**《算法与数据结构》**

**实 验 指 导 手 册**

# 北京邮电大学软件学院

# 2020-2021学年第1学期实验报告

**课程名称： 算法与数据结构**

**实验名称： 实验一 线性表**

**实验完成人：**

**姓名：**\_\_\_王宇涵\_\_\_\_\_**学号：**\_\_\_\_2020211730\_\_\_\_**成绩：**\_\_\_\_\_\_\_\_

**指导教师：**\_\_\_\_\_\_\_**\_\_贾红娓** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**日 期： 2021 年 10 月 8 日**

1. **实验目的**

（1）本次实验的主要目的在于熟悉线性表的基本运算在两种存储结构上的实现，其中以熟悉各种链表的操作为侧重点。

（2）熟悉线性表在软件工程项目中数据存储和处理中的运用。

1. **实验内容**
2. **城市链表**

**[问题描述]**

将若干城市的信息，存入一个带头结点的单链表。结点中的城市信息包括：城市名，城

市的位置坐标。要求能够利用城市名和位置坐标进行有关查找、插入、删除、更新等操作。

**[基本要求]**

　　（1） 给定一个城市名，返回其位置坐标；

　　（2） 给定一个位置坐标P和一个距离D，返回所有与P的距离小于等于D的城市。

**[测试数据]**

由学生依据软件工程的测试技术自己确定。注意测试边界数据。

1. **约瑟夫环**

**[问题描述]**

约瑟夫（Joeph）问题的一种描述是：编号为1,2,…,n的n个人按顺时针方向围坐一圈，

每人持有一个密码（正整数）。一开始任选一个正整数作为报数上限值m，从第一个人开始按顺时针方向自1开始顺序报数，报到m时停止报数。报m的人出列，将他的密码作为新的m值，从他在顺时针方向上的下一个人开始重新从1报数，如此下去，直至所有人全部出列为止。试设计一个程序求出出列顺序。

**[基本要求]**

利用单向循环链表存储结构模拟此过程，按照出列的顺序印出各人的编号。

**[测试数据]**

　　m的初值为20；密码：3，1，7，2，4，8，4（正确的结果应为6，1，4，7，2，3，5）。

**[实现提示]**

程序运行后首先要求用户指定初始报数上限值，然后读取各人的密码。设n≤30。

3**选做内容**

1. 向上述程序中添加在顺序结构上实现的部分。
2. **实验环境**

Dev C++

1. **实验过程和实验结果**
2. **城市链表**

**1.1问题分析**

1）将若干城市的信息，存入一个带头结点的单链表，结点中的城市信息包括：城市名，城市的位置坐标。所以要先创建一个结构体lnode,其中包含指向下一个结点的指针，该城市的坐标x,y，以及城市名

