教材部分：

6.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 13

typedef struct YSF //链表实现约瑟夫问题

{

int num;

struct YSF \*next;

}COU;

COU \*create() //创建链表

{

COU \*pre, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU)); //头结点申请空间

pre = head; //让结点与头结点处于同一位置

for(int i=1; i<=N; i++)

{

tem = (COU\*)malloc(sizeof(COU)); //临时变量申请空间

tem->num = i; //赋值

pre->next = tem; //指向

pre = tem; //后移

}

pre->next = tem;

return head;

}

void quit(COU \*head)

{

int tol = N;

COU \*p = head;

COU \*q;

while(tol > 1) //当前剩余人数

{

int cnt = 1;

while(cnt < 3) //报数移动

{

++cnt;

q = p;

p = p->next;

}

q->next = p->next; //删除报到3的点

free(p); //释放空间

p = q->next; //新起点

tol--; //人数-1

}

printf("剩下的人是:\n");

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

return;

}

int main()

{

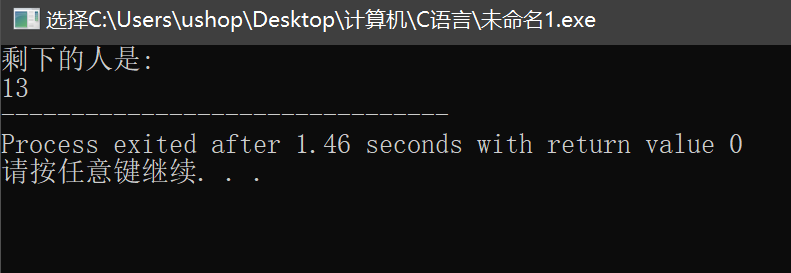
COU \*head;

head = create(); //创建链表

quit(head); //约瑟夫

return 0;

}



7.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 5

struct student

{

long num;

float score;

struct student \*next;

};

struct student \*del(struct student \*head, long num)

{

struct student \*p1, \*p2;

if(head == NULL) //如果链表为空

{

printf("\nlist null!\n");

return head;

}

p1 = head;

while(p1->num != num && p1->next != NULL) //找数

{

p2 = p1;

p1 = p1->next;

}

if(p1->next == NULL) printf("找不到该数！\n"); //找不到的情况

else {

// printf("\ndelete: %d\n",p1->num);

p2->next = p1->next; //删除并释放空间

free(p1);

}

return head;

}

struct student \*create() //创建

{

struct student \*end, \*tem, \*head ;

head = (struct student\*)malloc(sizeof(struct student));

end = head;

printf("输入数列\n");

for(int i=1; i<=N; i++)

{

tem = (struct student\*)malloc(sizeof(struct student));

scanf("%ld",&tem->num);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

void print(struct student \*p) //输出

{

puts("余下的数是：");

while(p!= NULL)

{

p = p->next;

printf("%ld ",p->num);

}

return;

}

int main()

{

long del\_num;

puts("要删除的数字是：");

scanf("%ld", &del\_num);

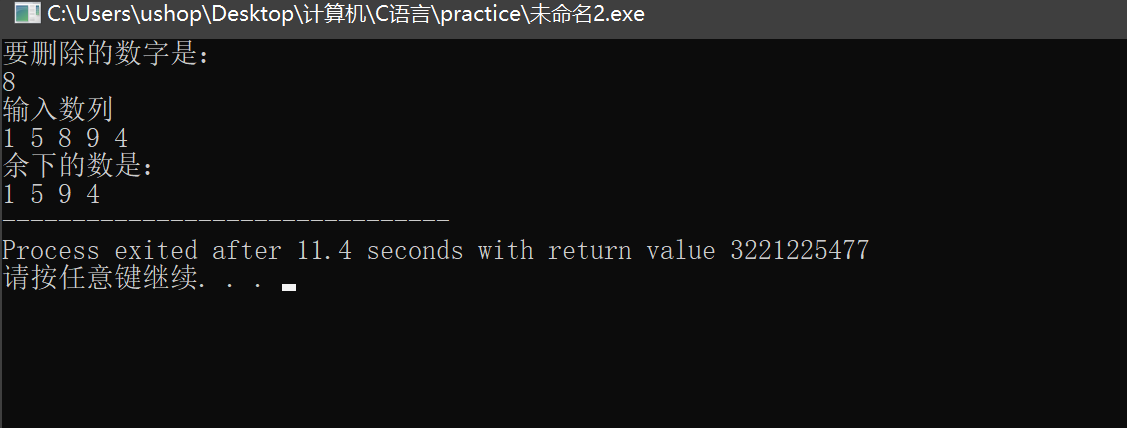
struct student \*head;

head = create();

head = del(head, del\_num);

print(head);

return 0;

} 

8.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 5

typedef struct STU

{

int num;

struct STU \*next;

}COU;

COU \*create() //创建

{

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

printf("输入升序数列\n");

for(int i=1; i<=N; i++)

{

tem = (COU\*)malloc(sizeof(COU));

scanf("%d",&tem->num);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

void insert(COU \*head, int x) //插入

{

COU \*p1 = head->next;

COU \*p2;

COU \*in = (COU\*)malloc(sizeof(COU));

in->num = x;

if(x < p1->num) //是最小数的情况

{

in->next = head->next;

head->next = in;

}

else

{

while(p1->num < x && p1!= NULL) //找插入位置

{

p2 = p1;

p1 = p1->next;

}

if(p1==NULL) p2->next = in; //尾结点的情况

else

{

in->next = p2->next; //通常情况的插入

p2->next = in;

}

}

return;

}

void print(COU \*p) //打印输出

{

while(p!= NULL)

{

p = p->next;

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

}

return;

}

int main()

{

COU \*head;

int insert\_num;

head = create();

printf("插入：\n");

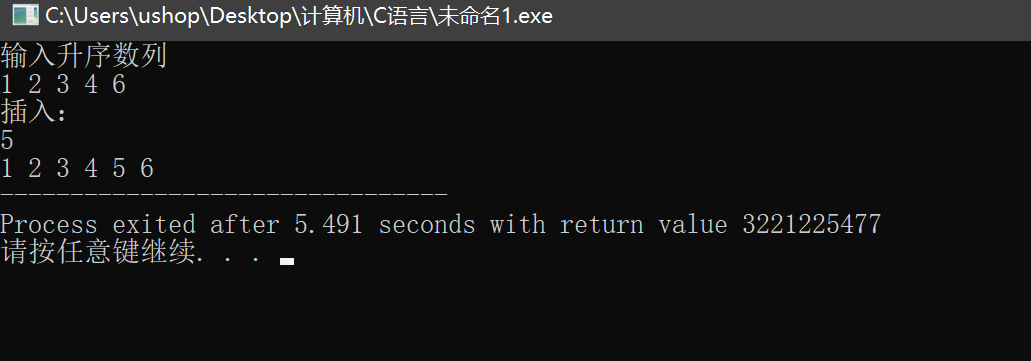
scanf("%d",&insert\_num);

insert(head,insert\_num);

print(head);

return 0;

}



9.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 5

typedef struct STU

{

int num;

struct STU \*next;

}COU;

COU \*create() //创建

{

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

printf("输入升序数列\n");

for(int i=1; i<=N; i++)

{

tem = (COU\*)malloc(sizeof(COU));

scanf("%d",&tem->num);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

void insert(COU \*head, int x) //插入x

{

COU \*p1 = head->next;

COU \*p2;

COU \*in = (COU\*)malloc(sizeof(COU));

in->num = x;

if(x < p1->num)

{

in->next = head->next;

head->next = in;

}

else

{

while(p1->num < x && p1!= NULL)

{

p2 = p1;

p1 = p1->next;

}

if(p1==NULL) p2->next = in;

else

{

in->next = p2->next;

p2->next = in;

}

}

return;

}

COU \*del(COU \*head, int num) //删除num

{

COU \*p1, \*p2;

if(head == NULL)

{

printf("\nlist null!\n");

return head;

}

p1 = head->next;

while(p1->num != num && p1 != NULL)

{

p2 = p1;

p1 = p1->next;

}

if(p1 == NULL) printf("找不到该数！\n");

else {

p2->next = p1->next;

free(p1);

}

return head;

}

void print(COU \*p) //打印输出

{

printf("结果为：\n");

while(p!= NULL)

{

p = p->next;

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

}

return;

}

int main()

{

COU \*head;

int insert\_num, delete\_num;

head = create();

printf("插入：\n");

scanf("%d",&insert\_num);

insert(head,insert\_num);

printf("删除：\n");

