#include<stdio.h>

#include<stdlib.h>

//带头结点的单项循环链表

#define ElemType int //未知类型，暂且设为int型，方便后期整体改变

typedef struct Node {

ElemType elem; //数据域

struct LoopNode\* next; //指针域

}LoopNode, \* ptr; //LoopNode为结构体的名字，ptr为指向结构体的指针,sqlistptr指针的指针

typedef ptr\* Sqlistptr;

void List\_Init(Sqlistptr Head); //初始化操作

void List\_Destory(Sqlistptr Head); //销毁线性表

void List\_Clear(Sqlistptr Head); //清空线性表

void List\_Empty(Sqlistptr Head); //判断线性表是否为空

int List\_Length(Sqlistptr Head); //求当前线性表的长度

void Get\_Elem(Sqlistptr Head, int pos, ElemType\* elem); //获得pos位置的值

void List\_Locate(Sqlistptr Head, ElemType elem, int\* pos);

void List\_Insert(Sqlistptr Head, int pos, ElemType elem);//插入

void List\_ConstantlyInsert(Sqlistptr Head, int pos);//连续插入

void List\_Delete(Sqlistptr Head, int pos); //删除

void List\_Print(Sqlistptr Head); //打印

void List\_LoopPrint(Sqlistptr Head); //循环输出

void List\_Command(void);

int main()

{

Sqlistptr Head;

Head = NULL;

int flag = 1;

while (flag) {

ElemType elem;

int len;

int pos;

int choice;

int L;

List\_Command();

scanf\_s("%d", &choice);

while (choice > 13 || choice < 1) {

printf("无效命令，请重新输入：\nCmd:");

scanf\_s("%d", &choice);

}

switch (choice) {

case 1:

List\_Init(&Head);

break;

case 2:

List\_Destory(&Head);

break;

case 3:

List\_Clear(&Head);

printf("线性表清除成功！\n");

break;

case 4:

List\_Empty(&Head);

break;

case 5:

L = List\_Length(&Head);

if (L == -1) {

break;

}

len = L;

printf("当前链表长度为%d\n", len);

break;

case 6:

L = List\_Length(&Head);

if (L == -1) {

break;

}

printf("请输入你要查询的位置:");

scanf\_s("%d", &pos);

while (pos <= 0 || pos > L) {

printf("定位错误,请重新输入位置：");

scanf\_s("%d", &pos);

}

Get\_Elem(&Head, pos, &elem);

printf("%d位置处的值为%d\n", pos, elem);

break;

case 7:

printf("请输出你要查询的值:");

scanf\_s("%d", &elem);

List\_Locate(&Head, elem, &pos);

break;

case 8:

L = List\_Length(&Head);

if (L == -1) {

break;

}

printf("请输入你要插入的位置:");

scanf\_s("%d", &pos);

while (pos <= 0 || pos > L + 1) {

printf("定位错误,请重新输入位置：");

scanf\_s("%d", &pos);

}

printf("请输入要插入的内容:");

scanf\_s("%d", &elem);

List\_Insert(&Head, pos, elem);

break;

case 9:

L = List\_Length(&Head);

if (L == -1) {

break;

}

printf("请输入你要插入的位置:");

scanf\_s("%d", &pos);

while (pos <= 0 || pos > L + 1) {

printf("定位错误,请重新输入位置：");

scanf\_s("%d", &pos);

}

List\_ConstantlyInsert(&Head, pos);

break;

case 10:

L = List\_Length(&Head);

if (L == -1) {

break;

}

printf("请输入你想删除元素的位置:");

scanf\_s("%d", &pos);

while (pos <= 0 || pos > L) {

printf("定位错误,请重新输入位置：");

scanf\_s("%d", &pos);

}

List\_Delete(&Head, pos);

break;

case 11:

List\_Print(&Head);

break;

case 12:

flag = 0;

break;

case 13:

List\_LoopPrint(&Head);

break;

}

}

return 0;

}

void List\_Init(Sqlistptr Head)

{

ptr p, p1;

if (\*Head == NULL);

{

p = (ptr)malloc(sizeof(LoopNode));

if (p == NULL)

{

printf("初始化失败！\n");

return;

}

\*Head = p;

(\*Head)->next = (\*Head);

p1 = (\*Head)->next;

p1->elem = NULL;

}

printf("初始化成功!\n");

}

void List\_Destory(Sqlistptr Head)

{

List\_Clear(Head);

if (\*Head) {

free(\*Head);

\*Head = NULL;

}

printf("链表销毁成功！\n");

}

void List\_Clear(Sqlistptr Head)

{

ptr p;

if (\*Head) {

p = (\*Head)->next;

while (p->next != (\*Head)) {

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

free(p);

p = (\*Head)->next;

}

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

free(p);

p = (\*Head)->next;

p->next = (\*Head);

}

}

void List\_Empty(Sqlistptr Head)

