#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

#define MAX 50

typedef int Datatype;

typedef struct LinkList {

Datatype\* L;

int length;

int maxlength;

}ListType;

bool InitList(ListType\* list); //初始化并创建线性表

void Insert(ListType\* list, int pos, Datatype data); //插入数据元素

void Append(ListType\* list, Datatype data); //在末尾插入数据元素

void ConstantlyInput(ListType\* list); //连续输入

void Delete(ListType\* list, int pos); //移除某个位置的元素

void Destory(ListType\* list); //销毁线性表

void ClearList(ListType\* list);

void Print(ListType\* list); //输出表中内容

bool IsEmpty(ListType\* list); //判断线性表是否为空

bool IsFull(ListType\* list); //判断线性表是否为满

int Locate(ListType\* list, Datatype a); //在线性表里寻找第一个a的下标

int Prev(ListType\* list, int pos); //求前驱

int Next(ListType\* list, int pos); //求后继

int GetElem(ListType\* list, int pos); //求pos位置的元素

int CurrentLength(ListType\* list); //求当前线性表的长度

void Sort(ListType\* list, int tag); //线性表的排序

//采用快速排序

void QkSort(int\* s, int i, int j);//s为待排序数组，i,j指示了对数组从i到j处的数据进行排序，i<j

int QkPass(int\* s, int i, int j);

void Reverse(int\* s, int length);

void Merge(ListType\* list1, ListType\* list2, int len2); //合并线性表

void Replace(ListType\* list1,ListType\* list2,int len2,int pos); //替换线性表

void Dispose(ListType\* list, ListType\* list1, ListType\* list2,int pos); //拆分线性表

void List\_Command(void);

int main()

{

ListType Mylist,list1,list2;

int choice, data;//用于选择

int flag = 1;

while (flag) {

List\_Command();

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

while (choice > 20 || choice < 1) {

printf("无效命令，请重新输入：\nCmd:");

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

}

switch (choice) {

int pos;

int temp, temp1, temp2, temp3;

int len2;

int data;

case 1:

InitList(&Mylist);

break;

case 2:

printf("请输入你要插入的位置\n");

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

printf("请输入要插入的内容\n");

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

Insert(&Mylist, pos, data);

break;

case 3:

printf("请输入你要追加的元素\n");

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

Append(&Mylist, data);

break;

case 4:

ConstantlyInput(&Mylist);

break;

case 5:

printf("请输入你想删除元素的位置\n");

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

Delete(&Mylist, pos);

break;

case 6:

Destory(&Mylist);

break;

case 7:

Print(&Mylist);

break;

case 8:

if (IsEmpty(&Mylist)) {

printf("线性表为空\n");

}

else {

printf("线性表不为空\n");

}

break;

case 9:

if (IsFull(&Mylist)) {

printf("线性表为满\n");

}

else {

printf("线性表不满\n");

}

break;

case 10:

printf("请输入你想查找的元素");

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

temp = Locate(&Mylist, data);

if (temp != 0) {

printf("你要查找的元素第一次出现在%d位置\n", temp);

}

else {

printf("线性表无该元素\n");

}

break;

case 11:

printf("请输入你要查找哪个位置元素的前驱\n");

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

temp1 = Prev(&Mylist, pos);

if (temp1 != 0) {

printf("%d位置处元素的前驱是%d\n", pos, temp1);

}

break;

case 12:

printf("请输入你要查找哪个位置元素的后继\n");

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

temp2 = Next(&Mylist, pos);

if (temp2 != 0) {

printf("%d位置处元素的后继是%d\n", pos, temp2);

}

break;

case 13:

printf("请输入你要查找的位置\n");

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

temp3 = GetElem(&Mylist, pos);

if (temp3 != 0) {

printf("%d位置处元素为%d\n", pos, temp3);

}

break;

case 14:

printf("当前线性表的长度为%d\n", CurrentLength(&Mylist));

break;

case 15:

printf("请输入排序方式：0(降序),1(升序)\n");

int tag;

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

Sort(&Mylist, tag);

break;

case 16:

InitList(&list2);

printf("请输入要合并的线性表长度\n");

scanf\_s("%d", &len2);

while (len2 > 50 || len2 < 0) {

printf("线性表长度大于50或者长度为负！,请重新输入长度：");

scanf\_s("%d", &len2);

