**C语言 第七次作业**

**2018011446 刘泓尊**

1. **计算日期的差值**

//判断日期差

#include<stdio.h>

struct time{

int year;

int month;

int day;

}date[2];

int JudgeLeapYear(int year)//判断闰年

{

if((year%4==0&&year%100!=0)||(year%400==0))return 1;

return 0;

}

int JudgeCorrect(time date)//判断输入日期是否合理

{

if(date.month<1||date.month>12||date.day<1)return 0;

else{

switch(date.month){

case 1:case 3:case 5:case 7:case 8:case 10:case 12:{

if(date.day>31)return 0;

break;

}

case 2:{

if((JudgeLeapYear(date.year)&&date.day>29)||((!JudgeLeapYear(date.year))&&date.day>28))

return 0;

break;

}

default:{

if(date.day>30)return 0;

break;

}

}

}

return 1;

}

int TimeDifference(time date1,time date2)//核心函数，求两个日期的差值

{

int day[2]={date1.day,date2.day};

int result=0;

int monthdate[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};

time tmp;

//判断第一个日期是否小于第二个日期，否则交换两个日期

if(date1.year>date2.year||(date1.year==date2.year&&date1.month>date2.month)||(date1.year==date2.year&&date1.month==date2.month&&date1.day>date2.day))

{

tmp=date1;

date1=date2;

date2=tmp;

}

if(date1.year==date2.year){

for(int i=date1.month;i<date2.month;i++)

{

if(i==2&&JudgeLeapYear(date1.year))result++;

result+=monthdate[i];

}

result+=date2.day-date1.day;

}

else{

//先分别计算两个日期是该年的第几天

for(int i=1;i<date1.month;i++)

{

if(i==2&&JudgeLeapYear(date1.year))day[0]++;

day[0]+=monthdate[i];

}

for(int i=1;i<date2.month;i++)

{

if(i==2&&JudgeLeapYear(date2.year))day[1]++;

day[1]+=monthdate[i];

}

result=day[1]-day[0];

for(int i=date1.year;i<date2.year;i++)

{

if(JudgeLeapYear(i))result++;

result+=365;

}

}

return result;

}

int main()

{

for(int i=0;i<2;i++){

printf("please input date%d:(格式为year month day)",i+1);

scanf\_s("%d%d%d",&date[i].year,&date[i].month,&date[i].day);

if(!JudgeCorrect(date[i])){

printf("Error!please input the correct date!\n");

i--;

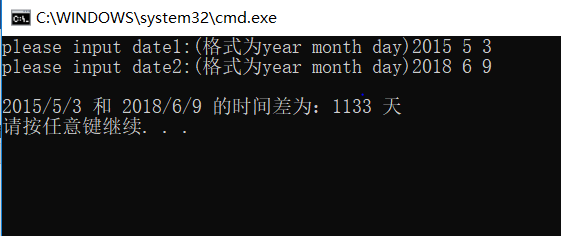
}

}

printf("\n%d/%d/%d 和 %d/%d/%d 的时间差为：%d 天\n",date[0].year,date[0].month,date[0].day,date[1].year,date[1].month,date[1].day,TimeDifference(date[0],date[1]));

return 0;

}



1. **结构体数组的应用**

//学生信息管理

#include<stdio.h>

#include<malloc.h>

#include<string.h>

struct student{

char name[20];

char number[20];

char nativeplace[50];

char ID[20];

int age;

char location[50];

char sex[8];

char contact[20];

};

student\*\*input(int \*n)

{

student\*\*p;

printf("please input the total number of students:");

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

p=(student\*\*)malloc((\*n)\*sizeof(student\*));

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

p[i]=(student\*)malloc(sizeof(student));

printf("please input the information of students:\nname |number |nativeplace |ID |age|location |sex |contact information\n");

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

{ scanf\_s("%s%s%s%s%d%s%s%s",p[i]->name,20,p[i]->number,20,p[i]->nativeplace,50,p[i]->ID,20,&(p[i]->age),p[i]->location,50,p[i]->sex,8,p[i]->contact,20);

fflush(stdin);

}

return p;

}

void print(student\*\*p,int n)

{

printf("\nThe information of all the students are:\n");

printf("name |number |nativeplace |ID |age|location |sex |contact information\n");

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

{

printf("%-5s|%-10s|%-15s|%-20s|%-3d|%-15s|%-5s|%-20s\n",p[i]->name,p[i]->number,p[i]->nativeplace,p[i]->ID,p[i]->age,p[i]->location,p[i]->sex,p[i]->contact);

}

}

student\*search(student\*\*p,int n,char number[])

{

int i;

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

{

if(strcmp(number,p[i]->number)==0)

return p[i];

}

return NULL;

}

int main()

{

int n=0;

student\*\*pointer,\*result;

char num[20];

pointer=input(&n);

print(pointer,n);

printf("\nplease input the number you wanna search:");

scanf\_s("%s",num,20);

result=search(pointer,n,num);

if(result!=NULL)

{

printf("\nThe information of this student is:\nname |number |nativeplace |ID |age|location |sex |contact information\n");

printf("%-5s|%-10s|%-15s|%-20s|%-3d|%-15s|%-5s|%-20s\n",result->name,result->number,result->nativeplace,result->ID,result->age,result->location,result->sex,result->contact);

}

else printf("Not Found!\n");