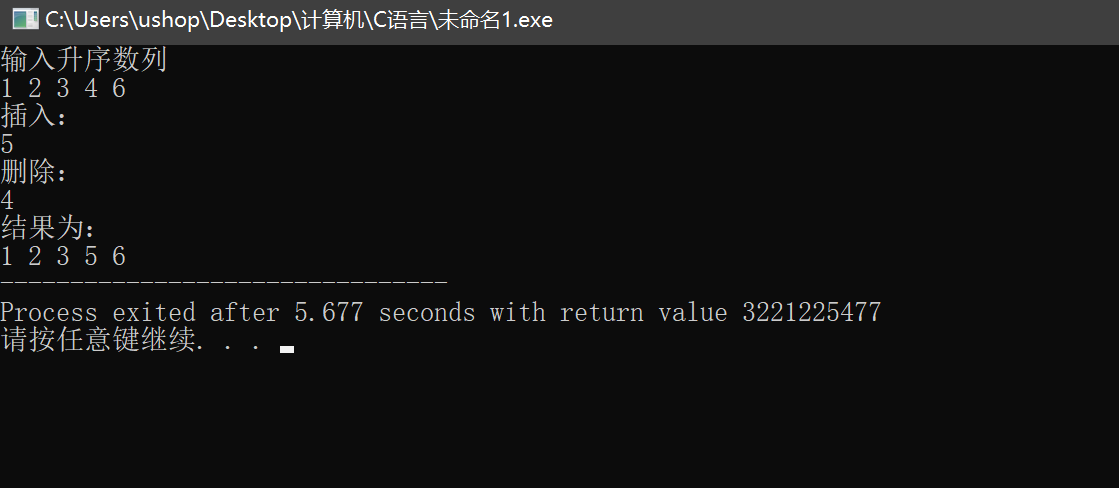
scanf("%d",&delete\_num);

del(head,delete\_num);

print(head);

return 0;

}



10.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct STU

{

char xh[20]; //学号

double score; //分数

struct STU \*next;

}COU;

COU \*create() //创建链表

{

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

while(1)

{

tem = (COU\*)malloc(sizeof(COU));

scanf("%s %lf",tem->xh,&tem->score);

if(tem->score == 0) break;

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

COU\* merge(COU\* a, COU \*b)

{

COU\* head = (COU\*)malloc(sizeof(COU));

a = a->next;

b = b->next;

COU \*end = (COU\*)malloc(sizeof(COU));

end = head;

while(a!=NULL && b!=NULL)

{

if(strcmp(a->xh, b->xh) < 0) //当学号a小于学号b，串起a

{

end->next = a;

end = a;

a = a->next;

}

else //当学号b大于等于学号a，串起b

{

end->next = b;

end = b;

b = b->next;

}

}

if(a==NULL) //当a到尾部的时候

{

while(b!=NULL) //如果b还有剩余

{

end->next = b;

end = b;

b = b->next;

}

}

else //当b到尾部的时候

{

while(a!=NULL) //如果a还有剩余

{

end->next = a;

end = a;

a = a->next;

}

}

end->next = NULL;

return head;

}

void print(COU \*p)

{

printf("合并结果为：\n");

while(p!= NULL)

{

p = p->next;

printf("%s %lf\n",p->xh, p->score);

}

return;

}

int main()

{

COU \*head1, \*head2, \*head;

printf("请输入链表a，以0表示结束\n");

head1 = create(); //创建链表a

printf("请输入链表b，以0表示结束\n");

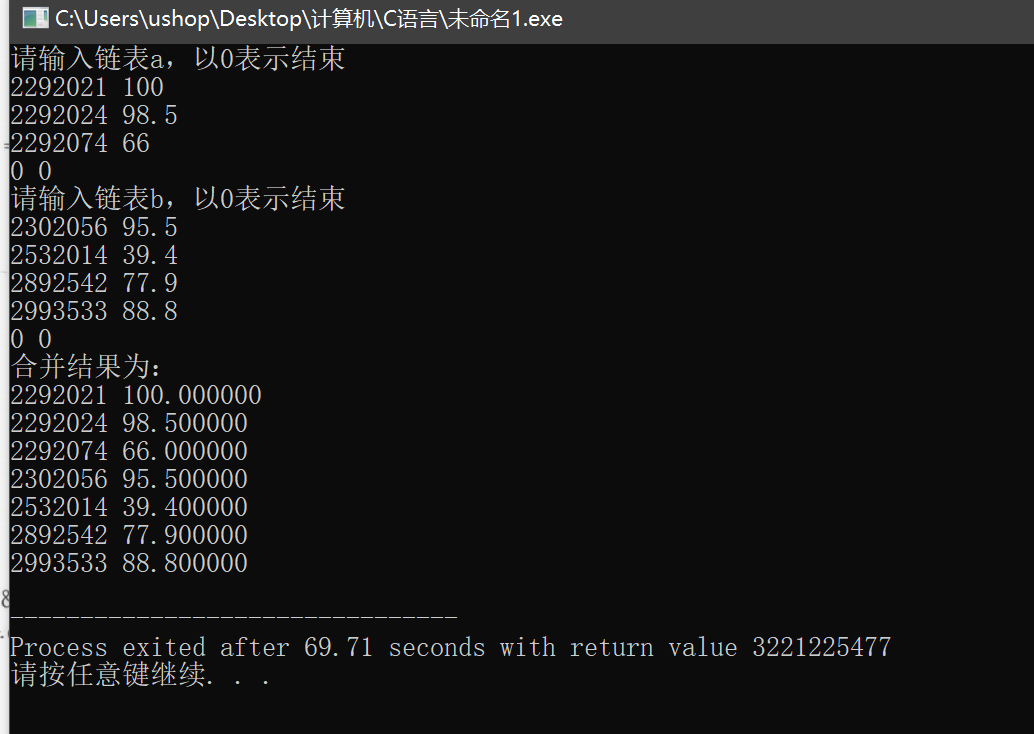
head2 = create(); //创建链表b

head = merge(head1, head2);

print(head);

return 0;

}



11.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct STU {

char xh[20]; //学号

char name[20]; //姓名

struct STU \*next;

} COU;

COU \*create() { //创建链表

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

while(1) {

tem = (COU\*)malloc(sizeof(COU));

scanf("%s %s", tem->xh,tem->name);

if(strcmp(tem->xh, "0") == 0) break;

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

COU \*del(COU \*head1, char s[]) {

COU \*p = head1->next;

COU \*pre = head1;

while(p != NULL) {

if(!strcmp(p->xh, s)) { //如果学号相等则删除

pre->next = p->next;

free(p);

p = p->next;

} else { //不相等则后移一位继续寻找

pre = p;

p = p->next;

}

}

return head1;

}

void print(COU \*p) {

printf("\n结果为：\n");

while(p!= NULL) {

p = p->next;

printf("%s %s\n",p->xh, p->name);

}

return;

}

int main() {

COU \*head1, \*head2;

printf("请输入链表a，以0表示结束\n");

head1 = create(); //创建链表a

printf("\n请输入链表b，以0表示结束\n");

head2 = create(); //创建链表b

COU \*p = head2->next;

while(p != NULL) { //遍历b中的学号，从a中找到并删除

head1 = del(head1, p->xh);

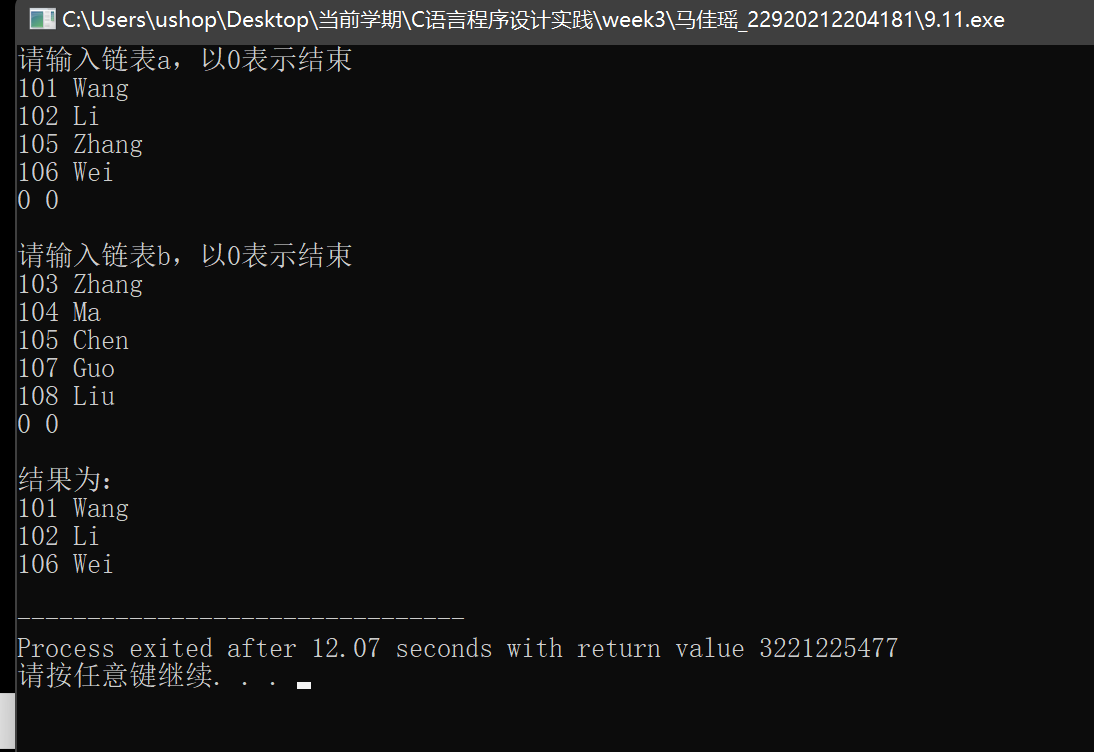
p = p->next;

}

print(head1);

return 0;

}



12.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 4

typedef struct STU {

char xh[20]; //学号

char name[20]; //姓名

char sex[8]; //性别

int age; //年龄

struct STU \*next;

} COU;

COU \*create() { //创建链表

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

for(int i = 0; i < 4; ++ i) {

tem = (COU\*)malloc(sizeof(COU));

scanf("%s %s %s %d", tem->xh,tem->name,tem->sex, &tem->age);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

COU \*del(COU \*head1, int x) {

COU \*p = head1->next;

