**单链表的基本操作**

一、实验目的

1、掌握线性表的链式存储方式

2、掌握单链表的基本操作包括初始化、插入和删除等；

二、实验内容

1、初始化单链表

2、实现任意结点的插入和删除

3、输出单链表中所有结点

三、代码提示

#include<stdio.h>

typedef struct LNode //链表和结点的定义

{

int data; //数据域

struct LNode \*next;//指针域

}LNode,\*LinkList;

/\*采用前插法建立n个元素的单链表\*/

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

L=new LNode;

L->next=NULL; //先建立一个带头结点的空链表

LNode \*p;

for(//补充完整){

p=new LNode; //生成新结点

scanf("%d",&p->data); //输入元素值

//补充完整，将新结点插入头结点之后

}

}

/\*在第i个结点前插入元素e，插入成功返回1，失败返回0\*/

int ListInsert(LinkList &L,int i,int e)

{ LNode \*p=L;

int j=0;

//补充完整，实现p指向第i-1个结点

if(!p||j>i-1)

{

printf("插入位置非法\n");

return 0;

}

LNode \*s=new LNode();//新元素结点

//补充完整，修改新结点的数据域，并建立其与前驱和后继的关系

return 1;

}

/\*删除第i个结点,删除成功返回1，失败返回0\*/

int ListDelete(LinkList &L,int i)

{

LNode \*p=L,\*q; //q暂存被删除的第i个结点的地址

int j=0;

//补充完整，实现p指向第i-1个结点

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

{

printf("删除位置非法\n");

return 0;

}

//补充完整，建立第i-1个结点和第i+1个结点的直接前驱后继关系，释放第i个结点的内存

return 1;

}

/\*输出链表中所有结点的数据值\*/

void ListPrint(LinkList L)

{

//补充完整

}

/\*返回链表中数据值最大的存储位置\*/

LNode\* ListMax(LinkList L)

{

//补充完整

}

void main()

{

LinkList L;

int i,x;

//1.采用前插法创建具有5个元素的单链表

printf("采用前插法输入链表的5个数据：");

ListCreate\_F(L,5);

ListPrint(L);

//2.在第i个结点前插入新数据

printf("依次输入插入位置和数据：");

//补充完整

ListPrint(L);

//3.调用ListDelete函数，删除位置i的节点;

printf("输入删除位置：");

//补充完整

ListPrint(L);

//4.调用ListMax函数，输出最大值；

//补充完整

}