2）能够利用城市名和位置坐标进行有关查找、插入、删除、更新等操作。

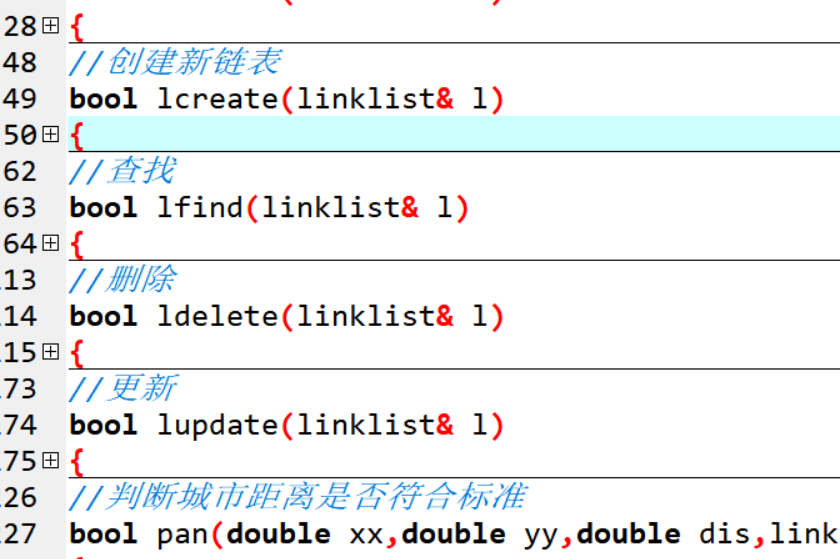
所以每种操作写一个函数。

3）给定一个城市名，返回其位置坐标。属于查找操作。

4）给定一个位置坐标P和一个距离D，返回所有与P的距离小于等于D的城市。对于此操作再写一个函数

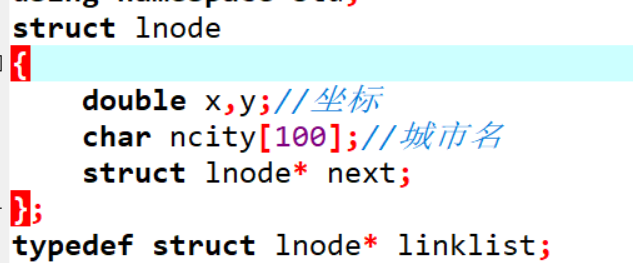
**1.2设计方案**

1. 用结构体先定义lnode其中包含坐标与城市名，其中还有指向下一节点的指针
2. 在创建链表前要先初始化，并实现一个创建链表的函数，来创造城市链表并保存所有的城市名称及其位置
3. 创造链表可以采用插入函数实现，用尾插法插入链表最后
4. 查找函数通过坐标和城市名查找两种方式，通过坐标查找城市，或者通过城市查找坐标
5. 删除同理，可以通过城市名删除该程式，也可以通过坐标数删除
6. 通过距离和坐标寻找城市函数，使用一个判断函数判断距离是否符合要求，然后将该函数的值传回计算距离函数



**1.3算法**

1）用lnode结构体来储存城市的信息，利用char ncity[100];来表示表示城市名，double x,y;用来储存城市的信息的坐标。



2）//创建新链表

bool linit(linklist& l)

{

l=(linklist)malloc(sizeof(struct lnode));//申请新节点

if(l==NULL)return false;//申请成功

l->next=NULL;

return true;

}

3）//插入链表

void linsert(linklist& l)

{

lnode \*prear,\*p;

p=(linklist)malloc(sizeof(struct lnode));

cout<<"城市名"<<endl;

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

//cout<<p->ncity;

cout<<"城市坐标"<<endl;

cin>>p->x>>p->y;

prear=l; //l指向头结点，要先标记

//尾插法

while(l->next!=NULL)

{

l=l->next;

} //向后查找找到最后一个节点

l->next=p;

p->next=NULL;

l=prear;

//cout<<l->next->x<<endl;

//cout<<p->x<<p->y;

}

//创建新链表

bool lcreate(linklist& l)

{

int k;

cout<<"输入创建城市数："<<endl;

cin>>k;

while(k)

{

linsert(l);//插入节点

k--;

//cout<<k<<endl;

}

return true;

}

4）

//查找

bool lfind(linklist& l)

{

int cha;

double xx,yy;

char ch[20];

linklist l1=l;

cout<<"1查城市"<<endl;

cout<<"2查坐标"<<endl;

cin>>cha;

if(cha==1)//通过坐标查找

{

cout<<"输入坐标"<<endl;

cin>>xx>>yy;

l=l->next;

//cout<<l->x<<l->y<<endl;

while((l!=NULL)&&!((l->x==xx)&&(l->y==yy)))

{

l=l->next;

}

//cout<<l->x<<l->y<<endl;

if(l==NULL)//没找到该城市

{

cout<<"不存在此城市"<<endl;

}

else

{

cout<<l->ncity<<endl;

}

}

else if(cha==2)//通过城市查找

{

cout<<"输入城市名"<<endl;

cin>>ch;

while(l!=NULL&&(strcmp(l->ncity,ch)!=0))//strcmp比较两个字符串

{

l=l->next;

}

if(l==NULL)

{

cout<<"不存在此城市"<<endl;

}

else

{

cout<<l->x<<" , "<<l->y<<endl;

}

}

l=l1;//要回到头结点方便以后的操作

return true;

}

5）

//更新

bool lupdate(linklist& l)

{

int cha;

double xx,yy;

char ch[20];

linklist l1=l;

cout<<"1通过坐标"<<endl;

cout<<"2通过城市"<<endl;

cin>>cha;

if(cha==1)//通过坐标

{

cout<<"输入坐标"<<endl;

cin>>xx>>yy;

//l=l->next;

//cout<<l->x<<l->y<<endl;

while(l->next!=NULL&&!((l->next->x==xx)&&(l->next->y==yy)))