COU \*pre = head1;

while(p != NULL) {

if(p->age == x) { //删除对应年龄

pre->next = p->next;

free(p);

p = p->next;

} else {

pre = p;

p = p->next;

}

}

return head1;

}

void print(COU \*p) {

printf("\n结果为：\n");

while(p!= NULL) {

p = p->next;

printf("%s %s %s %d\n",p->xh, p->name, p->sex, p->age);

}

return;

}

int main() {

COU \*head1;

printf("请输入学号、姓名、性别、年龄\n");

head1 = create(); //创建链表a

int age\_k;

puts("\n请输入要删除的年龄：");

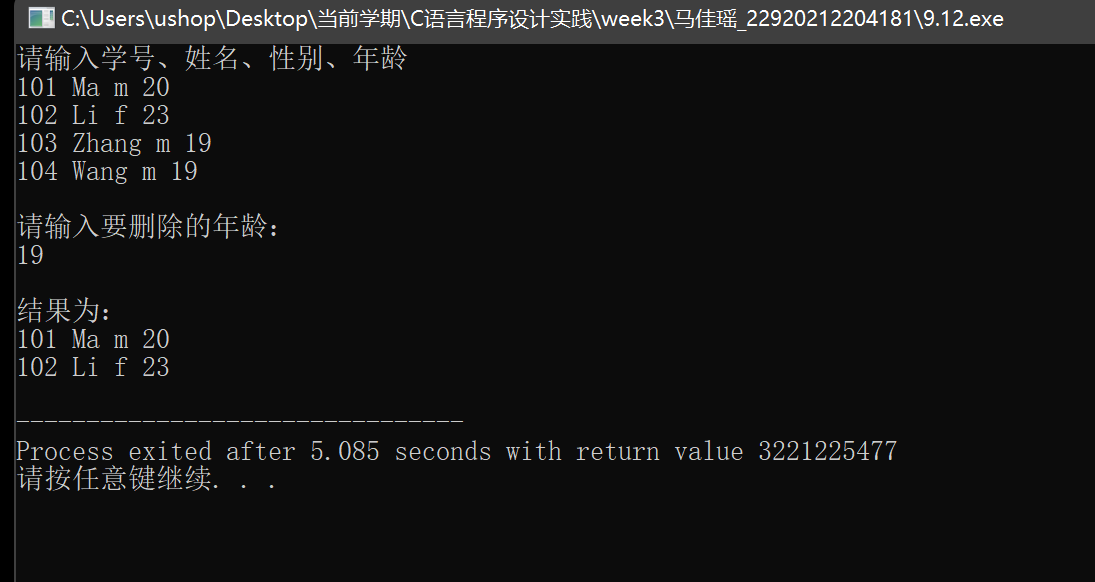
scanf("%d", &age\_k);

head1 = del(head1, age\_k);

print(head1);

return 0;

}



1.反转链表

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 7

typedef struct ListNode {

int val;

struct ListNode \*next;

} COU;

COU\* reverseList(COU\* head) {

if(head == NULL || head->next == NULL) return head;

COU \*pre, \*mid, \*end;

pre = NULL;

mid = head;

end = head->next;

while(1){

mid->next = pre; //修改指向

if(end == NULL) break; //当end为NULL时退出

pre = mid; //从左往右右移

mid = end;

end = end->next;

}

head = mid;

return head;

}

COU \*create() { //创建链表

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

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

tem = (COU\*)malloc(sizeof(COU));

scanf("%d", &tem->val);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

void print(COU \*p, COU \*end) { //因为还有没赋值的head，尾结点并不是NULL

printf("反转结果为：\n");

while(p != end) {

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

p = p->next;

}

return;

}

int main() {

COU \*head1, \*head2;

printf("请输入链表a\n");

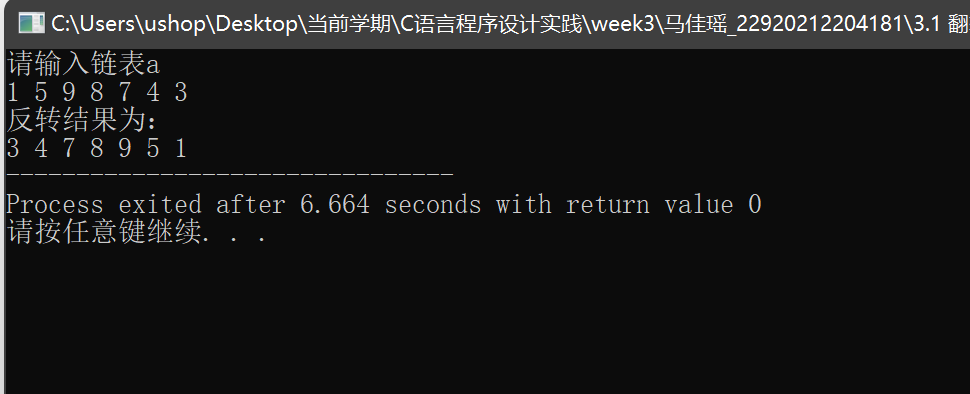
head1 = create(); //创建链表a

head2 = reverseList(head1);

print(head2, head1);

return 0;

}



1. 链表去重

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 7

typedef struct ListNode {

int val;

struct ListNode \*next;

} COU;

COU \*head1;

COU \*create() { //创建链表

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

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

tem = (COU\*)malloc(sizeof(COU));

scanf("%d", &tem->val);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

void print(COU \*p) {

printf("去重结果为：\n");

while(p != NULL) {

p = p->next;

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

}

return;

}

void del(COU \*pre, int x) {

COU \*p = pre->next;

while(p != NULL) {

if(p->val == x) {

pre->next = p->next;

free(p);

p = pre->next; //p=p->next是错的

} else {

pre = p;

p = p->next;

}

//print(head1);

}

}

int main() {

printf("请输入链表a\n");

head1 = create(); //创建链表a

COU \*p = head1->next;

while(p != NULL) {

del(p, p->val); //从节点p开始，往后找到和p一样的数字并删去

p = p->next;

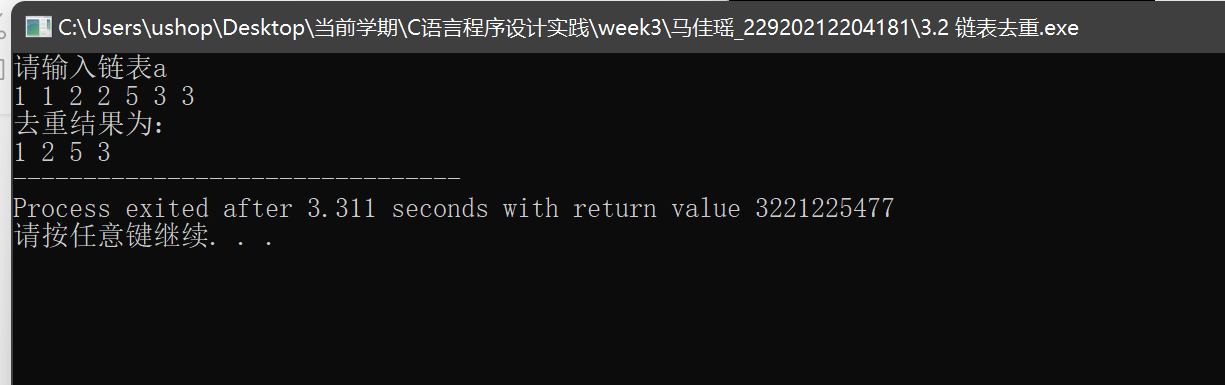
//printf("%d\n", p->val);

}

print(head1);

return 0;

}



1. 只保留出现一次的数字

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define N 7

typedef struct ListNode {

int val;

struct ListNode \*next;

} COU;

COU \*head1;

COU \*create() { //创建链表

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

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

tem = (COU\*)malloc(sizeof(COU));

scanf("%d", &tem->val);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

void print(COU \*p) {

printf("去重复数结果为：\n");

while(p != NULL) {

p = p->next;

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

}

return;

}

void del(COU \*pre) {

COU \*mid = pre->next, \*end = mid->next; //pre是重复数字的前一个，end是重复数字的后一个

if(mid == NULL || end == NULL) return; //空链表和只有一个结点的情况

while(1) {

while(mid->val == end->val){

end = end->next;

if(end == NULL) break; //当end为NULL时要退出，否则末位有重复数字会出错

}

if(mid->next != end) { //出现重复的情况

while(mid != end) {

pre->next = mid->next;

free(mid);

mid = pre->next;

}

}else{ //没有重复就全体右移

pre = mid;

mid = end;

}

if(mid == NULL) break; //边界

end = end->next;

if(end == NULL) break;//printf("%d %d %d\n", pre->val, mid->val, end->val);

//print(head1);

//puts("\*\*\*\*");

}

//print(head1);

return;

}

int main() {

printf("请输入链表a\n");

