实验 2 单链表操作

一 实验目的

1.掌握使用 VC++/DEV-C++上机调试线性表的基本方法；

2.掌握线性表链式存储结构上的基本操作：初始化、插入、删除、查找等运算在链式存储结构上的实现；

3.提交 OJ 系统进行验证。

二 实验要求

1．认真阅读并理解教材上相关操作函数。

2．正确编写本程序并能通过 OJ 系统验证。

3．必须完成：

1） 定义单链表结点结构

2） 建立单链表

3） 查找元素（给出位序，查找元素值）

4） 插入操作

5） 删除操作

6） 主函数（操作函数的调用）

三 实验内容

1、链表的初始化、取值、插入、删除、遍历基本操作

题目描述：链表是数据结构中一种最基本的数据结构，它是用链式存储结构实现的线性表。它较顺序表而言在插入和删除时不必移动其后的元素。现在给你一些整数，然后会频繁地插入和删除其中的某些元素，会在其中某些时候让你查找某个元素或者输出当前链表中所有的元素。

输入：输入数据只有一组，第一行有 n+1 个整数，第一个整数是这行余下的整数数目 n，后面是 n 个整数。这一行整数是用来初始化列表的，并且输入的顺序与列表中的顺序相反，也就是说如果列表中是 1、2、3 那么输入的顺序是 3、2、1。第二行有一个整数 m，代表下面还有 m 行。每行有一个字符串，字符串是“get”，“insert”，“delete”，“show”中的一种。如果是“get”或者“delete”，则其后跟着一个整数 a，代表获得或者删除第 a 个元素；如果是“insert”，则其后跟着两个整数 a 和 e，代表在第 a 个位置前面插入 e；“show”之后没有整数。

输出：如果获取成功，则输出该元素；如果删除成功则输出“delete OK”；如果获取失败

或者删除失败，则输出“get fail”以及“delete fail”。如果插入成功则输出“insert OK”，否则输出“insert fail”。如果是“show”则输出列表中的所有元素，如果列表是空的，则输出“Link list is empty”。注：所有的双引号均不输出。

输入样例：

```
3 3 2 1
21
show
delete 1
show
delete 2
show
delete 1
show
delete 2
insert 2 5
show
insert 1 5
show
insert 1 7
show
insert 2 5
show
insert 3 6
show
insert 1 8
show
get 2
```

输出样例

```
1 2 3
delete OK
```

```
2 3
delete OK
2
delete OK
Link list is empty
delete fail
insert fail
Link list is empty
insert OK
5
insert OK
7 5
insert OK
7 5 5
insert OK
7 5 6 5
insert OK
8 7 5 6 5
7
```

