**实验一：线性表实现**

**学号：B21080526 班级：B210416 姓名:单家俊**

1. **实验目的**

1、掌握线性表的顺序存储和链式存储这两种基本存储结构及其应用场合。

2、掌握顺序表和链式表的各种基本操作算法。

1. **实验任务**

设计一个单链表，实现单链表初始化、求数据元素个数操作、插入、删除、取数据元素、判非空操作。

1. **实验设备**

Visual Studio

1. **实验内容（步骤）**

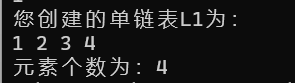
实现了单链表的增删改查，排序等基本运算。

实现了两个单链表求取交集的操作。

1. **实验结果（总结、分析）**

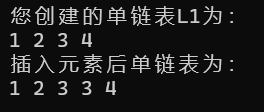
**实验结果分析：**

（1）求数据元素个数

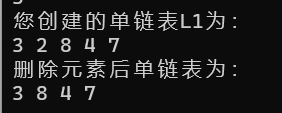


通过键盘输入创建了如图所示的单链表，通过调用函数ListLength\_L(LinkList L)统计了单链表中数据元素的个数，结果正确。

（2）数据元素插入

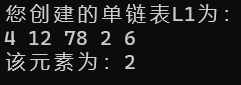
先通过输入创建如图所示的单链表，然后调用函数ListInsert\_L(L1, 3, 3)，该函数的作用是在L1链表第3个结点前插入元素3，结果正确。

（3）数据元素删除



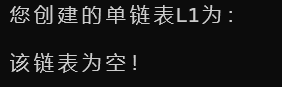
创建如图所示单链表，调用并传入参数ListDelete\_L(L1, 2, e)，实现功能在表L1中删除第二个结点的元素，并将该元素保存在变量e中，结果正确。

（4）取数据元素

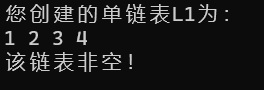


创建如图所示单链表，调用函数并传入参数GetElem\_L(L1, 4, e)，获取表L1中第四个结点的元素，结果正确。

（5）判断非空操作

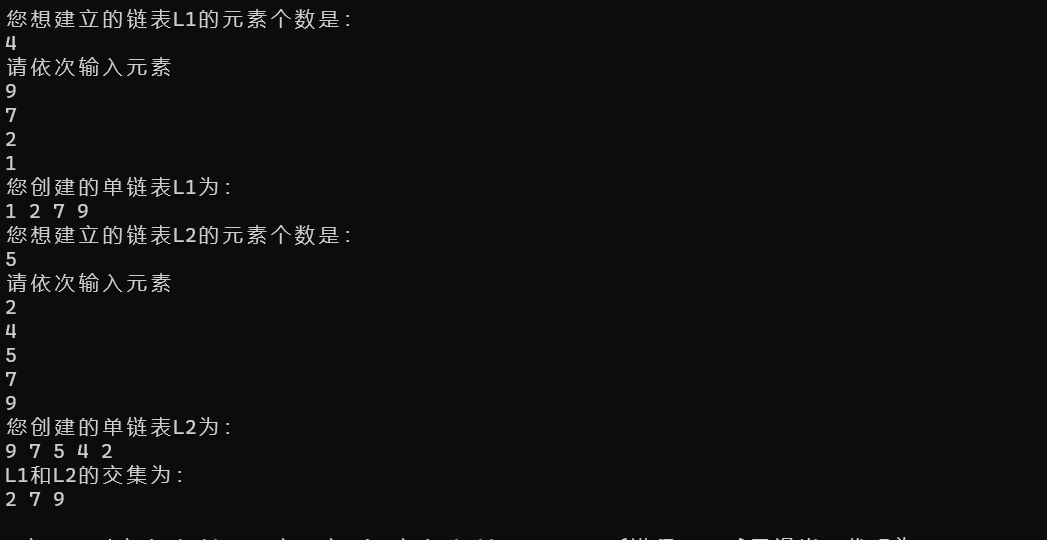


创建一个空链表，判断结果正确。



创建一个非空链表，结果正确。

（6）求两个单链表的交集



测试数据：1 2 7 9 和 9 7 5 4 2，共有元素为2 7 9，结果正确。

**实验过程遇到的问题：**

1、结构体变量与结构体指针内元素的引用符号出现了混淆，结构体变量引用使用的是“.”运算符而结构体指针中的元素的引用使用的是“->”运算符。

2、对于指针的操作有点陌生。

3、对于C++中的cin与cout用法不太熟练。

**体会：**

这是我们上数据结构这门课以来第一次做实验报告，感悟颇深。学习数据结构十一周以来，感觉到数据类型之多，存储方式之丰富，以及各种数据的应用之广泛。其实存储方式倒也没多丰富，不是以数组的方式存储就是以链表的方式存储。只不过链表的方式比较灵活，有顺序表、单链表、单循环链表、双循环链表等等。有链表就有指针，关于指针我真的是有点头疼的，指来指去有时候真的很容易弄混，这也是我在理解数据类型的实现及程序编写时最大的困难。但是，光看书确实很难理解的通透，只有亲身实践过，写过代码，才能对课本上的代码理解的更深刻。也只有在真正写代码的过程中，才能知道自己还有哪些不足。写一次实验报告不容易，我也会好好珍惜每一次机会，查漏补缺，认真学习。