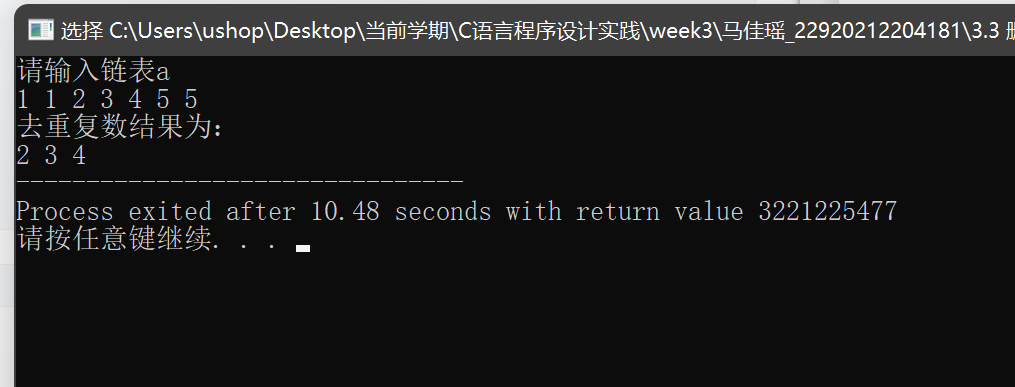
head1 = create(); //创建链表a

del(head1);

print(head1);

return 0;

}



3.4 链表相加

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct ListNode {

int val;

struct ListNode \*next;

} COU;

COU \*head1, \*head2, \*head3, \*head4;

COU \*create() { //创建链表

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

int len;

printf("请输入链表长度：");

scanf("%d", &len);

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

tem = (COU\*)malloc(sizeof(COU));

scanf("%d", &tem->val);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

COU \*add(COU \*p, COU\* q) {

int t = 0;

COU \*head, \*mid, \*end;

head = (COU\*)malloc(sizeof(COU)); //新建链表存储结果

mid = head;

while(p != NULL && q != NULL) {

t += p->val + q->val; //要考虑进位

end = (COU\*)malloc(sizeof(COU));

end->val = t % 10;

mid->next = end;

mid = end;

//puts("\*\*\*");

t /= 10;

p = p->next;

q = q->next;

}

while(p != NULL) {

end = (COU\*)malloc(sizeof(COU));

end->val = p->val;

if(t) { //考虑遗留进位

end->val += t;

t = 0;

}

mid->next = end;

mid = end;

p = p->next;

}

while(q != NULL) {

end = (COU\*)malloc(sizeof(COU));

end->val = q->val;

if(t) {

end->val += t;

t = 0;

}

mid->next = end;

mid = end;

q = q->next;

}

mid->next = NULL; //结尾

//puts("\*\*\*");

return head;

}

COU \*reverse(COU \*head) {

COU \*pre, \*mid, \*end;

pre = head;

mid = head->next;

if(pre == NULL || mid == NULL) return head;

end = mid->next;

pre = NULL; //因为头结点是不存数据的，所以直接当作结尾方便打印

while(1) {

mid->next = pre; //逐个翻转

if(end == NULL) break;

pre = mid;

mid = end;

end = end->next;

}

head = mid;

return head;

}

void print(COU \*p) {

printf("相加结果为：\n");

while(p != NULL) {

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

p = p->next;

}

return;

}

int main() {

printf("请输入链表a\n");

head1 = create(); //创建链表a

head2 = create();

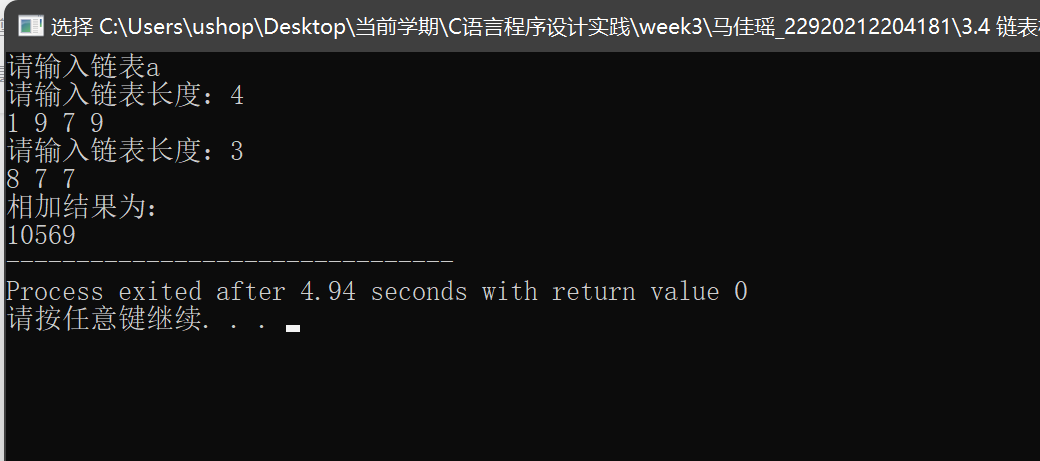
head3 = add(head1->next, head2->next);

head4 = reverse(head3);

print(head4);

return 0;

}



3.5 奇数节点偶数节点

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct ListNode {

int val;

struct ListNode \*next;

} COU;

COU \*head1, \*head2;

COU \*create() { //创建链表

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

int len;

printf("请输入链表长度：");

scanf("%d", &len);

printf("请输入链表a\n");

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

tem = (COU\*)malloc(sizeof(COU));

scanf("%d", &tem->val);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

COU \*edit(COU \*head1) {

COU \*head, \*mid, \*end;

head = (COU\*)malloc(sizeof(COU));

if(head1->next == NULL) return head;

mid = head;

COU \*p = head1->next;

while(1) { //存奇数

end = (COU\*)malloc(sizeof(COU));

end->val = p->val;

mid->next = end;

mid = end;

p = p->next;

if(p == NULL) break; //后移两个，NULL就结束

p = p->next;

if(p == NULL) break;

//puts("\*\*");

}

p = head1->next;

p = p->next;

while(1) {

end = (COU\*)malloc(sizeof(COU)); //存偶数

end->val = p->val;

mid->next = end;

mid = end;

p = p->next;

if(p == NULL) break;

p = p->next;

if(p == NULL) break;

//puts("\*\*\*");

}

mid->next = NULL;

return head;

}

void print(COU \*p) {

printf("结果为：\n");

while(p != NULL) {

p = p->next;

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

}

return;

}

int main() {

head1 = create(); //创建链表a

head2 = edit(head1);

print(head2);

return 0;

}



3.6 分割

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct ListNode {

int val;

struct ListNode \*next;

} COU;

COU \*head1, \*head2;

COU \*create() { //创建链表

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

int len;

printf("请输入链表长度：");

scanf("%d", &len);

printf("请输入链表a\n");

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

tem = (COU\*)malloc(sizeof(COU));

scanf("%d", &tem->val);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

COU \*edit(COU \*head1, int x) {

COU \*head, \*mid, \*end;

head = (COU\*)malloc(sizeof(COU));

if(head1->next == NULL) return head;

mid = head;

COU \*p = head1->next;

while(1) { //先串起小于x的数

if(p->val < x) {

end = (COU\*)malloc(sizeof(COU));

end->val = p->val;

mid->next = end;

mid = end;

}

p = p->next;

if(p == NULL) break;

//puts("\*\*");

}

p = head1->next;

while(1) { //再串起大于等于x的数

if(p->val >= x) {

end = (COU\*)malloc(sizeof(COU));

end->val = p->val;

mid->next = end;

mid = end;

}

p = p->next;

if(p == NULL) break;

//puts("\*\*");

}

mid->next = NULL;

return head;

}

void print(COU \*p) {

printf("结果为：\n");

while(p != NULL) {

p = p->next;

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

}

return;

}

int main() {

int x;

head1 = create(); //创建链表a

printf("请输入分割基准x：");

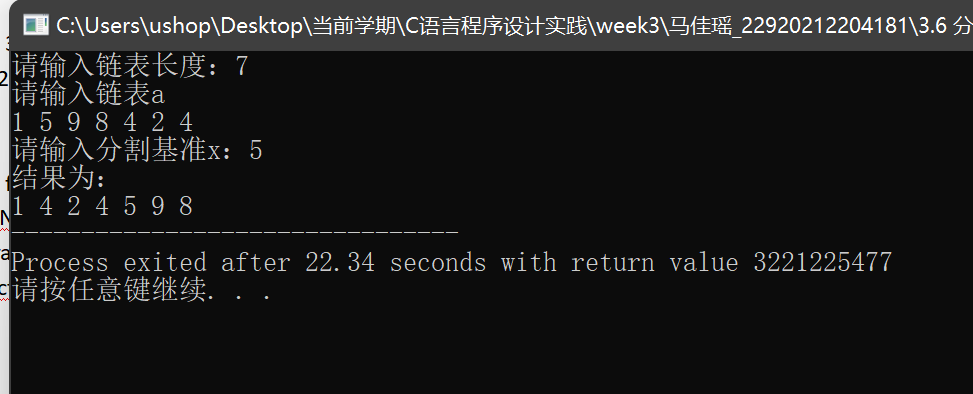
scanf("%d", &x);

head2 = edit(head1, x);

print(head2);

return 0;

}



3.7删除连续和

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct ListNode {

int val;

struct ListNode \*next;

} COU;

COU \*head1, \*head2;

COU \*create() { //创建链表

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

int len;

printf("请输入链表长度：");

scanf("%d", &len);

printf("请输入链表a\n");

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

tem = (COU\*)malloc(sizeof(COU));

scanf("%d", &tem->val);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

COU \*edit(COU \*head1) {

COU \*p, \*q, \*pre;

pre = head1; //要删的节点的前一个点

p = head1->next; //计入和的第一个点

if(p == NULL) return head1; //空链表

q = p->next;

if(q == NULL) return head1; //只有一个数

int sum = 0;

while(1) {

sum = p->val;

while(q != NULL) {

sum += q->val;//printf("%d\n", sum);

if(sum == 0) {

//printf("%d %d %d", pre->val, p->val, q->val);

pre->next = q->next;

p = pre->next;

if(p == NULL) break;

q = p->next;

sum = p->val;