程序框架：

```
#include <iostream>
#include <string.h>
using namespace std;
typedef int ElemType;    // 定义元素的类型为整型

typedef struct LNode{
    ElemType data;        // 定义数据元素
    struct LNode *next; // 定义下一节点的链接（指针）
}LNode, *LinkList;        // 定义节点类型以及链表类型（链表类型实际上是一个节点指针类型）

// L 为带头结点的单链表的头指针，当第 i 个元素存在时，其值赋给 e 并返回 1，否则返回 0
int GetElem_L(LinkList &L, int i, ElemType &e) {
    //======补充代码======

    return 1;
```

```
}

// 在带头结点的单链线性表 L 的第 i 个元素之前插入元素 e
int ListInsert_L(LinkList &L, int i, ElemType e) {
    //======补充代码======

    return 1;
}

// 在带头结点的单链线性表 L 中，删除第 i 个元素，并由 e 返回其值
int ListDelete_L(LinkList &L, int i, ElemType &e) {
    //======补充代码======

    return 1;
}

// 逆位序输入 n 个元素的值，建立带表头结点的单链线性表 L
void CreateList_L(LinkList &L, int n) {
    //======补充代码======

}

// 显示链表中的元素，返回值为链表元素的数目
int ShowList_L(LinkList L){
        int numOfList = 0;          // 记录链表中元素的数目
        LinkList p = L->next;     // 用来遍历链表元素的指针
    while(p){                     // 如果该结点不为空
                    if(numOfList){          // 如果该元素不是列表中的第一个元素（先判断值是否为 0，然后加 1）
            cout<<' ';
        }
        numOfList++;              // 元素的数目加 1
        cout<<p->data;                      // 输出元素
        p = p->next;                // 指针向后移动
    }
    if(numOfList == 0){         // 如果链表中的元素数目为 0，说明是空链表
        return 0;
    }
        else{
        cout<<endl;                         // 注意换行
        return numOfList;     // 返回链表中元素的数目
    }
}

int main(){
    int n,m,a;
    char strInst[30];           // 存储指令:instruction
```

```
    LinkList L;                 // 链表
    ElemType e;                 // 定义节点，用来存储获取的节点或者删除的节点
    cin>>n;
    CreateList_L(L, n);
    cin>>m;
    while(m--){                                                          // 做 m 次循环
        cin>>strInst;                       // 读取指令
        if(strcmp(strInst, "get") == 0){// 如果是需要获取某个元素
            cin>>a;                                                      // 读取元素的位置
            if(GetElem_L(L, a, e) == 1){      // 如果获取元素成功
                cout<<e<<endl;                  // 输出元素的值
            }
                                else{
                cout<<"get fail"<<endl;             // 输出获取元素的出错信息
            }
        }
                    else if(strcmp(strInst, "insert") == 0){// 如果是插入某个元素
            cin>>a>>e;                                                   // 获取待插入的
位置以及待插入的值
            if(ListInsert_L(L, a, e) == 1){             // 如果插入元素成功
                cout<<"insert OK"<<endl;
            }
                                else{
                cout<<"insert fail"<<endl;
            }
        }
                    else if(strcmp(strInst, "delete") == 0){// 如果是删除某个元素
            cin>>a;                                                      // 获得待删除元
素的位置
            if(ListDelete_L(L, a, e) == 1){             // 如果删除成功
                cout<<"delete OK"<<endl;
            }
                                else{
                cout<<"delete fail"<<endl;
            }
        }
                    else if(strcmp(strInst, "show") == 0){          // 如果是显示链表
            if(ShowList_L(L) == 0){                 // 如果链表为空
                cout<<"Link list is empty"<<endl;       // 显示量表为空的信息
            }
        }
    }
    return 0;
}
```

2、 查找链表中的最大值

题目描述：利用单链表表示一个整数序列，通过一趟遍历在单链表中确定值最大的结点。

输入：多组数据，每组数据有两行，第一行为链表的长度 n，第二行为链表的 n 个元素（元素之间用空格分隔）。当 n=0 时输入结束。

输出：对于每组数据分别输出一行，输出每个链表的最大值。

输入样例：

```
5

2 1 3 5 4

6

2 3 10 4 5 1

4

-1 -2 -3 -4

0
```

输出样例：

```
5

10

-1
```

参考代码：

```
#include <iostream>
using namespace std;
typedef struct LNode{
    int data;
    struct LNode *next;
}LNode,*linkList;

void CreateList_R(linkList &L,int n){
     L->next=NULL;
     linkList r=new LNode;
     r=L;
    for(int i=0;i<n;i++) {
         linkList p=new LNode;
         cin>>p->data;
         p->next=NULL;
```

```
            r->next=p;
            r=p;
        }
    }
    //确定单链表中值最大的结点
    int MaxData(linkList L){
        /**************begin*************/

        /**************end*************/
    }
    int main( ){
        int n;
        while(cin>>n)
        {
            if(n==0) break;
            linkList L=new LNode;
            CreateList_R(L,n);
            cout<<MaxData(L)<<endl;
        }
        return 0;
    }
```