{

l=l->next;

//cout<<l->x<<l->y<<endl;

}

//cout<<l->x<<l->y<<endl;

if(l->next==NULL)

{

cout<<"不存在此城市"<<endl;

}

else

{

cout<<"输入新名字"<<endl;

cin>>l->next->ncity;

}

}

else if(cha==2)//通过城市

{

cout<<"输入城市名"<<endl;

cin>>ch;

while(l->next!=NULL&&(strcmp(l->next->ncity,ch)!=0))

{

l=l->next;

}

if(l->next==NULL)

{

cout<<"不存在此城市"<<endl;

}

else

{

cout<<"输入新坐标"<<endl;

cin>>l->next->x>>l->next->y;

}

}

l=l1;

return true;

}

1. //判断城市距离是否符合标准

//直接用平方判断，避免了根号的精度问题

bool pan(double xx,double yy,double dis,linklist& l)

{

double dx=(l->x)-xx;

double dy=(l->y)-yy;

//cout<<dx<<" "<<dy<<endl;

if(((dx\*dx+dy\*dy)<=dis\*dis)&&(!((dx==0)&&(dy==0))))

//中心城市本身也在判定范围内

{

// cout<<l->x<<l->y<<endl;

return true;

}

else

{

return false;

}

}

//计算

bool lcompute(linklist& l)

{

double xx,yy;

double dis;

linklist l1=l;

cout<<"输入坐标"<<endl;

cin>>xx>>yy;

cout<<"输入距离"<<endl;

cin>>dis;

l=l->next;

while(l!=NULL)

{

if(pan(xx,yy,dis,l)==true)

{

cout<<l->ncity<<endl;

cout<<l->x<<","<<l->y<<endl;

}

l=l->next;

}

//cout<<l->x<<l->y<<endl;

l=l1;

return true;

}

1. 主函数

int main()

{

while(1)//循环操作，可以操作无数次

{

cout<<"此输入想要的操作（第一次请选择1）"<<endl;

cout<<"1创建"<<endl;

cout<<"2查询"<<endl;

cout<<"3插入"<<endl;

cout<<"4删除"<<endl;

cout<<"5更新"<<endl;

cout<<"6计算所有城市"<<endl;

cout<<"7退出"<<endl;

cin>>caozuo;//输入想要的操作

if(caozuo==7)//选择退出

{

break;

}

else

{

switch(caozuo)

{

case 1:

linit(la);

lcreate(la);

break;

case 2:

lfind(la);

break;

case 3:

linsert(la);

break;

case 4:

ldelete(la);

break;

case 5:

lupdate(la);

break;

case 6:

lcompute(la);

break;

default:

break;