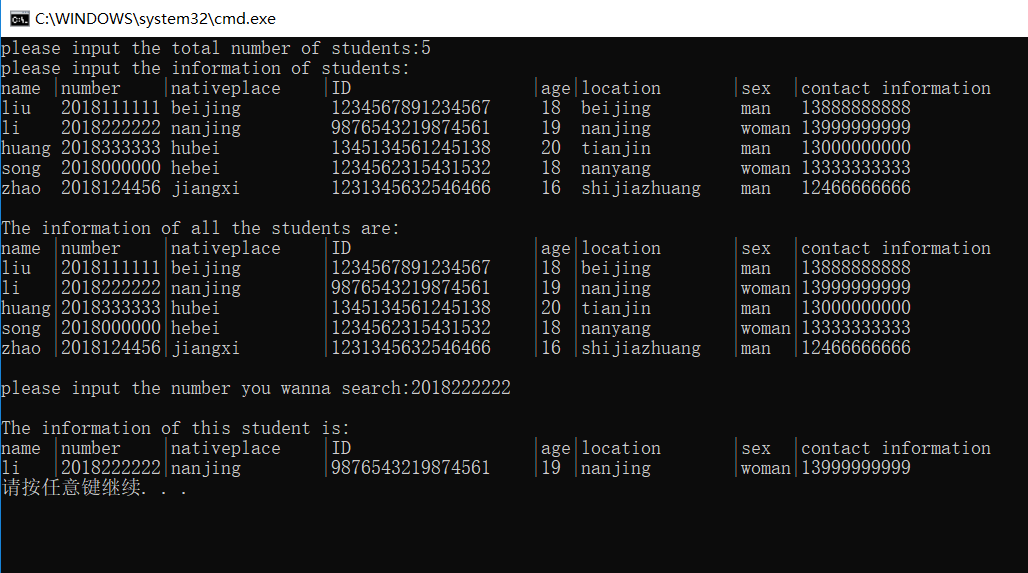
for(int i=0;i<n;i++)

free(pointer[i]);

free(pointer);

return 0;

}



1. **单向链表练习**

#include<stdio.h>

#include<malloc.h>

#include<string.h>

struct student{

char num[20];

char name[10];

student\*next;

};

student\*insert(student\*head,student\*stu)//递增插入新节点

{

student\*p0,\*p1,\*p2;

p0=stu;p1=head;

if(head==NULL){head=p0;p0->next=NULL;}

else{

while(strcmp(p0->num,p1->num)>0&&(p1->next!=NULL)){

p2=p1;

p1=p1->next;

}

if(strcmp(p0->num,p1->num)<=0){

if(head==p1)head=p0;

else p2->next=p0;

p0->next=p1;

}

else{

p1->next=p0;

p0->next=NULL;

}

}

return head;

}

student\*creat()

{

void list(student\*head);

student\*head,\*p1;int count=0,n;

printf("please input the number of students:");

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

printf("please input the information of students:\n");

head=NULL;

do{

p1=(student\*)malloc(sizeof(student));

scanf\_s("%s%s",p1->num,20,p1->name,10);

head=insert(head,p1);

list(head);//每次插入后输出当前链表

count++;

}while(count<n);

return head;

}

int f(student\*head)//节点计数

{

student\*p1;int count=0;

p1=head;

while(p1->next!=NULL){

p1=p1->next;

count++;

}

count++;

return count;

}

student\*fv(student\*head)//链表倒序

{

student\*p1,\*p2,\*p3;

p1=head;

p2=p3=head->next;

while(p2!=NULL){

p3=p2->next;

p2->next=p1;

p1=p2;

p2=p3;

}

head->next=NULL;

return p1;

}

void list(student\*head)//输出链表数据

{

student\*p1;

int count=1;

p1=head;

printf("当前链表:\nhead -> ");

while(p1!=NULL){

if(count%4==0)putchar(10);

printf("number:%s name：%s -> ",p1->num,p1->name);

count++;

p1=p1->next;

}

printf("NULL\n");

}

student\*search(student\*head,char\*num)//查找信息

{

student\*p1;

p1=head;

while(p1!=NULL){

if(strcmp(num,p1->num)==0)return p1;

p1=p1->next;

}

return NULL;

}

void myfree(student\*head)//释放内存，遍历释放

{

student\*p;

while(head!=NULL){

p=head;

head=head->next;

free(p);

}

}

int main()

{

student\*head,\*result;

char num[20];

head=creat();

if(head==NULL)printf("空链表！\n");

else{

printf("\n节点个数：%d\n",f(head));

head=fv(head);

printf("\n倒序后的链表：\n");

list(head);

printf("\nplease input the num you wanna search:");

scanf\_s("%s",num,20);

if((result=search(head,num))!=NULL)

printf("学号:%s,姓名:%s\n",result->num,result->name);

