#include<stdio.h>

#include<stdlib.h>

#define ElemType int

//带头指针的双向循环链表，头指针里不放元素，仅含前指针和后指针

typedef struct Node {

ElemType elem;

struct Node\* pre, \* next;//pre前指针，next后指针

}Node, \* ptr;

typedef ptr\* Sqlistptr;//指向ptr的指针

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); //正向打印

void List\_ReversePrint(Sqlistptr Head); //反向打印

void List\_Command(void);//命令操作提醒

int main()

{

Sqlistptr Head;

Head = NULL;

int flag = 1;

while (flag) {

ElemType elem;

int len;

int pos;//位置定位

int choice;

List\_Command();

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

while (choice > 11 || 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:

len = List\_Length(&Head);

if (len == -1) {

break;

}

printf("当前链表长度为%d\n", len);

break;

/\*

case 6:

printf("请输入你要查询的位置:");

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

while (pos <= 0 || pos > List\_Length(&Head)) {

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 6:

len = List\_Length(&Head);

if (len == -1) {

break;

}

printf("请输入你要插入的位置:");

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

while (pos <= 0 || pos > len+ 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 7:

len = List\_Length(&Head);

if (len == -1) {

break;

}

printf("请输入你要插入的位置:");

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

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

printf("定位错误,请重新输入位置：");

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

}

List\_ConstantlyInsert(&Head, pos);

break;

case 8:

len = List\_Length(&Head);

if (len == -1) {

break;

}

printf("请输入你想删除元素的位置:");

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

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

printf("定位错误,请重新输入位置：");

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

}

List\_Delete(&Head, pos);

break;

case 9:

List\_Print(&Head);

break;

case 10:

List\_ReversePrint(&Head);

break;

case 11:

flag = 0;

break;

}

}

return 0;

}

void List\_Init(Sqlistptr Head)

{

ptr p;

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

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

//有表头：数据差异化当作表头

if (p == NULL) {

printf("申请空间失败!\n");

}

else {

(\*Head) = p;

(\*Head)->pre = (\*Head)->next = \*Head;//初始时让Head的pre,next指针均指向自己

printf("初始化成功!\n");

}

}

else {

printf("线性表已初始化!\n");

}

return;

}

void List\_Destory(Sqlistptr Head)

{

List\_Clear(Head);

if (\*Head) {

free(\*Head);

(\*Head) = NULL;

}

printf("销毁成功!\n");

}

void List\_Clear(Sqlistptr Head)

{

if (\*Head) {

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

while (p != (\*Head)) {

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

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

free(p);

p = (\*Head)->next;

}

}

}

void List\_Empty(Sqlistptr Head)

{

if (\*Head == NULL) {

printf("当前链表未初始化或已被销毁!\n");

return;

}

else {

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

printf("当前循环链表为空!\n");

}

else {

printf("当前循环链表不空!\n");

}

return;

}

}

int List\_Length(Sqlistptr Head)

{

if (\*Head == NULL) {

printf("当前链表未初始化或已被销毁!\n");

return -1;

}

else {

int len = 0;

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

while (p != (\*Head)) {

len++;

p = p->next;

}

return len;

}

}

/\*

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

{

}

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

{

}

\*/

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

{

ptr p = \*(Head);

int i = 1;

while (i < pos) {

p = p->next;

i++;

}

ptr newNode = (ptr)malloc(sizeof(Node));//创建一个新的节点

if (newNode == NULL) {

printf("申请空间失败！\n");

return;

}

else {

newNode->elem = elem;

newNode->next = p->next;

p->next->pre = newNode;

p->next = newNode;

newNode->pre = p;

printf("插入成功!\n");

}

return;

}

void List\_ConstantlyInsert(Sqlistptr Head, int pos)

{

char ch=0;//用来判断输入是否结束

ptr p = (\*Head);

ptr NewNode;

ElemType elem;

int i = 1;//用来找位置

if (i < pos) {

p = p->next;

i++;

}

printf("请输入数据:(按回车结束，例如:3 4 5)\n");

while (ch != '\n') {

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

NewNode = (ptr)malloc(sizeof(Node));

NewNode->elem = elem;

NewNode->next = p->next;

p->next->pre = NewNode;

p->next = NewNode;

NewNode->pre = p;

p = p->next;

ch = getchar();

}

printf("插入成功!\n");

}

void List\_Delete(Sqlistptr Head, int pos)

{

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

printf("链表未初始化或者已被销毁！\n");

return;

}

else {

ptr q = (\*Head);

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

int i = 1;

while (i < pos) {

q = p;

p = p->next;

i++;

}

p->next->pre = q;

q->next = p->next;

printf("删除成功!\n");

}

}

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; //调节输出格式

int len = List\_Length(Head);

printf("当前线性表内容如下：\n");

while (flag<len) {

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

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

p = p->next;

if (flag % 10 == 0) {

printf("\n");

}

flag++;

}

//printf("%d\*\n", p->elem);

if (len != 0) {

printf("%d\n", p->elem);//输出最后一个元素，最后一个元素后没有空格

}

printf("\n");

}

}

void List\_ReversePrint(Sqlistptr Head)

{

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

printf("链表未初始化或者已被销毁！\n");

return;

}

else {

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

int flag = 1;

int len = List\_Length(Head);

printf("当前线性表内容如下：\n");

while (flag <len) {

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

flag++;

p = p->pre;

if (flag % 10 == 0){

printf("\n");

}

}

if (len != 0) {

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

}

printf("\n");

}

return;

}

void List\_Command(void)

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*请输入双向循环链表的操作命令\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\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.判断链表是否为空 5.求当前链表的长度\n6.插入 7.连续插入(按回车结束输入) 8.删除 9.打印 10.反向打印 11.退出程序\n");

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

printf("Cmd:");

}