}

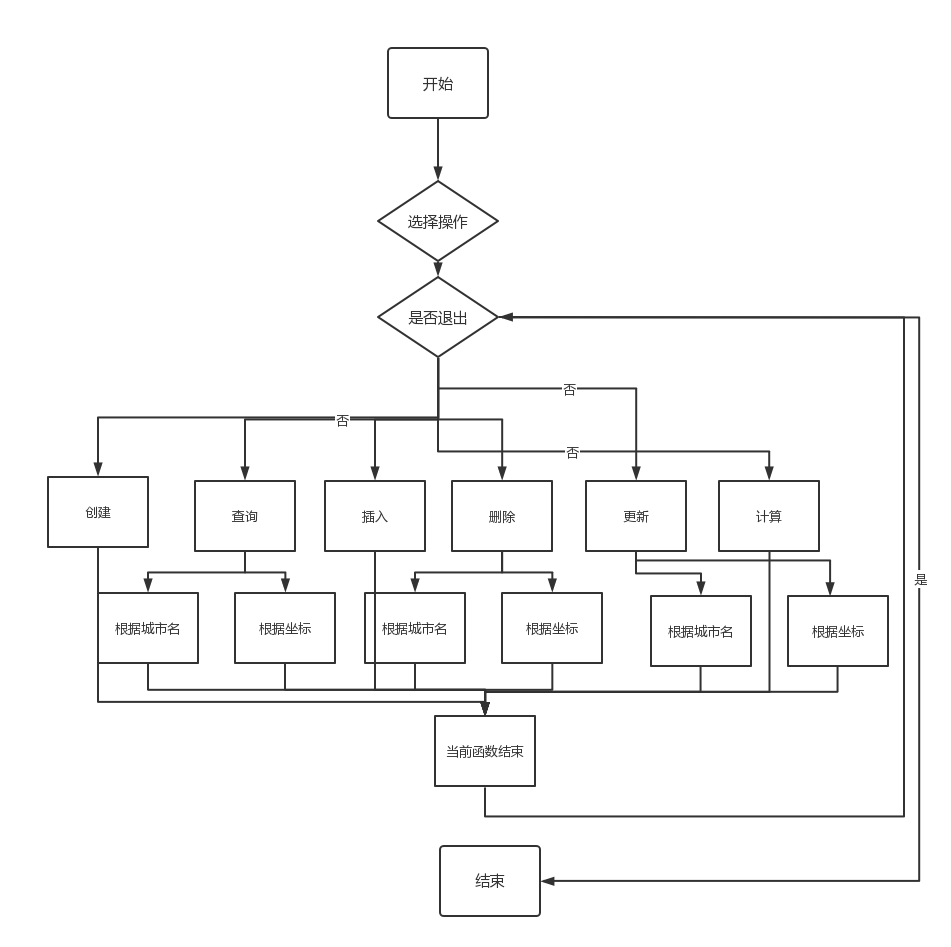
}

}

return 0;

}

**1.4设计图**



**1.5程序**

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<iostream>

#include<cmath>

using namespace std;

//结构体定义

struct lnode

{

double x,y;//坐标

char ncity[100];//城市名

struct lnode\* next;

};

typedef struct lnode\* linklist;

int caozuo;//选择操作类型

linklist la,lb,lc;

//创建新链表

bool linit(linklist& l)

{

l=(linklist)malloc(sizeof(struct lnode));//申请新节点

if(l==NULL)return false;//申请成功

l->next=NULL;

return true;

}

//插入链表

void linsert(linklist& l)

{

lnode \*prear,\*p;

p=(linklist)malloc(sizeof(struct lnode));

cout<<"城市名"<<endl;

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

//cout<<p->ncity;

cout<<"城市坐标"<<endl;

cin>>p->x>>p->y;

prear=l; //l指向头结点，要先标记

//尾插法

while(l->next!=NULL)

{

l=l->next;

} //向后查找找到最后一个节点

l->next=p;

p->next=NULL;

l=prear;

//cout<<l->next->x<<endl;

//cout<<p->x<<p->y;

}

//创建新链表

bool lcreate(linklist& l)

{

int k;

cout<<"输入创建城市数："<<endl;

cin>>k;

while(k)

{

linsert(l);//插入节点

k--;

//cout<<k<<endl;

}

return true;

}

//查找

bool lfind(linklist& l)

{

int cha;

double xx,yy;

char ch[20];

linklist l1=l;

cout<<"1查城市"<<endl;

cout<<"2查坐标"<<endl;

cin>>cha;

if(cha==1)

{

cout<<"输入坐标"<<endl;

cin>>xx>>yy;

l=l->next;

//cout<<l->x<<l->y<<endl;

while((l!=NULL)&&!((l->x==xx)&&(l->y==yy)))

{

l=l->next;

}

//cout<<l->x<<l->y<<endl;

if(l==NULL)//没找到该城市

{

cout<<"不存在此城市"<<endl;

}

else

{

cout<<l->ncity<<endl;

}

}

else if(cha==2)

{

cout<<"输入城市名"<<endl;

cin>>ch;

while(l!=NULL&&(strcmp(l->ncity,ch)!=0))//strcmp比较两个字符串

{

l=l->next;

}

if(l==NULL)

{

cout<<"不存在此城市"<<endl;

}

else

{

cout<<l->x<<" , "<<l->y<<endl;

}

}

l=l1;//要回到头结点方便以后的操作

return true;

}

//删除

bool ldelete(linklist& l)

{

int cha;

double xx,yy;

char ch[20];

linklist l1=l;

cout<<"1通过坐标"<<endl;

cout<<"2通过城市"<<endl;

cin>>cha;

if(cha==1)//通过坐标删除

{

cout<<"输入坐标"<<endl;

cin>>xx>>yy;

//l=l->next;

//cout<<l->x<<l->y<<endl;

while(l->next!=NULL&&!((l->next->x==xx)&&(l->next->y==yy)))

{

l=l->next;

//cout<<l->x<<l->y<<endl;

}

//cout<<l->x<<l->y<<endl;

if(l->next==NULL)