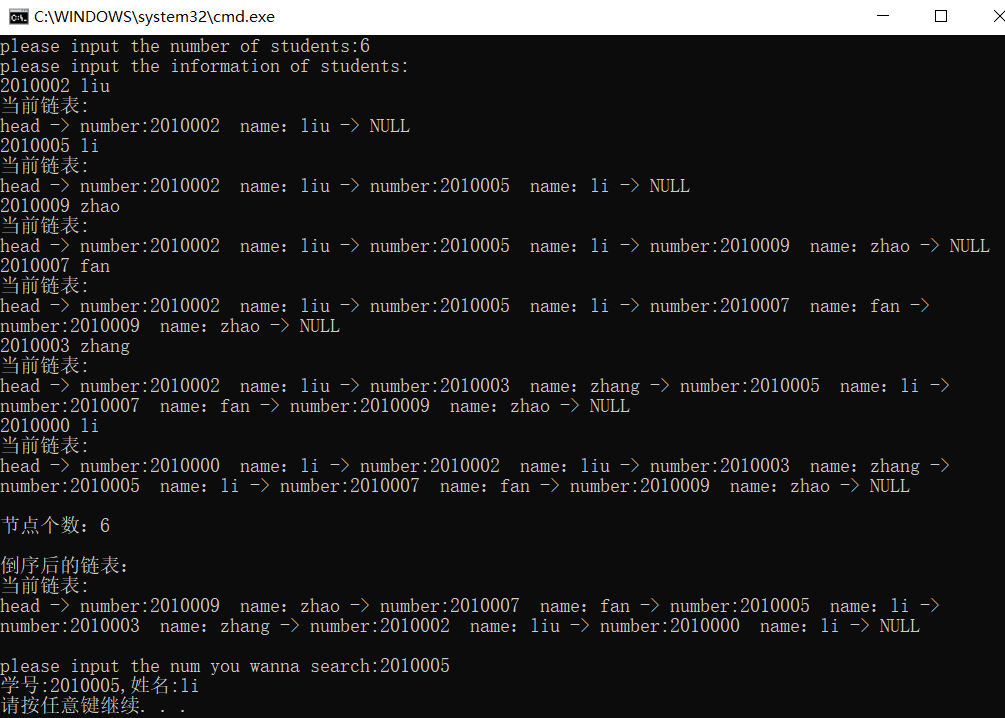
else printf("该学号不存在！\n");

}

myfree(head);

return 0;

}



1. **一元多项式加法**

//稀疏多项式加法

#include<stdio.h>

#include<math.h>

#include<malloc.h>

struct polynomial{

int coefficient;

int power;

polynomial\*next;

};

polynomial\*insert(polynomial\*head,polynomial\*p0)//递增插入项数

{

polynomial\*p1,\*p2;

p1=p2=head;

if(head==NULL){head=p0;p0->next=NULL;}

else{

while(p0->power>p1->power&&p1->next!=NULL){

p2=p1;

p1=p1->next;

}

if(p0->power<=p1->power){

if(head==p1){head=p0;p0->next=p1;}

else{

p2->next=p0;

p0->next=p1;

}

}

else{

p1->next=p0;

p0->next=NULL;

}

}

return head;

}

polynomial\*creat(polynomial\*head)//动态创建节点

{

polynomial \*p1;

int n,count=1;

printf("请输入多项式项数:");

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

printf("请分行输入各项的系数指数:\n");

p1=(polynomial\*)malloc(sizeof(polynomial));

scanf\_s("%d%d",&p1->coefficient,&p1->power);

head=insert(head,p1);

while(1){

p1=(polynomial\*)malloc(sizeof(polynomial));

scanf\_s("%d%d",&p1->coefficient,&p1->power);

head=insert(head,p1);//保证链表中的幂次是递增的

count++;

if(count==n)break;

}

return head;

}

void print(polynomial\*head)

{

polynomial\*p1;

p1=head;

printf("新多项式的系数和次数分别为:\n");

while(p1!=NULL){

printf("%4d %4d\n",p1->coefficient,p1->power);

p1=p1->next;

}

}

polynomial\*search(polynomial\*head,polynomial\*p)//查找是否有相同次数的项

{

polynomial\*p1;

p1=head;

while(p1!=NULL){

if(p1->power==p->power)return p1;

p1=p1->next;

}

return NULL;

}

polynomial\*add(polynomial\*head1,polynomial\*head2,int\*n)//实现多项式加法

{

int flag=0;

polynomial\*newhead,\*p1,\*p2,\*tmp,\*p;

newhead=head1;

for(p2=head2;p2!=NULL;){

if((p1=search(head1,p2))!=NULL){

p1->coefficient+=p2->coefficient;

p=p2;

p2=p2->next;

free(p);//释放内存

}

else{

tmp=p2->next;

head1=insert(head1,p2);

p2=tmp;

}

}

p1=head1;

while(p1!=NULL){//统计新多项式的项数

(\*n)++;

p1=p1->next;

}

return head1;

}

void myfree(polynomial\*head)//释放内存

{

polynomial\*p;

while(head!=NULL){

p=head;

head=head->next;

free(p);

}

}

int main()

{

int n=0;polynomial\*head1,\*head2;

head1=head2=NULL;

head1=creat(head1); head2=creat(head2);

head1=add(head1,head2,&n);

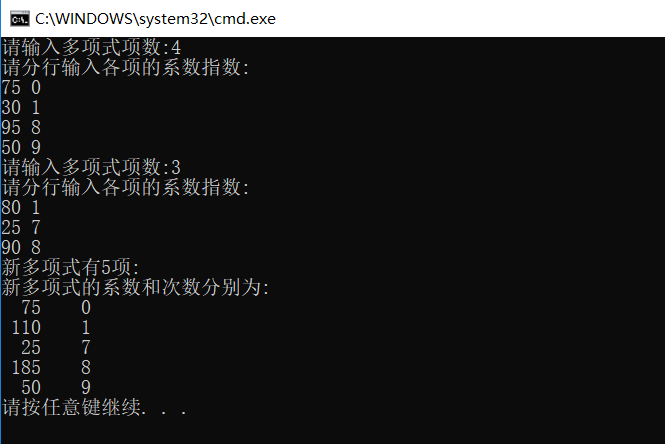
printf("新多项式有%d项:\n",n);

print(head1);

myfree(head1);

return 0;