} else

q = q->next;

}

pre = p;

if(pre == NULL) break; //链表没有输出的可能原因：NULL->next;

p = p->next;

if(p == NULL) break;

q = p->next;

}

return head1;

}

void print(COU \*p) {

printf("结果为：\n");

while(p != NULL) {

p = p->next;

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

}

return;

}

int main() {

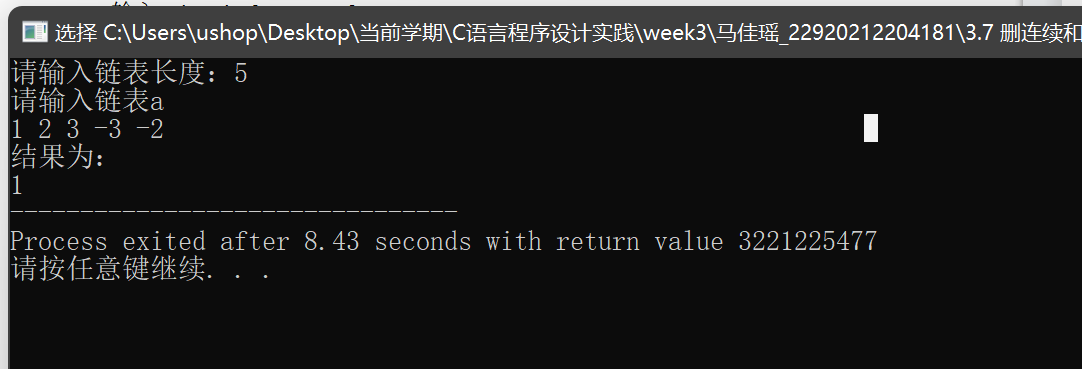
head1 = create(); //创建链表a

head2 = edit(head1);

print(head2);

return 0;

}



3.8 合并升序链表

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct Link

{

int val;

struct Link \*next;

}COU;

COU \*create() //创建链表

{

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

puts("请输入链表的长度：");

int n;

scanf("%d", &n);

printf("请输入链表\n");

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

{

tem = (COU\*)malloc(sizeof(COU));

scanf("%d",&tem->val);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

COU\* merge(COU\* a, COU \*b)

{

COU\* head = (COU\*)malloc(sizeof(COU));

a = a->next;

b = b->next;

COU \*end = (COU\*)malloc(sizeof(COU));

end = head;

while(a!=NULL && b!=NULL)

{

if(a->val <= b->val) //当学号a小于学号b，串起a

{

end->next = a;

end = a;

a = a->next;

}

else //当学号b大于等于学号a，串起b

{

end->next = b;

end = b;

b = b->next;

}

}

if(a==NULL) //当a到尾部的时候

{

while(b!=NULL) //如果b还有剩余

{

end->next = b;

end = b;

b = b->next;

}

}

else //当b到尾部的时候

{

while(a!=NULL) //如果a还有剩余

{

end->next = a;

end = a;

a = a->next;

}

}

end->next = NULL;

return head;

}

void print(COU \*p)

{

printf("合并结果为：\n");

while(p!= NULL)

{

p = p->next;

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

}

return;

}

int main()

{

COU \*head1, \*head2, \*head;

head1 = create(); //创建链表a

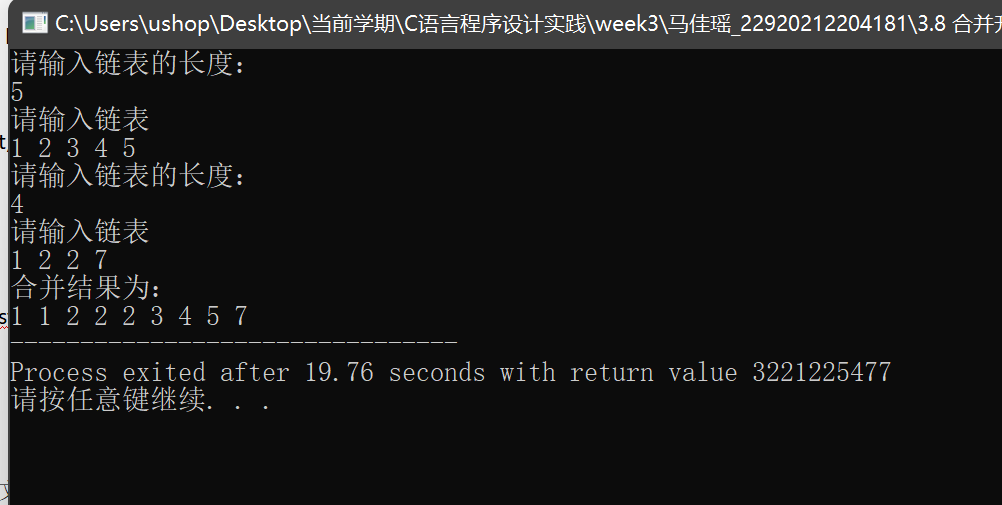
head2 = create(); //创建链表b

head = merge(head1, head2);

print(head);

return 0;

}



3.9 判断回文链表

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct ListNode {

int val;

struct ListNode \*next;

} COU;

int n;

COU\* reverseList(COU\* head) {

COU \*pre, \*mid, \*end;

pre = head->next;

mid = pre->next;

end = mid->next;

pre->next = NULL;

while(1) {

mid->next = pre; //修改指向

if(end == NULL) break; //当end为NULL时退出

pre = mid; //从左往右右移

mid = end;

end = end->next;

}

head = mid;

return head;

}

COU \*create() { //创建链表

COU \*end, \*tem, \*head;

head = (COU\*)malloc(sizeof(COU));

end = head;

printf("请输入链表长度：");

scanf("%d", &n);

printf("请输入链表：\n");

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

tem = (COU\*)malloc(sizeof(COU));

scanf("%d", &tem->val);

end->next = tem;

end = tem;

}

end->next = NULL;

return head;

}

//void print(COU \*p) {

//

// while(p != NULL) {

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

// p = p->next;

// }

// return;

//}

COU \*cpy(COU \*head1){

COU \*head2, \*mid, \*end, \*p;

p = head1->next;

head2 = (COU\*)malloc(sizeof(COU));

mid = head2;

while(p != NULL){

end = (COU\*)malloc(sizeof(COU));

end->val = p->val; //复制

p = p->next;

mid->next = end;

mid = end;

}

mid->next = NULL;

// COU \*q = head2->next;

// printf("%d\n",q->val);puts("\*\*\*");

return head2->next;

}

bool check(COU \*p, COU \*q){

while(p != NULL){

//printf("%d %d\n", p->val, q->val);

if(p->val != q->val) return false; //若不相同则返回false

p = p->next, q = q->next;

}

return true;

}

int main() {

COU \*head1, \*head2;

head1 = create(); //创建链表

if(n < 2) puts("true"); //只含1个数

else if(n < 3){ //只含2个数

COU \*a = head1->next, \*b = a->next;

if(a->val != b->val) puts("false");

else puts("true");

}

else{

head2 = cpy(head1); //printf("%d\n",head2->val); //将链表1复制到链表2

head1 = reverseList(head1); //反转链表1

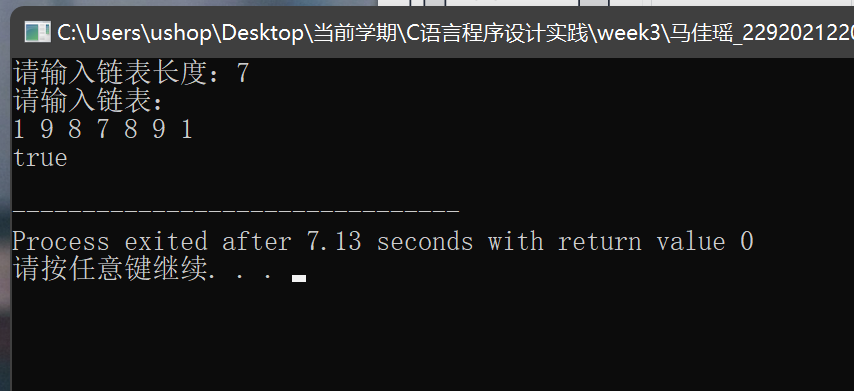
if(check(head1, head2)) puts("true"); //如果链表1和链表2相等，则是回文串

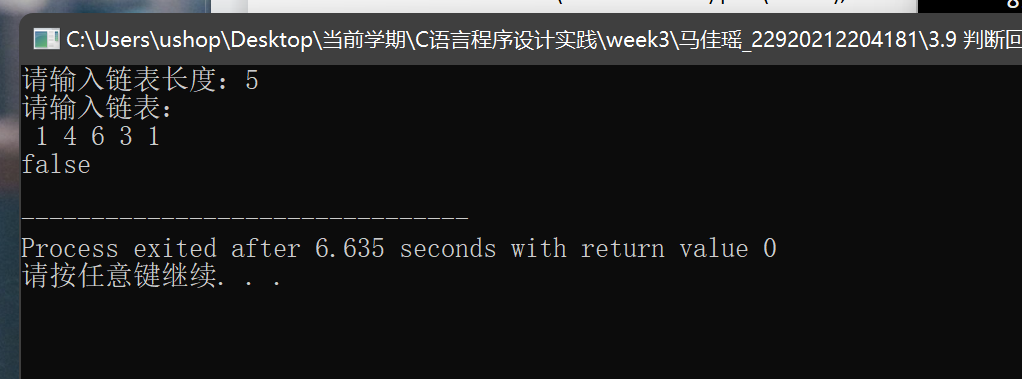
else puts("false");