{

cout<<"不存在此城市"<<endl;

}

else

{

lnode\*p;

p=l->next;

cout<<p->ncity<<endl;

l->next=p->next;

free(p);

}

}

else if(cha==2)//通过名称删除

{

cout<<"输入城市名"<<endl;

cin>>ch;

while(l->next!=NULL&&(strcmp(l->next->ncity,ch)!=0))

{

l=l->next;

}

if(l->next==NULL)

{

cout<<"不存在此城市"<<endl;

}

else

{

lnode\*p;

p=l->next;

cout<<l->next->x<<" , "<<l->next->y<<endl;

l->next=p->next;

free(p);

}

}

l=l1;

return true;

}

//更新

bool lupdate(linklist& l)

{

int cha;

double xx,yy;

char ch[20];

linklist l1=l;

cout<<"1通过坐标"<<endl;

cout<<"2通过城市"<<endl;

cin>>cha;

if(cha==1)//通过坐标

{

cout<<"输入坐标"<<endl;

cin>>xx>>yy;

//l=l->next;

//cout<<l->x<<l->y<<endl;

while(l->next!=NULL&&!((l->next->x==xx)&&(l->next->y==yy)))

{

l=l->next;

//cout<<l->x<<l->y<<endl;

}

//cout<<l->x<<l->y<<endl;

if(l->next==NULL)

{

cout<<"不存在此城市"<<endl;

}

else

{

cout<<"输入新名字"<<endl;

cin>>l->next->ncity;

}

}

else if(cha==2)//通过城市

{

cout<<"输入城市名"<<endl;

cin>>ch;

while(l->next!=NULL&&(strcmp(l->next->ncity,ch)!=0))

{

l=l->next;

}

if(l->next==NULL)

{

cout<<"不存在此城市"<<endl;

}

else

{

cout<<"输入新坐标"<<endl;

cin>>l->next->x>>l->next->y;

}

}

l=l1;

return true;

}

//判断城市距离是否符合标准

bool pan(double xx,double yy,double dis,linklist& l)

{

double dx=(l->x)-xx;

double dy=(l->y)-yy;

//cout<<dx<<" "<<dy<<endl;

if(((dx\*dx+dy\*dy)<=dis\*dis)&&(!((dx==0)&&(dy==0))))

{

// cout<<l->x<<l->y<<endl;

return true;

}

else

{

return false;

}

}

//计算

bool lcompute(linklist& l)

{

double xx,yy;

double dis;

linklist l1=l;

cout<<"输入坐标"<<endl;

cin>>xx>>yy;

cout<<"输入距离"<<endl;

cin>>dis;

l=l->next;

while(l!=NULL)

{

if(pan(xx,yy,dis,l)==true)

{

cout<<l->ncity<<endl;

cout<<l->x<<","<<l->y<<endl;

}

l=l->next;

}

//cout<<l->x<<l->y<<endl;

l=l1;

return true;

}

int main()

{

while(1)

{

cout<<"此输入想要的操作（第一次请选择1）"<<endl;

cout<<"1创建"<<endl;

cout<<"2查询"<<endl;

cout<<"3插入"<<endl;

cout<<"4删除"<<endl;

cout<<"5更新"<<endl;

cout<<"6计算所有城市"<<endl;

cout<<"7退出"<<endl;

cin>>caozuo;

if(caozuo==7)

{

break;

}

else

{

switch(caozuo)

{

case 1:

linit(la);

lcreate(la);

break;

case 2:

lfind(la);

break;

case 3:

linsert(la);

break;

case 4:

ldelete(la);

break;

case 5:

lupdate(la);

break;

case 6:

lcompute(la);

break;

default:

break;