}



1. **循环淘汰**

//约瑟夫问题，链表

#include<stdio.h>

#include<malloc.h>

struct joseph{

int state;//存放状态，未淘汰则为初始编号，淘汰为0

joseph\*next;

};

joseph\*creat(joseph\*head,int n)//动态创建环链表

{

joseph\*p1,\*p2;

p1=p2=(joseph\*)malloc(sizeof(joseph));

p1->state=1;

for(int i=1;i<n;i++){

if(head==NULL)head=p1;

p2=p1;

p1=(joseph\*)malloc(sizeof(joseph));

p1->state=i+1;

p2->next=p1;

}

p1->next=head;

return head;

}

joseph\*del(joseph\*head,joseph\*p)//删除相应节点

{

joseph\*p1,\*p2;

p1=p2=head;

do{

p2=p1;

p1=p1->next;

if(p1->state==p->state){

if(p1==head)head=p2;

p2->next=p1->next;

free(p1);//释放内存

p1=p2->next;

break;

}

}while(1);

return head;

}

int left(joseph\*head,int n,int m)

{

int i,alive=n,result;

joseph\*p1,\*tmp;

p1=head;

for(i=1;alive>1;i++){

tmp=p1->next;

if(i%m==0){

head=del(head,p1);

alive--;

}

p1=tmp;

}

result=head->state;

free(head);//释放最后一个节点

return result;

}

int main()

{

int n,m;joseph\*head;

head=NULL;

printf("please input N and m:");

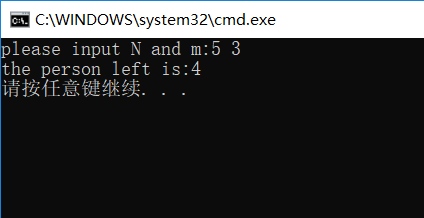
scanf\_s("%d%d",&n,&m);

head=creat(head,n);

printf("the person left is:%d\n",left(head,n,m));

return 0;

}



1. **工资单处理**

#include<stdio.h>

#include<malloc.h>

#include<string.h>

struct salary{

char num[10];

char name[20];

int money;

salary\*next;

};

salary\*insert(salary\*head,salary\*p0)//按升序插入新的节点

{

salary\*p1,\*p2;

p1=p2=head;

if(head==NULL){head=p0;p0->next=NULL;}

else{

while(strcmp(p0->num,p1->num)>0&&p1->next!=NULL){

p2=p1;

p1=p1->next;

}

if(strcmp(p0->num,p1->num)<=0){

if(p1==head){

head=p0;

p0->next=p1;

}

else{

p2->next=p0;

p0->next=p1;

}

}

else{

p1->next=p0;

p0->next=NULL;

}

}

return head;

}

salary\*combin(salary\*list1,salary\*list2)

{

salary\*list,\*p1,\*p2,\*tmp;

p1=list1;list=NULL;

while(p1!=NULL){

tmp=p1->next;

list=insert(list,p1);

p1=tmp;

}

p2=list2;

while(p2!=NULL){

tmp=p2->next;

list=insert(list,p2);

p2=tmp;

}

return list;

}

void print(salary\*list)

{

salary\*p;

p=list;

printf("链表内容为：\n");

while(p!=NULL){

printf("%s,%s,%d\n",p->num,p->name,p->money);

p=p->next;

}

}

void rank(salary\*list,salary\*pointer[])

{

salary\*p;

int min;

p=list;

for(int i=0;i<7;i++){

pointer[i]=p;

p=p->next;

}

//选择排序

for(int i=0;i<7;i++){

min=pointer[i]->money;

for(int j=i;j<7;j++){

if(pointer[j]->money<min)min=pointer[j]->money;

if(min!=pointer[i]->money){

salary\*tmp=pointer[i];

pointer[i]=pointer[j];

pointer[j]=tmp;

}

}

}

}

void print\_p(salary\*p[])

{

printf("\n按工资排序的结果为:\n");

for(int i=0;i<7;i++){

printf("%s,%s,%d\n",p[i]->num,p[i]->name,p[i]->money);

}

}

int main()

{

//链表初始化

salary a3={"003","name003",3500,NULL},a2={"005","name005",2500,&a3},a1={"002","name002",3000,&a2};

salary b4={"007","name007",3600,NULL},b3={"001","name001",3000,&b4},b2={"004","name004",3700,&b3},b1={"006","name006",2800,&b2};

salary\*list1,\*list2,\*list,\*p[7];

list1=&a1;list2=&b1;

list=combin(list1,list2);

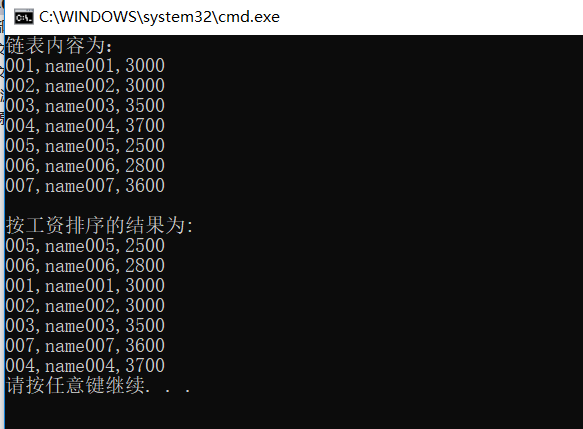
print(list);

rank(list,p);

print\_p(p);

return 0;