}

printf("请输入要合并线性表的元素:\n");

for (int i = 0; i < len2; i++) {

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

Append(&list2,data);

}

Merge(&Mylist, &list2, len2);

free(list2.L);

break;

case 17:

InitList(&list2);

printf("请输入替换的线性表长度：\n");

scanf\_s("%d", &len2);

while (len2 > MAX || len2 < 0) {

printf("线性表长度大于50或者长度为负！请重新输入长度：\n");

scanf\_s("%d", &len2);

}

printf("请输入要从哪个位置开始替换：\n");

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

while (pos > MAX || pos < 0) {

printf("替换的起始位置有误！请重新输入位置：\n");

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

}

printf("请输入要替换线性表的元素:\n");

for (int i = 0; i < len2; i++) {

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

Append(&list2, data);

}

Replace(&Mylist, &list2, len2,pos);

free(list2.L);

break;

case 18:

InitList(&list1);

InitList(&list2);

//将Mylist中的数据元素拆成奇偶线性表，将奇数存放在list1中，将list2存放在list2中||将Mylist按指定位置拆成两个线性表

printf("请选择拆分方式：奇偶拆分(0),指定位置拆分(1):\n");

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

pos = -1;

if (temp) {

printf("请输入指定位置:\n");

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

while (pos<0 || pos>Mylist.length) {

scanf\_s("输入位置有误，请重新输入:\n%d", &pos);

}

}

Dispose(&Mylist, &list1, &list2,pos);

case 19:

ClearList(&Mylist);

break;

case 20:

flag = 0;

break;

}

}

return 0;

}

bool InitList(ListType\* list)

{

list->L = (Datatype\*)malloc(sizeof(Datatype));

if (!list->L) {

printf("线性表初始化失败！\n");

exit(0);

}

list->length = 0;

list->maxlength = MAX;

printf("线性表初始化成功！\n");

return true;

}

void Insert(ListType\* list, int pos, Datatype data)

{

if (pos < 1 || pos > list->maxlength || list->length == list->maxlength) {

printf("插入失败");

return;

}

for (int i = list->length; i > pos - 1; --i) {

list->L[i] = list->L[i - 1];

}

list->L[pos - 1] = data;

list->length++;

return;

}

void Append(ListType\* list, Datatype data)

{

if (list->length >= list->maxlength) {

printf("追加元素失败！\n");

return;

}

list->L[list->length] = data;

list->length++;

return;

}

void ConstantlyInput(ListType\* list)

{

int n, x;

printf("请输入要连续输入的个数：\n");

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

if (n > list->maxlength) {

printf("输入元素过多,当前线性表最大长度为%d\n", list->maxlength);

}

else {

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

printf("请输入第%d个数据元素：", (i + 1));

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

list->L[list->length] = x;

list->length++;

}

}

}

void Delete(ListType\* list, int pos)

{

if (pos < 1 || pos > list->length) {

printf("位置错误!\n");

return;

}

for (int i = pos; i < list->length; ++i) {

list->L[i - 1] = list->L[i];

}

list->length--;

}

void Destory(ListType\* list)

{

free(list->L);

list->length = 0;

printf("线性表销毁成功\n");

return;

}

void ClearList(ListType\* list)

{

list->length = 0;

return;

}

void Print(ListType\* list)

{

if (list == NULL) {

printf("不存在线性表\n");

}

else {

if (list->length == 0) {

printf("当前线性表为空！\n");

}

else {

for (int i = 0; i < list->length; i++) {

printf("第%d项：%d\n", i + 1, list->L[i]);

}

}

}

return;

}

bool IsEmpty(ListType\* list)

{

if (list->length == 0) {

return true;

}

else {

return false;

}

}

bool IsFull(ListType\* list)

{

if (list->length == list->maxlength) {

return true;

}

else {

return false;

}

}

int Locate(ListType\* list, Datatype a)

{

int i = 0;

while (i < list->length && list->L[i] != a) {

i++;

}

if (i < list->length) {

return i + 1;

}

else {

return 0;

}

}

int Prev(ListType\* list, int pos)

{

if (pos < 1 || pos >list->length) {

printf("location error\n");

