**上机报告（二）**

班级：软工11601

姓名：赵惹儿

学号：201603659

班级序号：37

**一．测试代码**：

1. **Test.cpp:**

#include"LinkList.h"

#include<iostream>

using namespace std;

Status fun(ElemType e)

{

cout << e << endl;

return OK;

}

int main()

{

LinkList L;

initList\_L(L);

CreateList\_L(L, 3);

cout << "---逆序输入L的三个元素---" << endl;

ListTraverse(L, fun);

cout << "---输出元素---" << endl;

cout << ListLength\_( L) << endl;

cout << "------元素个数------" << endl;

ElemType e;

GetElem\_L(L, 3, e);

cout << e << endl;

cout << "---输出第三个元素的值---" << endl;

e = rand();

ListInsert\_L(L, 1, e);

cout << e << endl;

cout << "---插入元素的值---" << endl;

ListTraverse(L, fun);

cout << "---插入后输出元素的值---" << endl;

ListDelete\_L(L, 3, e);

cout << e << endl;

cout << "---删除第三个元素的值---" << endl;

ListTraverse(L, fun);

cout << "---删除后输出元素---" << endl;

LinkList La,Lb, Lc;

initList\_L(Lb);//创建空链表Lb

CreateList\_L(Lb, 3);

cout << "------逆序输入Lb的3个元素------" << endl;

MergeList\_L(La, Lb, Lc);//合并La,Lb成Lc

ListTraverse(Lc, fun);

cout << "------输出Lc的元素------" << endl;

cout << ListEmpty\_L(L) << endl;

cout << "------不为空表，返回ERROR，值为0------" << endl;

cout << DestroyList\_L(L) << endl;

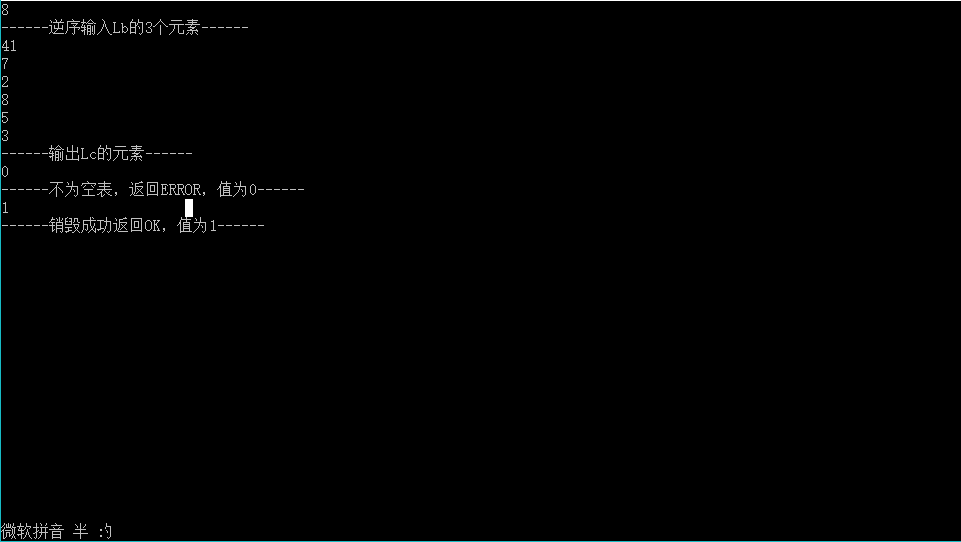
cout << "------销毁成功返回OK，值为1------" << endl;

return 0;

}

**二．测试结果：**





**三．测试过程分析：**

**出现问题**

**1.**

void CreateList\_L(LinkList&L, int n)//算法2.11

{

LinkList p = L;

L = (LinkList)malloc(sizeof(LNode));

L->next = NULL;

int i;

for (i = n; i > 0; --i) {

p= (LinkList)malloc(sizeof(LNode));

scanf\_s ("%d",&p->data);

p->next = L->next; L->next = p;

}

}//

在LinkList.cpp代码中C语言中的知识点，语句为scanf(&p->data)出现错误，随后将其改为scanf("%d",&p->data)，才可编译不出错。