}

return 0;

}





案例3;

调试链表版：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define MaxSize 20

typedef struct guest\_info {

char name[8]; //姓名

int sum; //人数

char time[10]; //时间

int number; //编号

struct guest\_info \*next;

} GuestLink, \*Pointer;

void Insert (Pointer \*Head ); //插入

void Search( Pointer Head ); //查询

void Update( Pointer Head ); //修改

void Delete( Pointer \*Head ); //删除

void Show( Pointer Head ); //显示

int main() {

Pointer Head=NULL;

int i;

do {

printf("1———插入(Insert)\n");

printf("2———查询(Serch)\n");

printf("3———修改(Update)\n");

printf("4———删除(Delete)\n");

printf("5———显示(Show)\n");

printf("6———退出(Exit)\n");

scanf("%d", &i); //输入功能选择

switch(i) {

case 1:

Insert(&Head);

break;

case 2:

Search(Head);

break;

case 3:

Update(Head);

break;

case 4:

Delete(&Head);

break;

case 5:

Show(Head);

break;

case 6:

break;

default:

printf("错误选择！请重选");

break;

}

} while(i!=6); //6则退出

return 0;

}

//这个程序不限数量，不对！

void Insert(Pointer \*Head) {

int in\_number;

Pointer p,q,r;

scanf("%d", &in\_number); //输入编号

p=q=\*Head;

while(p!=NULL) { //查找，走到链尾

if( p->number==in\_number) {

printf("已有相同编号："); //已有，不输入

return;

} else {

q=p; //q指向当前，p指向下一个

p=p->next;

}

}

r=( Pointer )malloc( sizeof( GuestLink ) ); //申请空间

r->next=NULL; //设置队尾指针域

if(r==NULL) {

printf("分配空间失败"); //应前移

return;

}

if(q==NULL) //空表？

\*Head=r; //空表则新结点为头结点

else {

q->next=r; //否则接入表尾

}

r->number=in\_number;

printf("请输入姓名：");

scanf("%s", r->name); //输入信息

printf("请输入人数：");

scanf("%d", &r->sum);

printf("请输入用餐时间：");

scanf("%s",r->time); //应该有输入错误处理！

}

void Search(Pointer Head) {

int number, flag=1;

Pointer p=Head;

printf("请输入编号");

scanf("%d", &number); //输入编号

while( p!=NULL && flag ) {

if(p->number==number) { //有则显示

printf("姓名：%s", p->name);

printf("人数：%d", p->sum);

printf("时间是：%s", p->time);

flag=0;

} //flag可不可省？

else p=p->next; //不是则指向下一结点

}

if( flag ) printf("没有查询到！！"); //这样用就对了！

}

//应该先显示再修改

void Update(Pointer Head) {

int number, flag=1;

Pointer p=Head;

printf("请输入编号");

scanf("%d", &number); //输入编号

while( p!=NULL && flag ) {

if( p->number==number ) { //是则修改

scanf("%s", p->name);

scanf("%d", &p->sum); //书上漏&

scanf("%s", p->time);

flag=0;

} //flag可不可省？

else p=p->next; //不是则指向下一结点

}

if(flag) printf("没有找到要修改的记录！！");

}

//缺点：删除花时间长。

void Delete(Pointer \*Head) {

int number, flag=1;

Pointer p,q;

printf("请输入编号");

scanf("%d", &number); //输入编号

p=q=\*Head;

while( p!=NULL && flag ) {

if( p->number==number) { //有则删除

if(p==\*Head) {

\*Head=p->next;

free(p);

} else {

q->next=p->next;

free(p);

}

flag=0;

} //flag可不可省？

else {

q=p; //q指向当前,p指向下一结点

p=p->next;

}

}

if( flag ) printf("没有找到可以删除的数据！！"); //这样用就对了

}

//行数多要换页

void Show(Pointer Head) {

Pointer p=Head;

printf("\n");

printf(" 编号 姓名 人数 时间\n");

while( p!=NULL ) { //列表显示

printf("%10d", p->number);

printf("%12s", p->name);

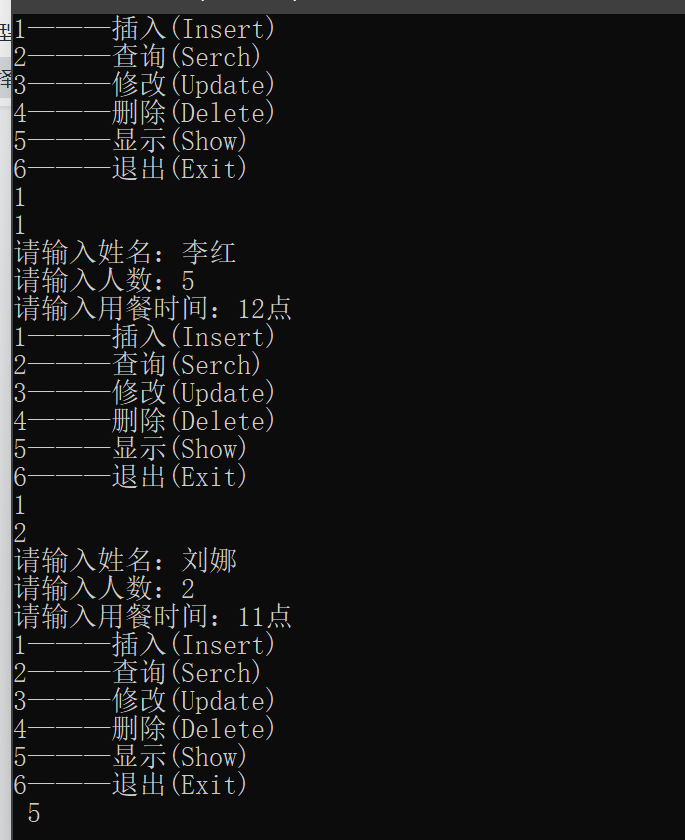
printf("%10d", p->sum);

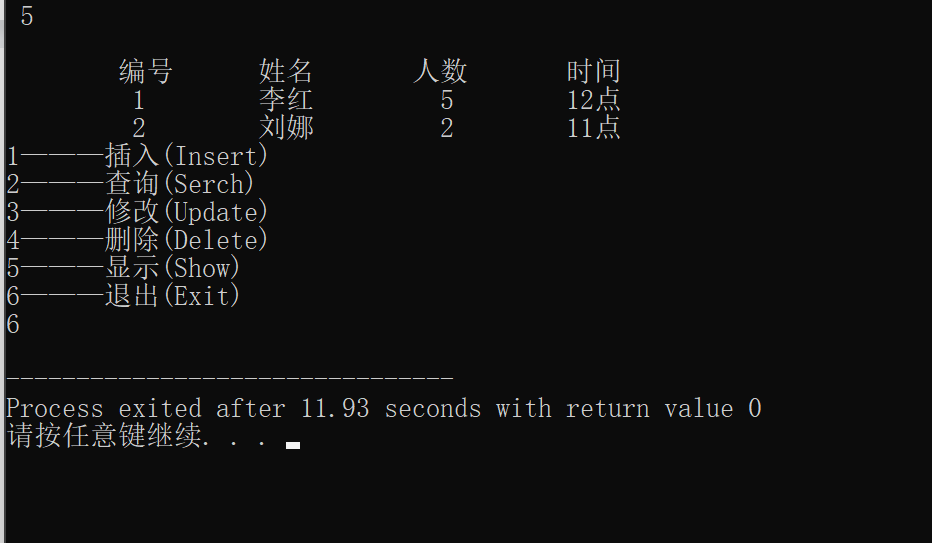
printf("%12s\n", p->time);

p=p->next; //指向下一结点

}

}





改进版：

改进1：午餐晚餐分开记：

输入部分：

if(day\_cnt >= MaxSize && night\_cnt >= MaxSize) {

puts("抱歉，预约已满！！");

} else {

printf("请输入姓名：");

scanf("%s", r->name); //输入信息

printf("请输入人数：");

scanf("%d", &r->sum);

printf("请输入用餐时间：");

scanf("%s",r->time); //应该有输入错误处理！

if(strcmp(r->time, "18点") < 0) {

if(day\_cnt + 1 > MaxSize) {

printf("该时间段已满，请另约时间\n");

while(scanf("%s", r->time), strcmp(r->time, "18点") < 0) {

printf("该时间段已满，请另约时间\n");

}

} else ++ day\_cnt;

} else {

if(night\_cnt + 1 > MaxSize) {

printf("该时间段已满，请另约时间\n");

while(scanf("%s", r->time), strcmp(r->time, "18点") >= 0) {

printf("该时间段已满，请另约时间\n");

}

} else ++ night\_cnt;

}

}

删除部分：

if(strcmp(p->time, "18点") < 0) --day\_cnt;

else -- night\_cnt;

修改部分：

while( p!=NULL && flag ) {

if(strcmp(p->time, "18点") < 0) --day\_cnt;

else -- night\_cnt;

if( p->number==number ) { //是则修改

scanf("%s", p->name);

scanf("%d", &p->sum); //书上漏&

scanf("%s", p->time);

flag=0;

if(strcmp(p->time, "18点") < 0) ++ day\_cnt;

else ++ night\_cnt;