}



1. **链表解环**

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <time.h>

struct node

{

int num;

struct node \*next;

};

struct node \*create();

void openloop(struct node \*);

void print(struct node \*);

void main()

{

struct node \*head;

srand((unsigned)time(NULL));

head = create();

openloop(head);

print(head);

}

struct node \*create()

{

int n = 0;

struct node \*p1, \*p2, \*head;

int i;

int randomIndex ;

head=NULL;

p1=NULL;

p2=NULL;

while(n==0)//环链的长度随机

n=((int)rand())%20;

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

{

p1=(struct node\*)malloc(sizeof(struct node));

p1->next = NULL;

p1->num = ((int)rand());//每节点内容随机

if(i == 0)

head = p1;

else

p2->next = p1;

p2=p1;

}

//此时p1保留了尾节点的信息

if (head!=NULL)//当链表不为空时，随机成环

{

randomIndex =((int)rand())%n; //随机成环的节点

p2=head;

for(i=0; i<randomIndex; i++)

p2 = p2->next;//找到成环的节点

p1->next = p2;

}

return head;

}

void openloop(node\*head)

{

node\*p1,\*p2;

int count=0,i=0,j,\*add,flag=0;

p1=p2=head;

add=(int\*)malloc(20\*sizeof(int));

while(1){

flag=0;

\*(add+i)=(int)(p1);//动态创建数组存储链表每个节点的地址

for(j=0;j<i;j++){

if(\*(add+j)==\*(add+i)){flag=1;break;}

}

if(flag==1){

p2->next=NULL;

break;

}

else{

p2=p1;

p1=p1->next;

}

i++;

}

}

void print(node\*head)

{

node\*p1;

p1=head;

printf("当前链表内容为：\n");

while(p1!=NULL){

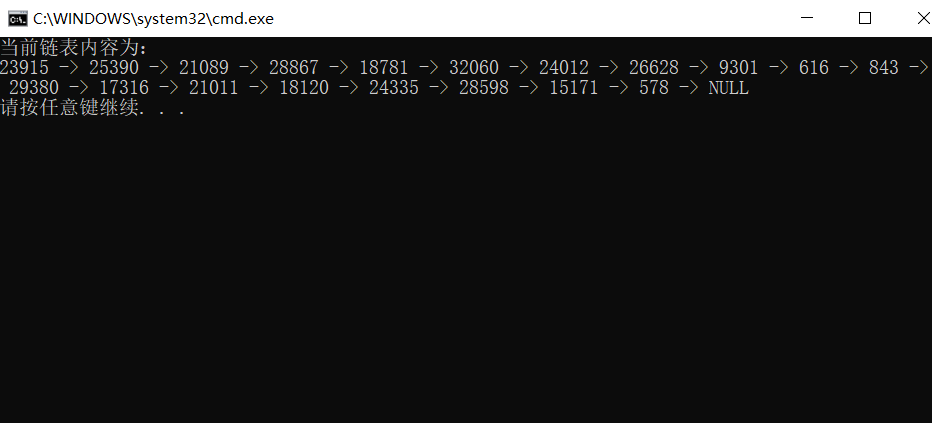
printf("%d -> ",p1->num);

p1=p1->next;

}

if(p1==NULL)printf("NULL\n");

}



1. **空闲块模拟管理**

#include<stdio.h>

#include<malloc.h>

#define Ceiling 20

struct storage{

int len;

long addr;

storage\*next;

};

storage\*insert(storage\*head,storage\*p0)//按长度递增顺序插入新的节点

{

storage\*p1,\*p2;

p1=p2=head;

if(head==NULL){head=p0;p0->next=NULL;}

else{

while(p0->len>p1->len&&p1->next!=NULL){

p2=p1;

p1=p1->next;

}

if(p0->len<=p1->len){

if(p1!=head){

p2->next=p0;

p0->next=p1;

}

else{

head=p0;

p0->next=p1;

}

}

else{

p1->next=p0;

p0->next=NULL;

}

}

return head;

}

storage\* Newenter(storage\*head)

{

int length,flag=0,count=0;

long add;

storage\*p,\*p1=head;

printf("输入长度：");

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

printf("输入起始地址：");

scanf\_s("%ld",&add);

while(p1!=NULL){

if(p1->addr==add){flag=1;break;}//判断该存储空间之前是否已经插入过

if(p1->len==length)count++;

if(count>Ceiling){flag=-1;break;}//判断该长度的空间是否超过Ceiling

p1=p1->next;

}

//对不符合条件情况的处理

if(flag==1)printf("您不能插入之前已插入的内存空间！\n");

else if(flag==-1)printf("您插入该长度存储块已经达到上限！\n");

else if(flag==0){

p=(storage\*)malloc(sizeof(storage));//创建新节点

p->len=length;p->addr=add;

head=insert(head,p);

}

return head;

}

storage\*Mymalloc(storage\*\*head)

{

int len;

storage\*p1=\*head,\*p2;

printf("输入长度：");

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

while(len>p1->len&&p1->next!=NULL){

p2=p1;

p1=p1->next;