{

if (\*Head == NULL) {

printf("当前链表未初始化或者已被销毁!\n");

return;

}

else {

if ((\*Head)->next == (\*Head)) {

printf("当前线性表为空！\n");

}

else {

printf("当前线性表不空！\n");

}

return;

}

}

int List\_Length(Sqlistptr Head)

{

if (\*Head == NULL) {

printf("当前链表未初始化或者已被销毁!\n");

return -1;

}

else {

int len = 0;

ptr p = (\*Head)->next;

while (p->next != (\*Head)) {

len++;

p = p->next;

}

if (p->elem != NULL) {

len++;

}

return len;

}

}

void Get\_Elem(Sqlistptr Head, int pos, ElemType\* elem)

{

if (\*Head == NULL) {

printf("当前链表未初始化或者已被销毁!\n");

return;

}

else {

ptr p = (\*Head)->next;

int i = 1; //计数器

while (p->next != (\*Head) && i < pos) {

i++;

p = p->next;

}

if (p->next != (\*Head) && i == pos) {

\*elem = p->elem;

}

}

}

void List\_Locate(Sqlistptr Head, ElemType elem, int\* pos)

{

if (\*Head == NULL) {

printf("当前链表未初始化或者已被销毁!\n");

return;

}

else {

ptr p = (\*Head)->next;

int i = 1; //计数器

int flag = 0;

while (p->next != (\*Head)) {

if (p->elem == elem) {

flag = 1;

printf("%d首次出现在位置%d\n", elem, i);

break;

}

i++;

p = p->next;

}

if (flag == 0) {

printf("当前链表中无你要查找的元素！\n");

}

return;

}

}

void List\_Insert(Sqlistptr Head, int pos, ElemType elem)

{

int i = 1;

ptr p = (\*Head)->next;

ptr q = (\*Head);

while (i < pos) {

q = p;

p = p->next;

i++;

}

if (i == pos || i == pos - 1) {

p = (ptr)malloc(sizeof(LoopNode));

if (p) {

p->elem = elem;

p->next = q->next; //顺序不能变

q->next = p;

}

}

printf("插入成功！\n");

return;

}

void List\_ConstantlyInsert(Sqlistptr Head, int pos)

{

ptr p = (\*Head)->next;

ptr q = (\*Head);

int i = 0;

char ch = 0;

ElemType elem;

while (i < pos-1) {

q = p;

p = p->next;

i++;

}

printf("格式如下:3 4 5(按回车结束输入)\n");

while (ch != '\n') {

scanf\_s("%d", &elem);

p = (ptr)malloc(sizeof(LoopNode));

p->elem = elem;

p->next = q->next; //顺序不能变

q->next = p;

q = p;

p = p->next;

ch = getchar();

}

printf("插入成功！\n");

return;

}

void List\_Delete(Sqlistptr Head, int pos)

{

ptr p = (\*Head)->next;

ptr q = (\*Head);

int i = 1;

while (i < pos) {

q = p;

p = p->next;

i++;

}

if (i == pos) {

q->next = p->next;

free(p);

p = NULL;

printf("删除成功！\n");

}

return;

}

void List\_Print(Sqlistptr Head)

{

if ((\*Head)==NULL) {

printf("链表未初始化或者已被销毁！\n");

return;

}

else {

ptr p = (\*Head)->next;

int flag = 1;

printf("当前线性表内容如下：\n");

/\*int i = 0; //检测是否循环

while (p) {

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

//i++;

/\*if (i > 30) {

break;

}

if (p->next != (\*Head)) {

p = p->next;

}

else {

p = (\*Head)->next;

}

if (flag % 10 == 0) {

printf("\n");

}

flag++;

}\*/

while (p->next != (\*Head)) {

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

p = p->next;

if (flag % 10 == 0) {

printf("\n");

}

flag++;

}

if (List\_Length(Head)!=0) {

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

}

printf("\n");

}

}

void List\_LoopPrint(Sqlistptr Head)

{

if ((\*Head) == NULL) {

printf("链表未初始化或者已被销毁！\n");

return;

}

else {

ptr p = (\*Head)->next;

ptr q;

int flag = 1;

int i = 0; //计数器

int n; // 用户选择循环输出多少次

printf("请输入你想循环多少次：\n");

scanf\_s("%d", &n);

printf("当前线性表内容如下：\n");

while (i < n) {

while (p->next != (\*Head)) {

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

p = p->next;

}

printf("%d\n", p->elem);//末尾元素

q = p->next;

p = q->next;

i++;

}

}

}

void List\_Command(void)

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*请输入操作命令\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("操作命令提醒：\n1.初始化链表 2.销毁链表 3.清空链表 4.判断当前链表是否为空 5.求当前链表的长度 6.求链表指定位置处的值 7.求首次出现指定值的位置 8.插入 9.连续插入 10.删除 11.打印 12.退出程序 (区别于不循环单链表)13.循环输出\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Cmd:");

}