1. **附录（代码）**

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

#define ERROR 0

#define OK 1

typedef int ElemType;

typedef int Status;

typedef struct LNode

{

ElemType data;

struct LNode\* next;

}LNode, \* LinkList;

//初始化

Status InitList\_L(LinkList& L)

{

L = new LNode;

L->next = NULL;

return OK;

}

//求链表长度

int ListLength\_L(LinkList L)

{

int i = 0;

LinkList p;

p = L->next;

while (p)

{

i++;

p = p->next;

}

return i;

}

//判断链表是否为空

int ListEmpty(LinkList L)

{

if (L->next)

return 0;

else

return 1;

}

//查询元素

Status GetElem\_L(LinkList L, int i, ElemType& e)

{

LinkList p;

p = L->next;

int j = 1;

while (p && j < i)

{

p = p->next; ++j;

}

if (!p || j > i)return ERROR;

e = p->data;

return OK;

}

//插入

Status ListInsert\_L(LinkList& L, int i, ElemType e)

{

LinkList p = L;

int j = 0;

while (p && j < i - 1) { p = p->next; ++j; }

if (!p || j > i - 1)return ERROR;

LNode\* s = new LNode;

s->data = e;

s->next = p->next;

p->next = s;

return OK;

}

//删除

Status ListDelete\_L(LinkList& L, int i, ElemType& e) {

LinkList p = L;

int j = 0;

while (p->next && j < i - 1) {

p = p->next; ++j;

}

if (!(p->next) || j > i - 1) return ERROR;

LNode\* q = p->next;

p->next = q->next;

e = q->data;

delete q;

return OK;

}

//前差法创建链表

void CreateList\_F(LinkList& L, int n) {

int i;

L = new LNode;

L->next = NULL;

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

LNode\* p = new LNode;

cin >> p->data;

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

}

}

//输出链表

void ListOutput(LinkList L)

{

LinkList p;

p = L->next;

while (p != NULL) {

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

//求两个集合交集

LinkList Getjiaolist(LinkList LA, LinkList LB, LinkList LC)

{

LinkList p, q, p1;

p = LA->next;

//LA->next = NULL;

p1 = LC;

q = LB->next;

while (p)

{

if (p->data == q->data)

{

LinkList a;

a = new LNode; //申请一个新结点

if (a == NULL)

{

cout << "error";

exit(0);

}

else a->next = NULL;

a->data = p->data;

a->next = NULL;

p1->next = a;

p1 = a;

p = p->next;

q = LB->next;

}

else

{

if (q->next != NULL)q = q->next;

else

{

p = p->next;

q = LB->next;

}

}

}

return LC;

}

int main()

{

LinkList L1;

LinkList L2;

LinkList LC;

InitList\_L(L1);

InitList\_L(L2);

InitList\_L(LC);

int n;

cout << "您想建立的链表L1的元素个数是：" << endl;

cin >> n;

cout << "请依次输入元素" << endl;

CreateList\_F(L1, n);

cout << "您创建的单链表L1为：" << endl;

ListOutput(L1);

// int count = ListLength\_L(L1);

// cout << "元素个数为：" << count;

// ListInsert\_L(L1, 3, 3);

// cout << "插入元素后单链表为：" << endl;

// ListOutput(L1);

// int e ;

// ListDelete\_L(L1, 2, e);

// cout << "删除元素后单链表为：" << endl;

// ListOutput(L1);

// GetElem\_L(L1, 4, e);

// cout << "该元素为：" << e << endl;

// e=ListEmpty(L1);

// if (e == 1){cout << "该链表为空！" << endl; }

// else cout << "该链表非空！" << endl;

cout << "您想建立的链表L2的元素个数是：" << endl;

cin >> n;

cout << "请依次输入元素" << endl;

CreateList\_F(L2, n);

cout << "您创建的单链表L2为：" << endl;

ListOutput(L2);

LC = Getjiaolist(L1, L2,LC);

cout << "L1和L2的交集为：" << endl;

ListOutput(LC);

return 0;

}