}//遍历寻找相应节点

if(len<=p1->len){

printf("\n申请成功！起始地址=%d,长度=%d KBytes\n",p1->addr,p1->len);

if(\*head==p1)\*head=p1->next;//将该节点删除

else p2->next=p1->next;

p1->next=NULL;

}else{

printf("\n申请失败!\n");

return NULL;

}

return p1;

}

storage\*Myfree(storage\*head,storage\*p)

{

if(p!=NULL){

printf("\n释放成功！\n");

return insert(head,p);//重新插入上一次删除的节点

}

else return head;

}

void List(storage\*head)

{

storage\*p1,\*p2;

p1=p2=head;

printf("\n当前空闲块表列：\n长度 起始地址\n");

while(p1!=NULL){//遍历输出节点信息

if(p1->len==p2->len&&p1!=head){//长度相同时不重复输出长度

printf(" %-10d\n",p1->addr);

}else{

printf("%-8d%-10d\n",p1->len,p1->addr);

}

p2=p1;

p1=p1->next;

}

putchar(10);

}

void operate(storage\*head)

{

char ch;

int flag=-1,t=0;

storage\*tmp;

printf("E<nter> or I<nsert> or M<alloc> or F<ree> or Q<uit>\n");

if((ch=getchar())=='E'||ch=='e'){t=1;getchar();}//按e/E开始

while(t){

printf("E<nter> or I<nsert> or M<alloc> or F<ree> or Q<uit>\n");

ch=getchar();

if(ch=='q'||ch=='Q')break;

else{

switch(ch){

case'i':case'I':{

head=Newenter(head);

List(head);

break;

}

case'M':case'm':{

tmp=Mymalloc(&head);//tmp暂时存储了被删除节点的信息

flag=1;//已经申请过置1

List(head);

break;

}

case'F':case'f':{

if(tmp!=NULL&&flag==1){

head=Myfree(head,tmp);

flag=0;//已经释放过置0

List(head);

}

//对不合理的情况的处理

else if(flag==0)printf("不能连续两次free同一块内存空间！\n");

else if(flag==-1)printf("您还没有申请过内存空间！\n");

break;

}

default:printf("Your input cannot be recognized!Please enter again!\n");

}

}

fflush(stdin);

}

}

int main()

{

storage\*head=NULL,\*p;

operate(head);

p=head;

while(p!=NULL){//释放链表

head=head->next;

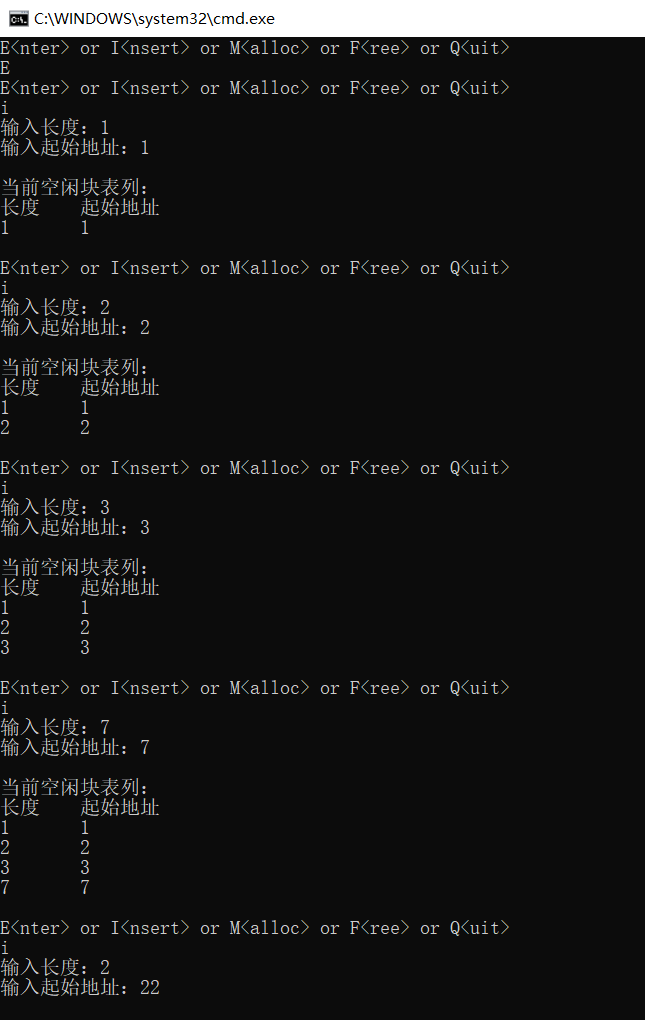
free(p);