}

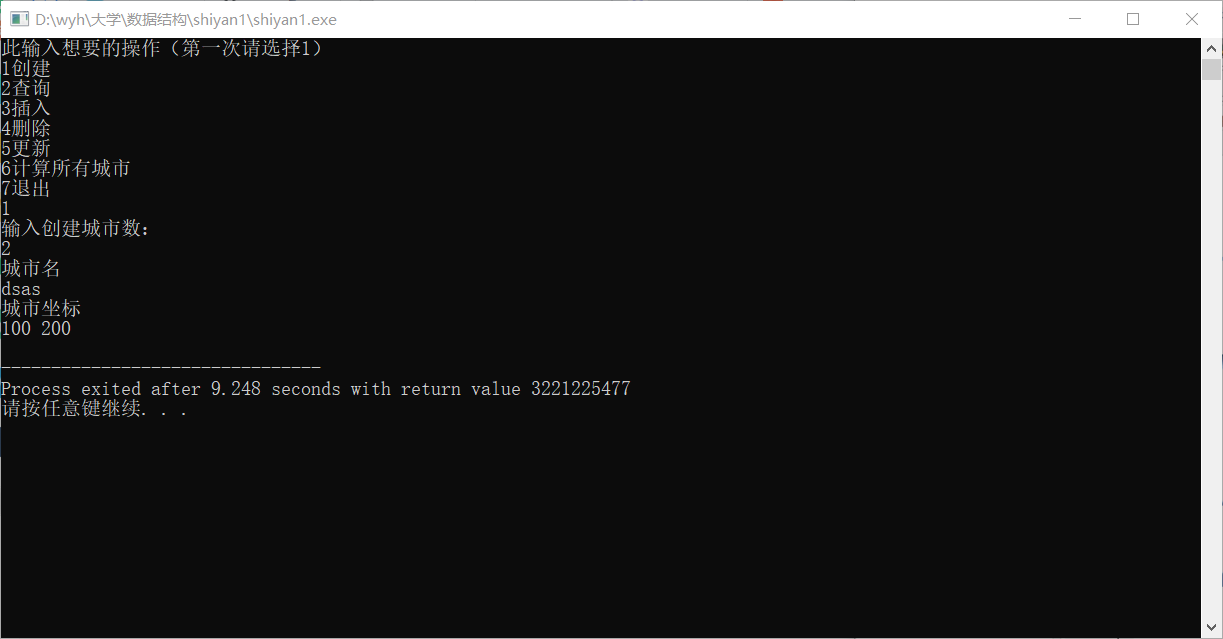
}

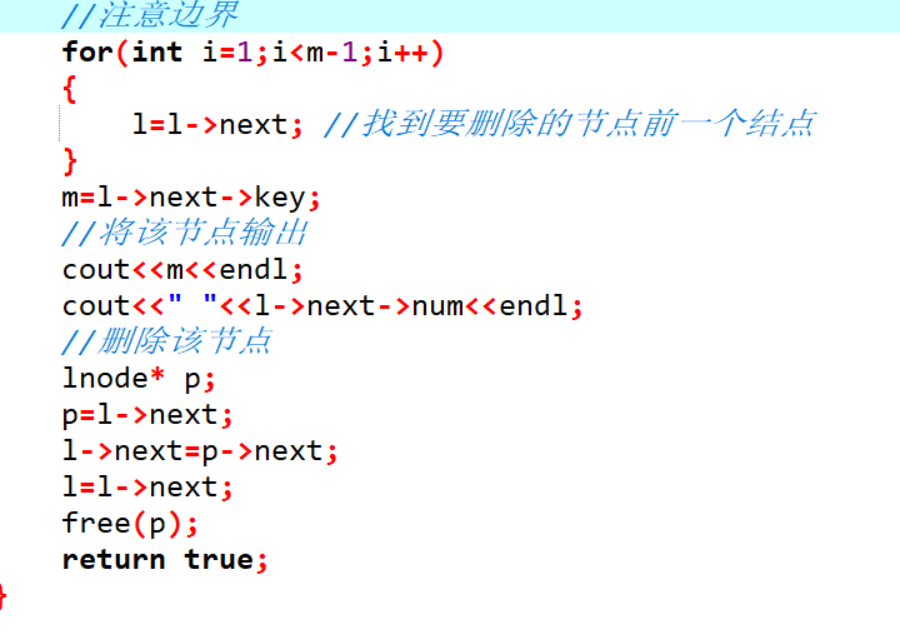
}

return 0;