**2**.

LinkList La,Lb, Lc;

initList\_L(Lb);//创建空链表Lb

CreateList\_L(Lb, 3);

cout << "------逆序输入Lb的3个元素------" << endl;

MergeList\_L(La, Lb, Lc);//合并La,Lb成Lc

ListTraverse(Lc, fun);

cout << "------输出Lc的元素------" << endl;

在test.cpp中合并La,Lb成为Lc的过程中程序运行中断，MergeList\_L(L,Lb,Lc)函数没能够正常运行，经查寻错误，有两处原因：（1）没有构造新的空链表Lb，（2）main函数中写了MergeList\_L(La,Lb,Lc)的声明格式：void MergeList\_L(LinkList &La,LinkList&Lb,LinkList&Lc)，这就无法遍历Lc。

**3**.

e = rand();

ListInsert\_L(L, 1, e);

cout << e << endl;

cout << "---插入元素的值---" << endl;

ListTraverse(L, fun);

cout << "---插入后输出元素的值---" << endl;

在调试上述代码中出现：输出插入元素的值，输出e和p->data都无法得到结果或者有错误，后来将rand()赋值给e，再输出e得到了正确结果。

**4**.

同时，在编译时出现的一些简单的语法错误

**四．解释测试**

函数的测试顺序是在一次次的调试过程中最终确定的。首先构造空链表，接下来初始化链表，遍历链表和输出链表长度，这两个函数也可以在插入和删除后进行。需要注意的是归并两个链表要再构造一个新的空链表，否则无法进行归并。最后调用的函数就是销毁和是否是空链表，通过TRUE或者ERROR来反馈结果。

**书中重点算法的自然语言描述：**

**算法2.9**

1. 从头结点遍历寻找第i-1个结点位置
2. 判断i值的合法性，即不能小于1，不能大于表长加1
3. 生成一个由s指向的结点空间，在数据域中存储e
4. 根据插入操作的逻辑定义，修改相关结点的指针域

**算法2.10**

1，从第一个结点开始遍历需找第i-1个结点，赋值给p，也就是待删除结点的前驱  
 2，判断i的合法性  
 3，根据删除的逻辑定义，修改相关结点指针域  
 4，释放资源

**算法2.12**

1．设立三个指针pa,pb,和pc,其中pa,pb分别指向La.和Lb中当前比较插入的结点，pc指向Lc中当前最后一个结点。

2．若pa->data<=pb->data,则将pa所指结点链接到pc所指结点之后，否则将pb所指结点链接到pc所指结点之后。

3．指针的初始状态为：当LA和LB为非空表时，pa和pb分别指向La和Lb表中第一个结点，否则为空。

4．Pc指向空表Lc中的头结点。由于链表的长度为隐含的，则第一个循坏执行的条件是Pa和pb皆非空，当其中一个为空时，说明一个表的元素已归并完，则只需要将另一个表的剩余段链接在pc所指结点后即可。

**五．总结：**  
在归并链表时，不需要另建新表结点空间，只需把原来两个链表结点之间的关系解除（从第一个结点开始依次解除关系），在重新组成一个链表。  
对于算法2.12：  
    用pc把两个表中的已经解除关系(符合条件的)的结点串起来。假设表La为 1 3 5 7，Lb为2 4 6，   
    那么就先按123456串起来，然后在把剩余段串到表尾。有因pc和Lc都指向La的头指针，   
    pa、pb指向的是第一个结点的指针，谁符合条件pc就跟在谁后面，   
。  
注意：书中的算法也可以传两个参数，但是La必须要是可变的(&La)。这样在测试的时候输出La就行了。   
即使传的是三个参数，在测试的时候也可以输出La，Lc和La是共享的。  
对于算法2.12另一中实现的方式：  
    与书中算法的主要不同：  
    是在组合两个表结点之间的关系时，上一个算法是：一条线pc把两个表中的结点串了起来，  
    此算法是：先把Lb表中符合条件结点的next域和La建立关系，在把该结点粘在a的后面，这样使用a就把两个表串起来了，最后输出La就行了。