/
17                                  共有17个操作（对顺序存储结构）
0 1 3                               在第3个位置后插入1（非法操作）
Wrong Position!
 SeqListEnd!
0 1 0                               在第0个位置后插入1
1  SeqListEnd!
0 2 1                               在第1个位置后插入2
1 2  SeqListEnd!
0 3 2                               在第2个位置后插入3
1 2 3  SeqListEnd!
1 1                                 删除第1个位置的元素
2 3  SeqListEnd!
1 1                                 删除第1个位置的元素
3  SeqListEnd!
1 1                                 删除第1个位置的元素
 SeqListEnd!
0 1 3                               在第3个位置后插入1（非法操作）
Wrong Position!
 SeqListEnd!
7 4 1 3 2 4                         输入一串长为4，分别为1 3 2 4的线性表
1 3 2 4  SeqListEnd!
2                                   删除重复元素（无法删除，因为不是有序的）
List is Not Sorted
1 3 2 4  SeqListEnd!
7 12 1 2 2 3 3 4 4 5 5 5 6 7        输入一串长为12，分别为1 2 2 3 3 4 4 5 5 5 6 7的线
性表
1 2 2 3 3 4 4 5 5 5 6 7  SeqListEnd!
2                                   删除重复元素
1 2 3 4 5 6 7  SeqListEnd!
3                                   逆置
```

```
7 6 5 4 3 2 1  SeqListEnd!
4 2                                   循环左移2位
5 4 3 2 1 7 6  SeqListEnd!
5 3                                   循环右移3位
1 7 6 5 4 3 2  SeqListEnd!
6                                     合并两个已排好序的线性表（为了方便，测试时需要重新输入两个线性表，但不成功，一个是由小到大，一个是由大到小）
Pleae input 2 sorted list as the following form "Length Data1 Data2 ...":
5 1 3 5 7 9
5 9 8 6 4 2
Two List are Not In The Same Order!
 SeqListEnd!
6                                     合并两个已排好序的线性表
Pleae input 2 sorted list as the following form "Length Data1 Data2 ...":
5 1 3 5 7 9
5 2 4 6 8 9
1 2 3 4 5 6 7 8 9 9  SeqListEnd!




17                                    共有17个操作（对顺序存储结构）
0 1 3                                 在第3个位置后插入1（非法操作）
Wrong Position!
 LinkListEnd!
0 1 0                                 在第0个位置后插入1
1  LinkListEnd!
0 2 1                                 在第1个位置后插入2
1 2  LinkListEnd!
0 3 2                                 在第2个位置后插入3
1 2 3  LinkListEnd!
1 1                                   删除第1个位置的元素
2 3  LinkListEnd!
1 1                                   删除第1个位置的元素
3  LinkListEnd!
1 1                                   删除第1个位置的元素
 LinkListEnd!
0 1 3                                 在第3个位置后插入1（非法操作）
Wrong Position!
 LinkListEnd!
7 4 1 3 2 4                           输入一串长为4，分别为1 3 2 4的线性表
1 3 2 4  LinkListEnd!
2                                     删除重复元素（无法删除，因为不是有序的）
List is Not Sorted
1 3 2 4  LinkListEnd!
7 12 1 2 2 3 3 4 4 5 5 5 6 7          输入一串长为12，分别为1 2 2 3 3 4 4 5 5 5 6 7的线性表
1 2 2 3 3 4 4 5 5 5 6 7  LinkListEnd!
2                                     删除重复元素
1 2 3 4 5 6 7  LinkListEnd!
3                                     逆置
7 6 5 4 3 2 1  LinkListEnd!
4 2                                   循环左移2位
5 4 3 2 1 7 6  LinkListEnd!
5 3                                   循环右移3位
1 7 6 5 4 3 2  LinkListEnd!
6                                     合并两个已排好序的线性表（为了方便，测试时需要重新输入两个线性表，但不成功，一个是由小到大，一个是由大到小）
```

```
Pleae input 2 sorted list as the following form "Length Data1 Data2 ...":
5 1 3 5 7 9
5 9 8 6 4 2
Two List are Not In The Same Order!
 LinkListEnd!
6                                    合并两个已排好序的线性表
Pleae input 2 sorted list as the following form "Length Data1 Data2 ...":
5 1 3 5 7 9
5 2 4 6 8 9
1 2 3 4 5 6 7 8 9 9  LinkListEnd!
```

测试截图

Windows PowerShell

```
PS D:\Dedsecr\Data Structure\Homework1> .\Homework1.exe
17
0 1 3
Wrong Position!
 SeqListEnd!
0 1 0
1  SeqListEnd!
0 2 1
1 2  SeqListEnd!
0 3 2
1 2 3  SeqListEnd!
1 1
2 3  SeqListEnd!
1 1
3  SeqListEnd!
1 1
 SeqListEnd!
0 1 3
Wrong Position!
 SeqListEnd!
7 4 1 3 2 4
1 3 2 4  SeqListEnd!
2
List is Not Sorted
1 3 2 4  SeqListEnd!
7 12 1 2 2 3 3 4 4 5 5 5 6 7
1 2 2 3 3 4 4 5 5 5 6 7  SeqListEnd!
2
1 2 3 4 5 6 7  SeqListEnd!
3
7 6 5 4 3 2 1  SeqListEnd!
4 2
5 4 3 2 1 7 6  SeqListEnd!
5 3
1 7 6 5 4 3 2  SeqListEnd!
6
Pleae input 2 sorted list as the following form "Length Data1 Data2 ...":
5 1 3 5 7 9
5 9 8 6 4 2
Two List are Not In The Same Order!
 SeqListEnd!
6
Pleae input 2 sorted list as the following form "Length Data1 Data2 ...":
5 1 3 5 7 9
5 2 4 6 8 9
1 2 3 4 5 6 7 8 9 9  SeqListEnd!



17
0 1 3
Wrong Position!
 LinkListEnd!
0 1 0
1  LinkListEnd!
0 2 1
1 2  LinkListEnd!
0 3 2
1 2 3  LinkListEnd!
1 1
2 3  LinkListEnd!
1 1
3  LinkListEnd!
1 1
 LinkListEnd!
0 1 3
Wrong Position!
 LinkListEnd!
7 4 1 3 2 4
1 3 2 4  LinkListEnd!
2
List is Not Sorted
1 3 2 4  LinkListEnd!
7 12 1 2 2 3 3 4 4 5 5 5 6 7
1 2 2 3 3 4 4 5 5 5 6 7  LinkListEnd!
2
1 2 3 4 5 6 7  LinkListEnd!
3
7 6 5 4 3 2 1  LinkListEnd!
4 2
5 4 3 2 1 7 6  LinkListEnd!
5 3
1 7 6 5 4 3 2  LinkListEnd!
6
Pleae input 2 sorted list as the following form "Length Data1 Data2 ...":
5 1 3 5 7 9
5 9 8 6 4 2
Two List are Not In The Same Order!
 LinkListEnd!
6
Pleae input 2 sorted list as the following form "Length Data1 Data2 ...":
5 1 3 5 7 9
5 2 4 6 8 9
1 2 3 4 5 6 7 8 9 9  LinkListEnd!
```