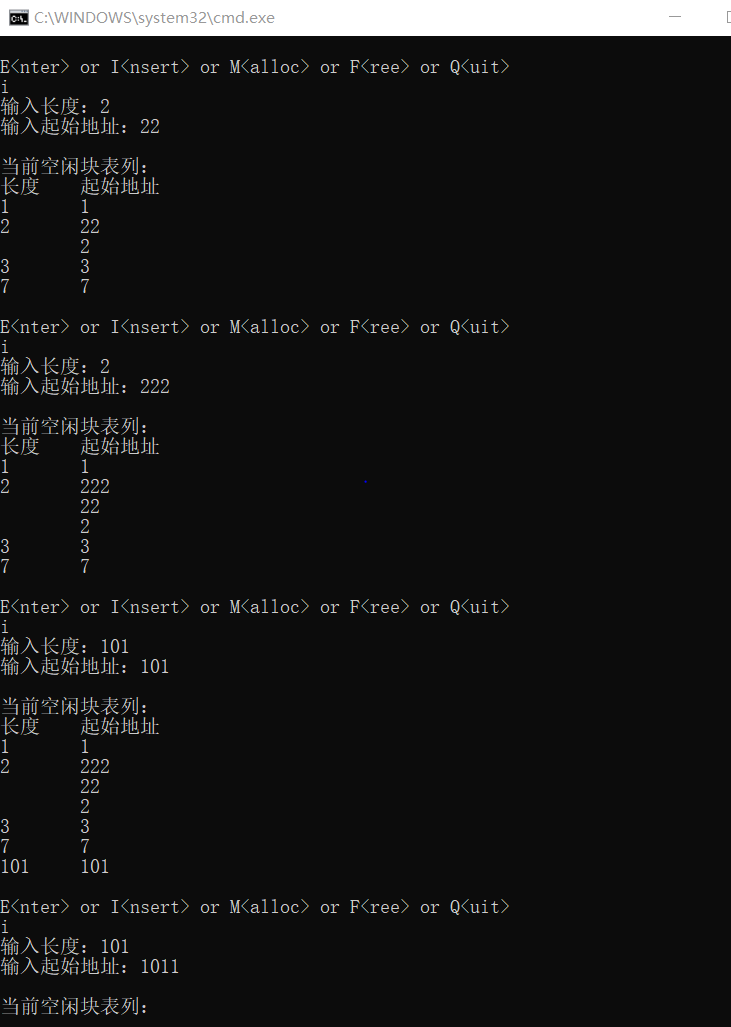
p=head;

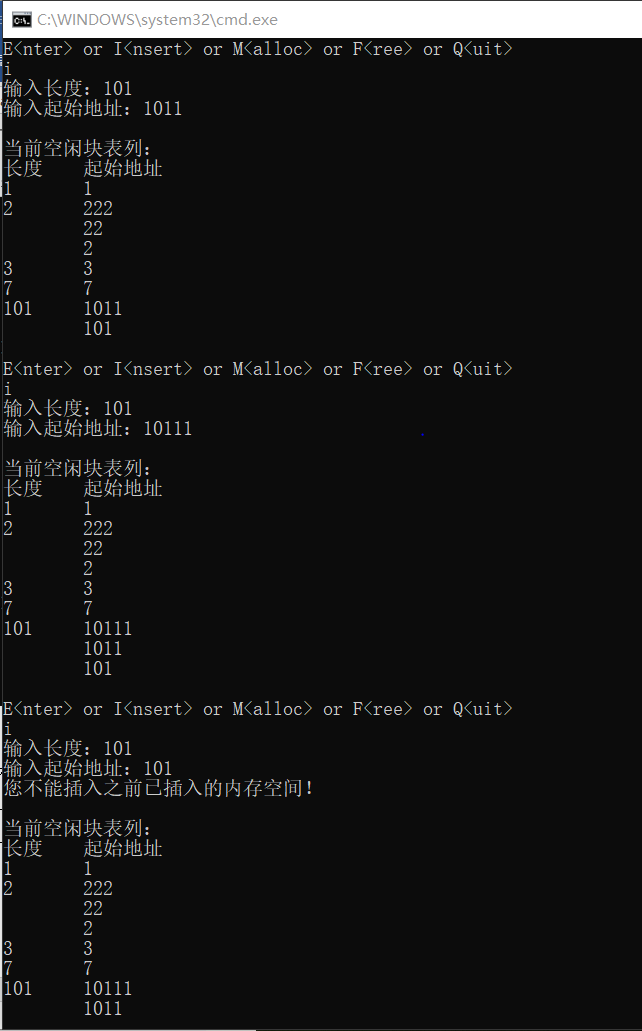
}

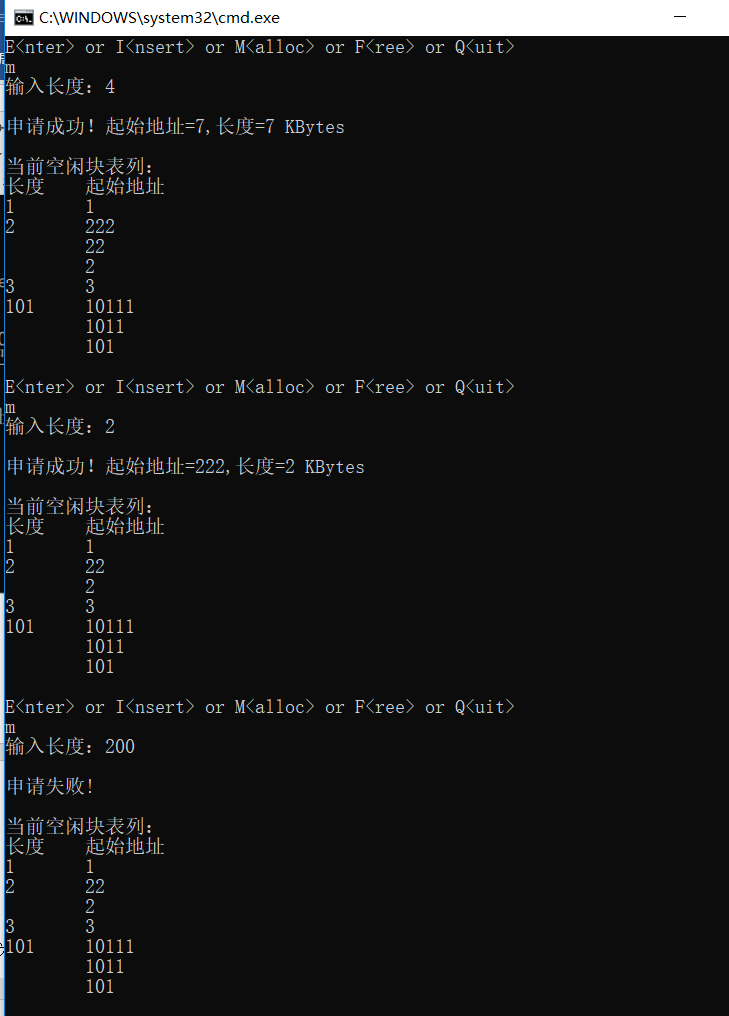
return 0;

