#include<stdio.h>

#include<stdlib.h>

typedef int elemtype;

typedef struct Node //定义一个结构体

{

elemtype data;

struct Node \*next;

}Node;

typedef struct Node \*LinkList;

int InitList(LinkList \*L) //带有头节点的单链表的初始化

{

(\*L) = (LinkList)malloc(sizeof(Node));

if (!L)

{

printf("分配内存失败！\n");

exit(0);

}

(\*L)->next = NULL;

return 0;

}

void CreateListHead(LinkList \*L) //头插法创建一个单链表，n为要插入的元素个数

{

int i, n;

LinkList p;

(\*L) = (LinkList)malloc(sizeof(Node));

(\*L)->next = NULL;

printf("请输入您要插入元素的个数：");

scanf("%d", &n);

printf("请输入你要插入的元素值(用空格隔开)：");

for (i = 0; i < n; i++)

{

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

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

p->next = (\*L)->next;

(\*L)->next = p;

}

}

void CreateListTail(LinkList \*L) //尾插法创建一个单链表，n为要插入的元素个数

{

int i, n;

LinkList p,r;

(\*L) = (LinkList)malloc(sizeof(Node));

r = \*L;

printf("请输入您要插入元素的个数：");

scanf("%d", &n);

printf("请输入你要插入的元素值(用空格隔开)：");

for (i = 0; i < n; i++)

{

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

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

r->next = p;

r = p;

}

r->next = NULL;

}

int LengthList(LinkList \*L) //计算链表长度

{

int length = 0;

LinkList p;

p = (\*L)->next;

while (p)

{

length++;

p = p->next;

}

return length;

}

int GetElem(LinkList L, int i, elemtype \*e) //用e返回L中第i个元素值

{

int j = 1;

LinkList p;

p = L->next;

while (p && j < i)

{

p = p->next;

++j;

}

if (!p || j > i)

{

printf("查询不到该元素！\n");

return 0;

}

\*e = p->data;

return 0;

}

int InsertList(LinkList \*L, int i, elemtype e) //在L中第i个位置插入元素e

{

LinkList p, s;

int j = 1;

p = \*L;

while (p && j < i)

{

p = p->next;

++j;

}

if (!p || j > i)

{

printf("插入元素失败！\n");

return 0;

}

s = (LinkList)malloc(sizeof(Node));

s->data = e;

s->next = p->next;

p->next = s;

return 0;

}

int DeleteList(LinkList \*L, int i, elemtype \*e) //删除L中第i个元素，并用e返回其值

{

LinkList p, q;

int j = 1;

p = \*L;

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

{

p = p->next;

++j;

}

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

{

printf("删除元素失败！\n");

return 0;

}

q = p->next;

p->next = q->next;

\*e = q->data;

free(q);

return 0;

}

int ClearList(LinkList \*L) //单链表的整表删除

{

LinkList p, q;

p = (\*L)->next;

while (p)

{

q = p->next;

free(p);

p = q;

}

(\*L)->next = NULL;

return 0;

}

void ShowList(LinkList \*L) //打印整个链表

{

LinkList p;

p = (\*L)->next;

if (p == NULL)

{

printf("这是一个空链表！\n");

}

printf("单链表");

while (p)

{

printf(" -> %d", p->data);

p = p->next;

}

printf("\n");

}

int main()

{

LinkList L;

InitList(&L);

int k,i;

elemtype m ;

CreateListHead(&L);

ShowList(&L);

printf("单链表的长度为%d\n", LengthList(&L));

CreateListTail(&L);

ShowList(&L);

printf("单链表的长度为%d\n",LengthList(&L));

GetElem(L, 5, &m);

printf("得到的元素值为：%d\n", m);

InsertList(&L, 4, 25);

printf("插入元素后的");

ShowList(&L);

DeleteList(&L, 3, &m);

printf("删除元素后的");

ShowList(&L);

printf("删除的元素值为：%d\n", m);

ClearList(&L);

system("pause");

return 0;

}