#include<stdio.h>

#include<stdlib.h>

typedef int datatype;

typedef struct link\_node {

int data\_info;

struct link\_node\* next;

}node;

node\* init() {

return NULL;

}

void display(node\*head) {

node\* p;

p = head;

if (!p)printf("当前链表为空\n");

else {

printf("链表内元素为：\n");

while (p){

printf("%5d", p->data\_info);

p = p->next;

}

}

}

//查询数据

node\* find(node\* head, int i) {

int j = 1;

node\* p = head;

if (i<1) return NULL;

while (p&&i!=j) {

p = p->next;

j++;

}

return p;

}

//数据输入

node\* beg\_indata(node\*head,int i) {//前插法

node\* p, \* pre;

int x;

pre = find(head, i);

printf("请输入数据：");

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

p = (node\*)malloc(sizeof(node));

p->data\_info = x;

if (i == 0) {

p->next = head;

head = p;

}

else {

p->next = pre->next;

pre->next = p;

}

return head;

}

node\* end\_indata(node\* head, int i) {//后插法

node\* p, \* pre;

int x;

pre = head;

pre = find(head, i);

printf("请输入数据：");

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

p = (node\*)malloc(sizeof(node));

p->data\_info = x;

if (i == 0) {

p->next = head;

head = p;

}

else {

p->next = pre->next;

pre->next = p;

}

return head;

}

node\* indata(node\*head)

{

printf("请选择插入方式\n1)前插\n2)后插\n");

int m, i;

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

printf("请输入插入位置");

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

if (m == 1)head=beg\_indata(head, i);

else if (m == 2)head=end\_indata(head, i);

else printf("请输入正确的插入方式对应数字\n");

return head;

}

//数据删除

node\* dele(node\* head, datatype x) {

node\* p, \* pre;

p = head;

pre = NULL;

if (head == NULL) {

printf("当前链表为空");

return NULL;

}

while (p->data\_info!=x && p) {

pre = p;

p = p->next;

}

if (p) {

if (!pre)head = head->next;

else pre->next = p->next;

free(p);

}

return head;

}

//题目3.2

int many(node\*head) {

node\* p = head;

int i = 0;

if (!head)i = 0;

else {

while (p) {

p = p->next;

i++;

}

}

return i;

}

//题目3.5

bool Orderly(node\* head) {

node\* p, \* pre;

pre = head;

p = pre->next;

if (p->data\_info >= pre->data\_info) {

while (p) {

if (p->data\_info < pre->data\_info)

return false;

pre = p;

p = p->next;

}

}

else if (p->data\_info < pre->data\_info) {

while (p) {

if (p->data\_info > pre->data\_info)

return false;

pre = p;

p = p->next;

}

}

return true;

}

//习题3.6

node\*ni(node\* head) {

node\* p, \* pro;

p = head->next;

head->next = NULL;

while (p)

{

pro = p;

p = p->next;

pro->next = head;

head = pro;

}

return head;

}

void show() {//选择界面

printf("0)退出\n1)插入数据\n2)打印数据\n3)查询节点位置\n4)删除指定数据\n5)输出节点数目\n6)各个节点值是否单调有序\n7)倒序\n");

}

int main() {

node\* head;

head = NULL;

int m;//储存输入数据

int n;//临时储存数据

int max, min;//专门用于第七题

while (1) {

show();

printf("请输入操作数字：");

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

if (m == 0)break;

else if (m == 1) { head = indata(head); }//在结尾输入数据

else if (m == 2) { display(head); }//打印数据

else if (m == 3) { printf("输入要查询的节点位数："); scanf\_s("%d", &n); printf("地址为：%d\n", find(head, n)); }

else if (m == 4) { printf("输入要删除的数据："); scanf\_s("%d", &n); head=dele(head, n); }

else if (m == 5) { printf("节点数目为：%d", many(head)); }

else if (m == 6) { if (Orderly(head))printf("各个节点值单调有序\n"); else printf("各个节点值不单调有序\n"); }

else if (m == 7) { head = ni(head); }

else printf("输入错误，请重新输入\n");

printf("\n");

system("pause");

system("cls");

}

return 0;

}