} //flag可不可省？

else p=p->next; //不是则指向下一结点

}

改进2：不按编号输入，自动编号，订桌时间已经记录

新增变量idx：

有效输入部分:

++ idx;

printf("编号为：%d", idx);

改进3：不按编号查询，按照人名查询。

char name[30];

Pointer p=Head;

printf("请输入要查询的姓名");

scanf("%s", name); //输入编号

while( p!=NULL && flag ) {

if(strcmp(name, p->name) == 0) { //有则显示

printf("姓名：%s", p->name);

printf("人数：%d", p->sum);

printf("时间是：%s", p->time);

flag=0;

} //flag可不可省？

else p=p->next; //不是则指向下一结点

}

改进4：不按编号，按照人名修改。并在修改前显示要修改的数据。

printf("请输入姓名");

scanf("%s", name); //输入编号

while( p!=NULL && flag ) {

if(strcmp(p->time, "18点") < 0) --day\_cnt;

else -- night\_cnt;

if( strcmp(name, p->name) == 0 ) { //是则修改

puts("原数据为：");

printf("姓名：%s\n", p->name);

printf("人数：%d\n", p->sum);

printf("时间是：%s\n", p->time);

puts("请输入新的数据：");

scanf("%s", p->name);

scanf("%d", &p->sum); //书上漏&

scanf("%s", p->time);

flag=0;

if(strcmp(p->time, "18点") < 0) ++ day\_cnt;

else ++ night\_cnt;

} //flag可不可省？

else p=p->next; //不是则指向下一结点

}

改进5：删除前显示信息，并提示是否删除：

puts("将要删除的数据为：");

printf("姓名：%s\n", p->name);

printf("人数：%d\n", p->sum);

printf("时间是：%s\n", p->time);

puts("按1继续删除，按0放弃删除：");

int k;

scanf("%d", &k);

if(k == 0) break;

改进6：午餐晚餐分开显示。

void Show(Pointer Head) {

Pointer p=Head;

printf("\n");

printf(" 编号 姓名 人数 时间\n");

while( p!=NULL ) { //列表显示

if(strcmp(p->time, "18点") < 0) { //白天

printf("%10d", p->number);

printf("%12s", p->name);

printf("%10d", p->sum);

printf("%12s\n", p->time);

}

p=p->next; //指向下一结点

}

p=Head;

while( p!=NULL ) { //列表显示

if(strcmp(p->time, "18点") >= 0) { //晚餐

printf("%10d", p->number);

printf("%12s", p->name);

printf("%10d", p->sum);

printf("%12s\n", p->time);

}

p=p->next; //指向下一结点

}

}

改进7：存到文件中

if((fp = fopen("d:\\Order.bin", "w+")) == NULL) {

puts("can not open file!");

}

//在主程序

if((fwrite(Head, sizeof(GuestLink), n, fp)) != n) {

puts("Can not write file!");

}

fclose(fp);

改进8： 输入部分的上限判断，如果当时预约已满，则不允许再插入数据。输入有误则重新输入。

if(strcmp(r->time, "18点") < 0) {

if(day\_cnt + 1 > MaxSize) {

printf("该时间段已满，请另约时间\n");

while(scanf("%s", r->time), strcmp(r->time, "18点") < 0) {

printf("该时间段已满，请另约时间\n");

}

} else ++ day\_cnt;

} else {

if(night\_cnt + 1 > MaxSize) {

printf("该时间段已满，请另约时间\n");

while(scanf("%s", r->time), strcmp(r->time, "18点") >= 0) {

printf("该时间段已满，请另约时间\n");

}

} else ++ night\_cnt;

}

综合版：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define MaxSize 20

typedef struct guest\_info {

char name[8]; //姓名

int sum; //人数

char time[10]; //时间

int number; //编号

struct guest\_info \*next;

} GuestLink, \*Pointer;

void Insert (Pointer \*Head ); //插入

void Search( Pointer Head ); //查询

void Update( Pointer Head ); //修改

void Delete( Pointer \*Head ); //删除

void Show( Pointer Head ); //显示

int day\_cnt, night\_cnt, idx;

FILE \*fp;

int main() {

Pointer Head=NULL;

int i;

if((fp = fopen("d:\\Order.bin", "w+")) == NULL) {

puts("can not open file!");

}

do {

printf("\n1———插入(Insert)\n");

printf("2———查询(Serch)\n");

printf("3———修改(Update)\n");

printf("4———删除(Delete)\n");

printf("5———显示(Show)\n");

printf("6———退出(Exit)\n");

scanf("%d", &i); //输入功能选择

switch(i) {

case 1:

Insert(&Head);

break;

case 2:

Search(Head);

break;

case 3:

Update(Head);

break;

case 4:

Delete(&Head);

break;

case 5:

Show(Head);

break;

case 6:

break;

default:

printf("错误选择！请重选");

break;

}

} while(i!=6);

int n = day\_cnt + night\_cnt;

if((fwrite(Head, sizeof(GuestLink), n, fp)) != n) {

puts("Can not write file!");

}

fclose(fp); //6则退出

return 0;

}

void Insert(Pointer \*Head) {

Pointer p,q,r;

p=q=\*Head;

while(p!=NULL) { //查找，走到链尾

if( p->number==idx + 1) {

printf("已有相同编号："); //已有，不输入

return;

} else {

q=p; //q指向当前，p指向下一个

p=p->next;

}

}

r=( Pointer )malloc( sizeof( GuestLink ) ); //申请空间

r->next=NULL; //设置队尾指针域

if(r==NULL) {

printf("分配空间失败"); //应前移

return;

}

if(q==NULL) //空表

\*Head=r; //空表则新结点为头结点

else {

q->next=r; //否则接入表尾

}

if(day\_cnt >= MaxSize && night\_cnt >= MaxSize) {

puts("抱歉，预约已满！！");

return;

} else {

++ idx;

printf("当前编号为：%d\n", idx);

r->number=idx;

printf("请输入姓名：");

scanf("%s", r->name); //输入信息

printf("请输入人数：");

scanf("%d", &r->sum);

printf("请输入用餐时间：");

scanf("%s",r->time);

if(strcmp(r->time, "18点") < 0) {

if(day\_cnt + 1 > MaxSize) {

printf("该时间段已满，请另约时间\n");

while(scanf("%s", r->time), strcmp(r->time, "18点") < 0) {

printf("该时间段已满，请另约时间\n");

}

} else ++ day\_cnt;

} else {

if(night\_cnt + 1 > MaxSize) {

printf("该时间段已满，请另约时间\n");

while(scanf("%s", r->time), strcmp(r->time, "18点") >= 0) {

printf("该时间段已满，请另约时间\n");

}

} else ++ night\_cnt;

}

}

}

void Search(Pointer Head) {

int flag=1;

char name[30];

Pointer p=Head;

printf("请输入要查询的姓名：\n");

scanf("%s", name); //输入编号

while( p!=NULL && flag ) {

if(strcmp(name, p->name) == 0) { //有则显示

printf("姓名：%s\n", p->name);

printf("人数：%d\n", p->sum);

printf("时间是：%s\n", p->time);

flag=0;

} //flag可省

else p=p->next; //不是则指向下一结点

}

if( flag ) printf("没有查询到！！\n");

}

//应该先显示再修改

void Update(Pointer Head) {

int flag=1;

char name[30];

Pointer p=Head;

printf("\n请输入姓名：");

scanf("%s", name); //输入编号

while( p!=NULL && flag ) {

if(strcmp(p->time, "18点") < 0) --day\_cnt;

else -- night\_cnt;

if( strcmp(name, p->name) == 0 ) { //是则修改

puts("原数据为：");

printf("姓名：%s\n", p->name);

printf("人数：%d\n", p->sum);

printf("时间是：%s\n", p->time);

puts("\n请输入新的数据：");

scanf("%s", p->name);

scanf("%d", &p->sum); //书上漏&

scanf("%s", p->time);

flag=0;

if(strcmp(p->time, "18点") < 0) ++ day\_cnt;

else ++ night\_cnt;

} //flag可不可省？

else p=p->next; //不是则指向下一结点

}

if(flag) printf("没有找到要修改的记录！！");

}

//缺点：删除花时间长。

void Delete(Pointer \*Head) {

int flag=1;

char name[30];

Pointer p,q;

printf("\n请输入姓名：");

scanf("%s", name); //输入编号

p=q=\*Head;

while( p!=NULL && flag ) {

if( strcmp(name, p->name) == 0) { //有则删除

flag=0;

puts("将要删除的数据为：");

printf("姓名：%s\n", p->name);

printf("人数：%d\n", p->sum);

printf("时间是：%s\n", p->time);

puts("按1继续删除，按0放弃删除：");

int k;

scanf("%d", &k);

if(k == 0) break;

if(p==\*Head) {

\*Head=p->next;

free(p);

} else {

q->next=p->next;

free(p);

}

if(strcmp(p->time, "18点") < 0) --day\_cnt;

else -- night\_cnt;

} //flag可不可省？

else {

q=p; //q指向当前,p指向下一结点

p=p->next;

}

}

if( flag ) printf("没有找到可以删除的数据！！"); }

//行数多要换页

void Show(Pointer Head) {

Pointer p=Head;

printf("\n");

puts("\n午餐预约名单：\n");

printf(" 编号 姓名 人数 时间\n");

while( p!=NULL ) { //列表显示

if(strcmp(p->time, "18点") < 0) { //白天

printf("%10d", p->number);

printf("%12s", p->name);

printf("%10d", p->sum);

printf("%12s\n", p->time);