}

else if (pos == 1) {

printf("第一个位置的元素没有前驱\n");

}

else {

return list->L[pos - 1 - 1];

}

return 0;

}

int Next(ListType\* list, int pos)

{

if (pos <1 || pos > list->length) {

printf("location error\n");

}

else if (pos == list->L[list->length - 1]) {

printf("最后一个位置的元素没有后继\n");

}

else {

return list->L[pos - 1 + 1];

}

return 0;

}

int GetElem(ListType\* list, int pos)

{

int n;

if (pos<1 || pos>list->length) {

printf("Location error\n");

n = 0;

}

else {

n = list->L[pos - 1];

}

return n;

}

int CurrentLength(ListType\* list)

{

return list->length;

}

void QkSort(int\* s, int i, int j)

{

if (i < j) {

int k = QkPass(s, i, j);

//对前部分排序

QkSort(s, i, k - 1);

//对后部分排序

QkSort(s, k + 1, j);

}

}

int QkPass(int\* s, int i, int j)

{

int key = s[i];

while (i < j) {

while (i < j && s[j] >= key) {

j--;

}

s[i] = s[j];

while (i < j && s[i] >= key)

j--;

while (i < j && s[i] <= key)

i++;

s[j] = s[i];

}

s[i] = key;

return i;

}

void Reverse(int\* L, int length)

{

int temp;

for (int i = 0; i < (length + 1) / 2; i++)

{

temp = L[i];

L[i] = L[length - i - 1];

L[length - i - 1] = temp;

}

return;

}

void Sort(ListType\* list, int tag)

{

QkSort(list->L, 0, list->length - 1);

if (tag == 0) {

Reverse(list->L, list->length);

}

return;

}

void Merge(ListType\* list1, ListType\* list2, int len2)

{

int flag = 1;

if (list1->length + len2 > list1->maxlength) {

printf("两个线性表长度之和超过原本线性表的最大长度,是否选择继续合并：yes(1),no(0)");

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

}

if (flag) {

int i = 0;

while (list1->length < list1->maxlength&&len2>0) {

list1->L[list1->length] = list2->L[i++];

list1->length++;

len2--;

}

}

else {

printf("你选择放弃合并！\n");

}

return;

}

void Replace(ListType\* list1,ListType\* list2,int len2,int pos)

{

int flag = 1;

if (pos + len2 > list1->maxlength) {

printf("替换后长度溢出,是否选择继续替换：yes(1),no(0)");

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

}

if (flag) {

int i = 0;

while (len2>0&&pos<=list1->length) {

list1->L[pos] = list2->L[i++];

len2--;

pos++;

}

while (len2 > 0) {

list1->L[list1->length] = list2->L[i++];

list1->length++;

len2--;

}

}

else {

printf("你选择放弃替换！\n");

}

return;

}

void Dispose(ListType\* list,ListType\* list1,ListType\* list2,int pos)

{

int t1 = 0;

int t2 = 0;

if (pos == -1) {

for (int i = 0; i < list->length; i++)

{

if (list->L[i] % 2 == 1) {

list1->L[t1] = list->L[i];

list1->length++;

t1++;

}

else {

list2->L[t2] = list->L[i];

list2->length++;

t2++;

}

}

printf("拆分后奇数线性表为：\n");

Print(list1);

printf("拆分后偶数线性表为：\n");

Print(list2);

}

else {

for (int i = 0; i < list->length; i++)

{

if (i < pos) {

list1->L[t1] = list->L[i];

list1->length++;

t1++;

}

else {

list2->L[t2] = list->L[i];

list2->length++;

t2++;

}

}

printf("拆分后pos之前的线性表为：\n");

Print(list1);

printf("拆分后pos之后的线性表为：\n");

Print(list2);

}

return;

}

void List\_Command(void)

{

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

printf("操作命令提醒：\n1.Initial 2.Insert 3.Append 4.Constantlyinput 5.Delete 6.Destory\n 7.Print 8.IsEmpty 9.IsFull 10.Locate 11.Prev(前驱) 12.Next(后继)\n 13.GetElem 14.CurrentLength 15.Sort 16.Merge 17.Replace 18.Dispose 19.Clear 20.Exit\n");

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

printf("Cmd:");

}