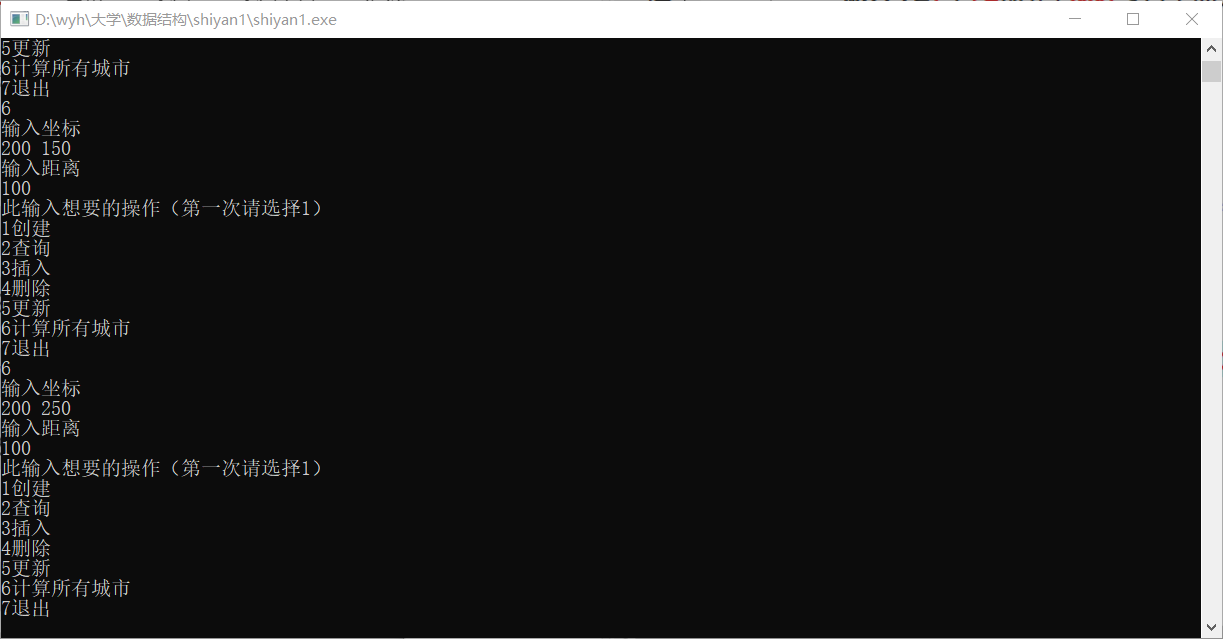
}

**1.6调试过程截图**

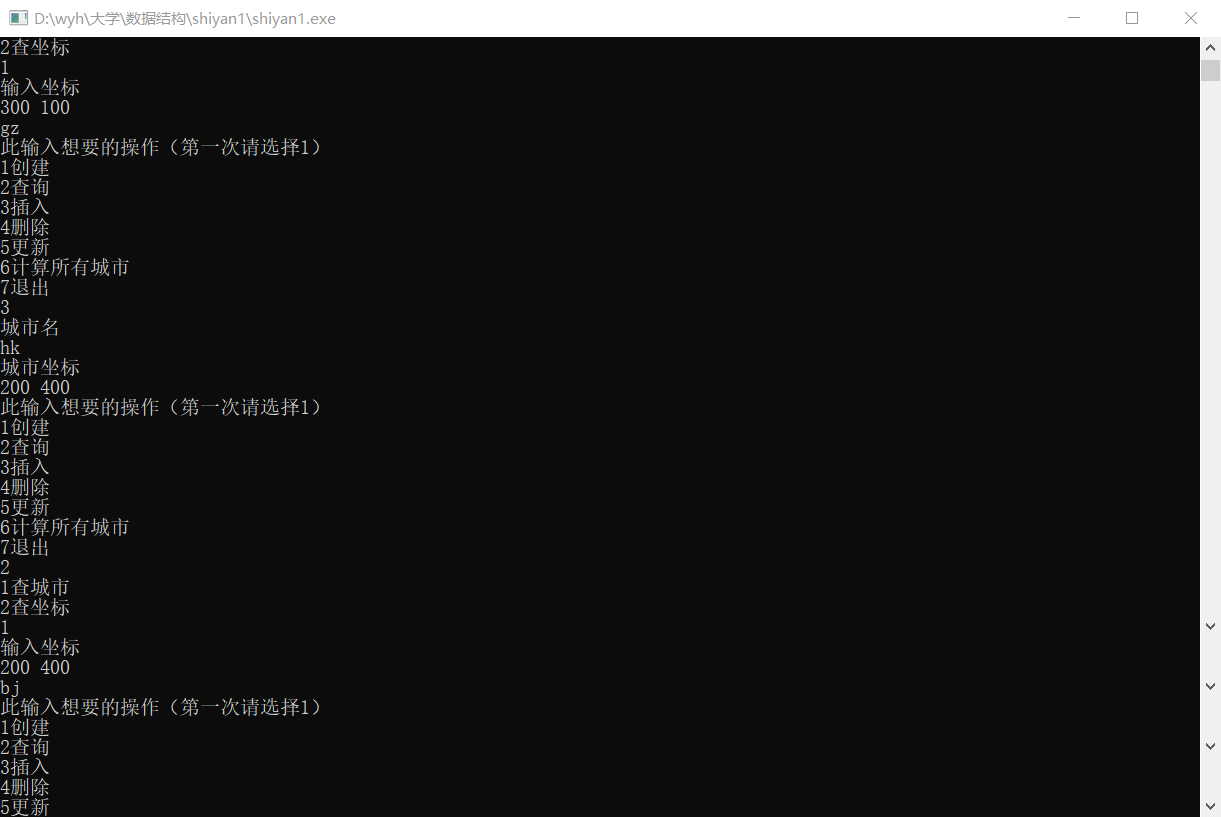
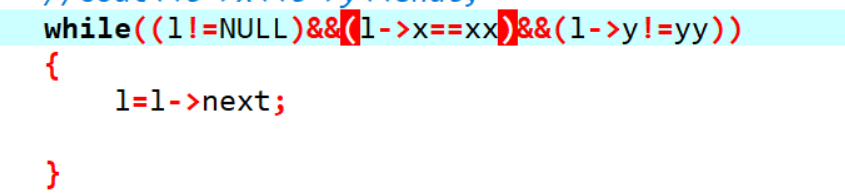
****