}

puts("");

p=p->next; //指向下一结点

}

p=Head;

puts("\n晚餐预约名单：\n");

printf(" 编号 姓名 人数 时间\n");

while( p!=NULL ) { //列表显示

if(strcmp(p->time, "18点") >= 0) { //晚餐

printf("%10d", p->number);

printf("%12s", p->name);

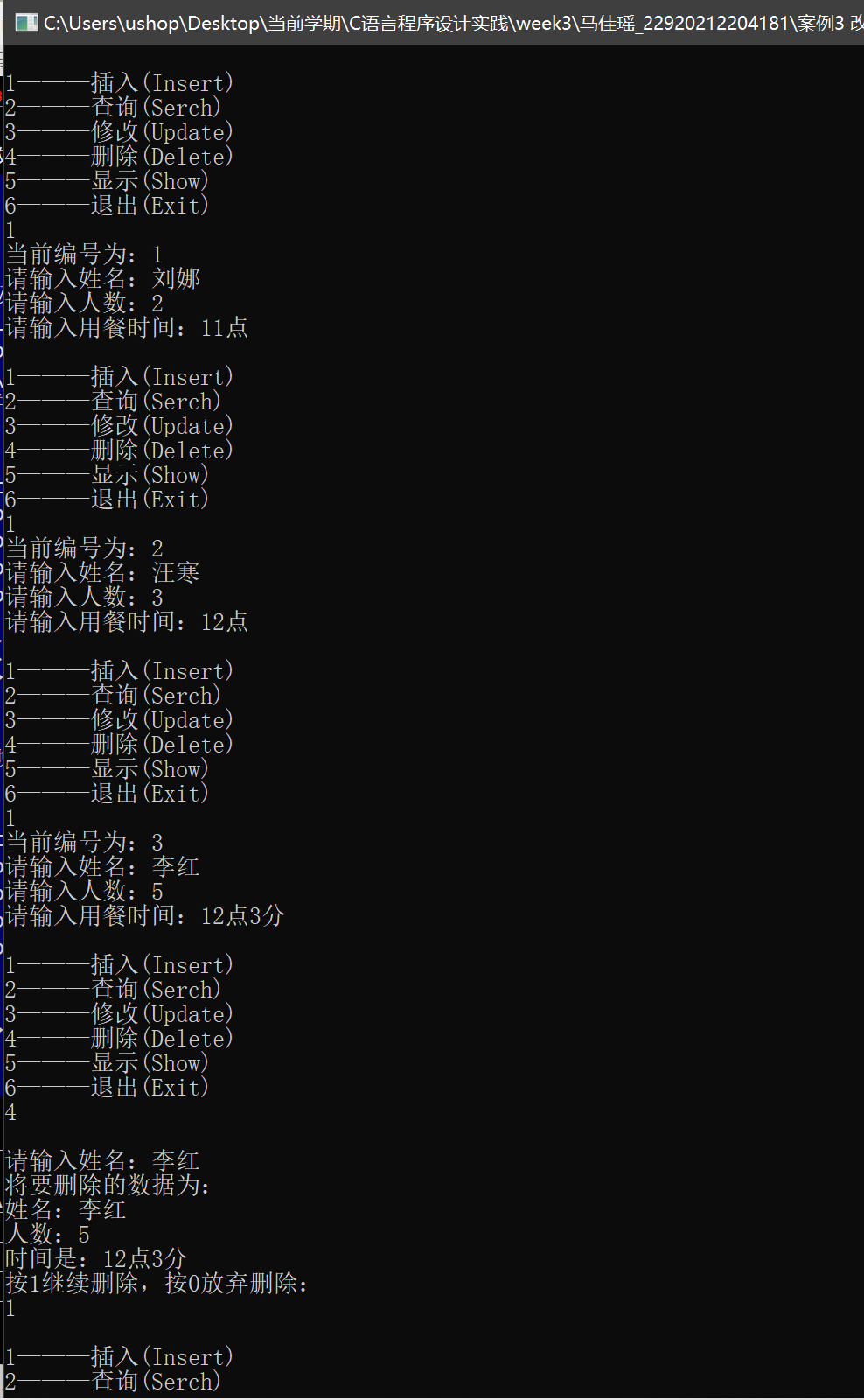
printf("%10d", p->sum);

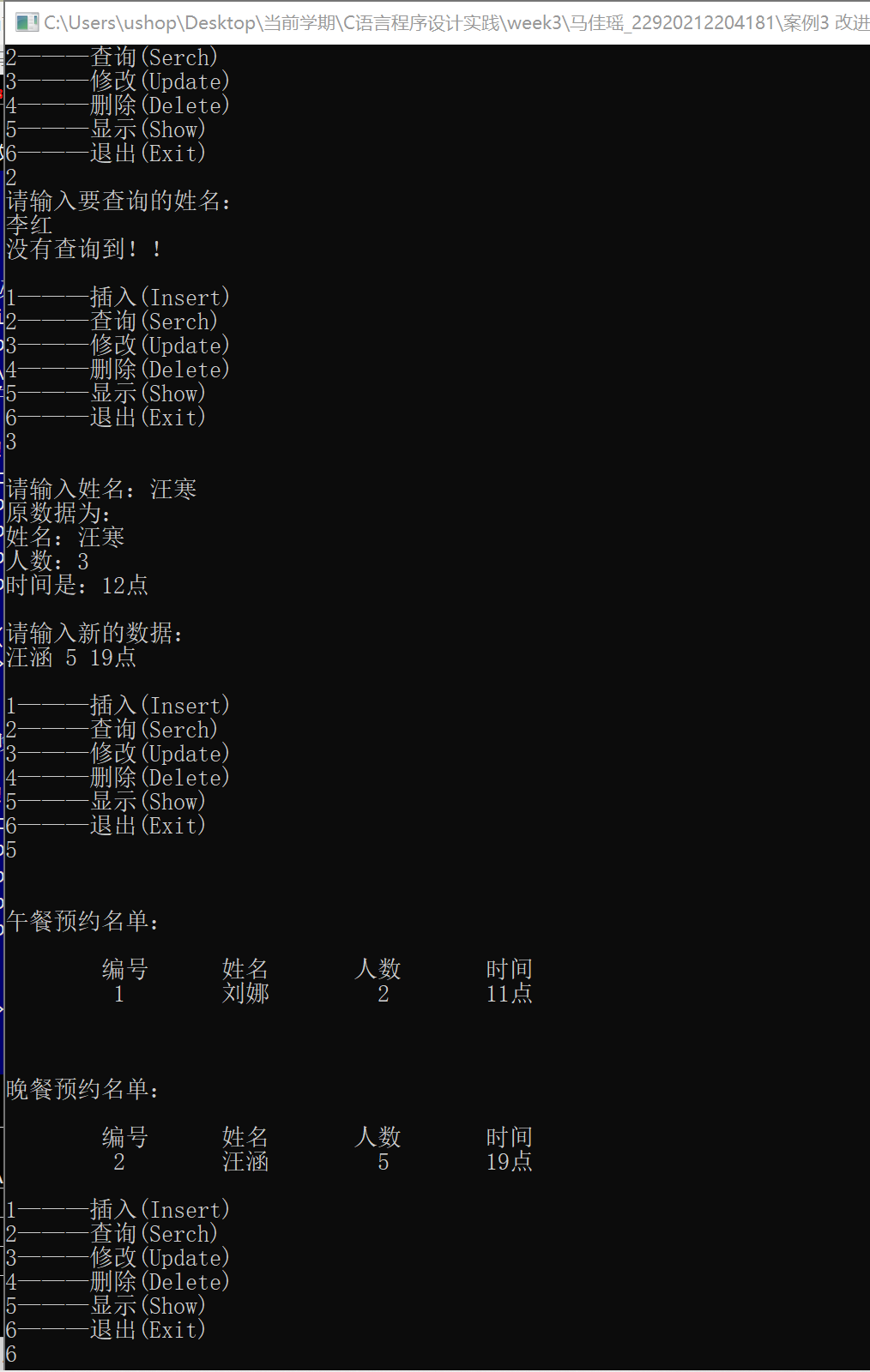
printf("%12s\n", p->time);

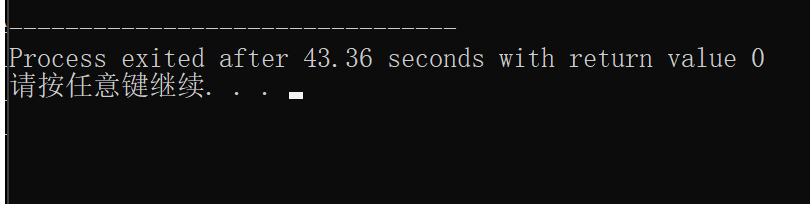
}

p=p->next; //指向下一结点

}

}





另：其实按名字输入查询也不太合理，因为要考虑重名的情况。按照手机号查询比较合理。

栈

1）b是非法序列，a, c, d是合法序列

2）

#include<stdio.h>

int main(){

char s[100];

scanf("%s", s);

int cnt = 0;

bool flag = true;

for(int i = 0; s[i] != '\0'; ++ i){

if(s[i] == 'I') ++ cnt;

else --cnt;

if(cnt < 0){

flag = false;

puts("非法序列！");

break;

}

//printf("%d\n", cnt);

}

if(flag) puts("合法序列！");

return 0;

}