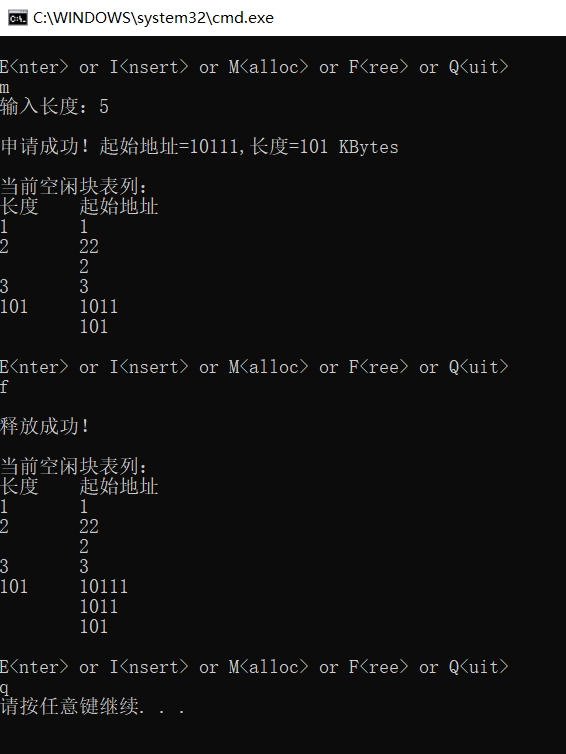
}











1. **（选作）循环单链表**

//循环链表

#include<stdio.h>

#include<malloc.h>

#include<string.h>

struct student{

char num[20];

char name[10];

student\*next;

};

student\*insert(student\*head,student\*stu)//递增插入新节点，创建循环链表

{

student\*p0,\*p1,\*p2,\*last;

p0=stu;p1=head;

while(head!=NULL&&p1->next!=head){

p2=p1;p1=p1->next;

}

last=p1;//找到表尾

p1=head;

if(head==NULL){head=p0;p0->next=head;}

else{

while(strcmp(p0->num,p1->num)>0&&(p1->next!=head)){

p2=p1;

p1=p1->next;

}

if(strcmp(p0->num,p1->num)<=0){

if(head==p1){//创建新表头后和原表尾连接

head=p0;

last->next=head;

}

else p2->next=p0;

p0->next=p1;

}

else{

p1->next=p0;

p0->next=head;

}

}

return head;

}

student\*creat()

{

void list(student\*head);

student\*head,\*p1;int count=0,n;

printf("please input the number of students:");

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

printf("please input the information of students:\n");

head=NULL;

do{

p1=(student\*)malloc(sizeof(student));

scanf\_s("%s%s",p1->num,20,p1->name,10);

head=insert(head,p1);

list(head);//每次插入后输出当前链表

count++;

}while(count<n);

return head;

}

int f(student\*head)//节点计数

{

student\*p1;int count=0;

p1=head;

while(p1->next!=head){

p1=p1->next;

count++;

}

count++;

return count;

}

student\*fv(student\*head)//环形链表倒序

{

student\*p1,\*p2,\*p3;

p1=head;

p2=p3=head->next;

while(p2!=head){

p3=p2->next;

p2->next=p1;

p1=p2;

p2=p3;

}

p2->next=p1;

return p1;

}

void list(student\*head)//输出链表数据

{

student\*p1;

int count=1;

p1=head;

printf("当前链表:\n head -> ");

do{

if(count%4==0)putchar(10);

printf("number:%s name：%s -> ",p1->num,p1->name);

count++;

p1=p1->next;

}while(p1!=head);

printf("head\n");

}

student\*search(student\*head,char\*num)//查找信息

{

student\*p1;

p1=head;

do{

if(strcmp(num,p1->num)==0)return p1;

p1=p1->next;

}while(p1!=head);

return NULL;

}

void myfree(student\*head)//释放内存

{

student\*p,\*tmp;

p=head->next;

head->next=NULL;//先解环后释放

while(p!=NULL){

tmp=p;

p=p->next;

free(tmp);

}

}

int main()

{

student\*head,\*result;

char num[20];

head=creat();

if(head==NULL)printf("空链表！\n");

else{

printf("\n节点个数：%d\n",f(head));

head=fv(head);

printf("\n倒序后的链表：\n");

list(head);

printf("\nplease input the num you wanna search:");

scanf\_s("%s",num,20);

if((result=search(head,num))!=NULL)

printf("学号:%s,姓名:%s\n",result->num,result->name);

else printf("该学号不存在！\n");

}

myfree(head);

return 0;

}