创建的问题：只输入了一个就结束程序了

删除的问题：本来想找到前一个结点，删除本来的节点，但是这样会漏掉一些节点没法删，所以得将后一个节点迁移



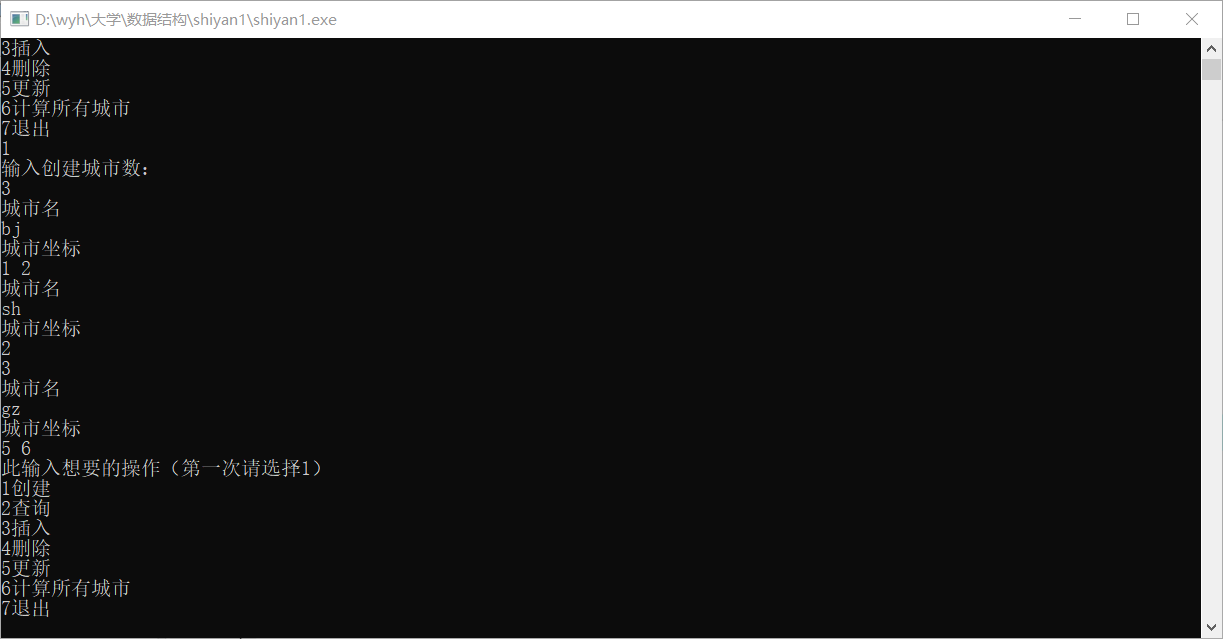
计算的问题：没有输出应有的城市



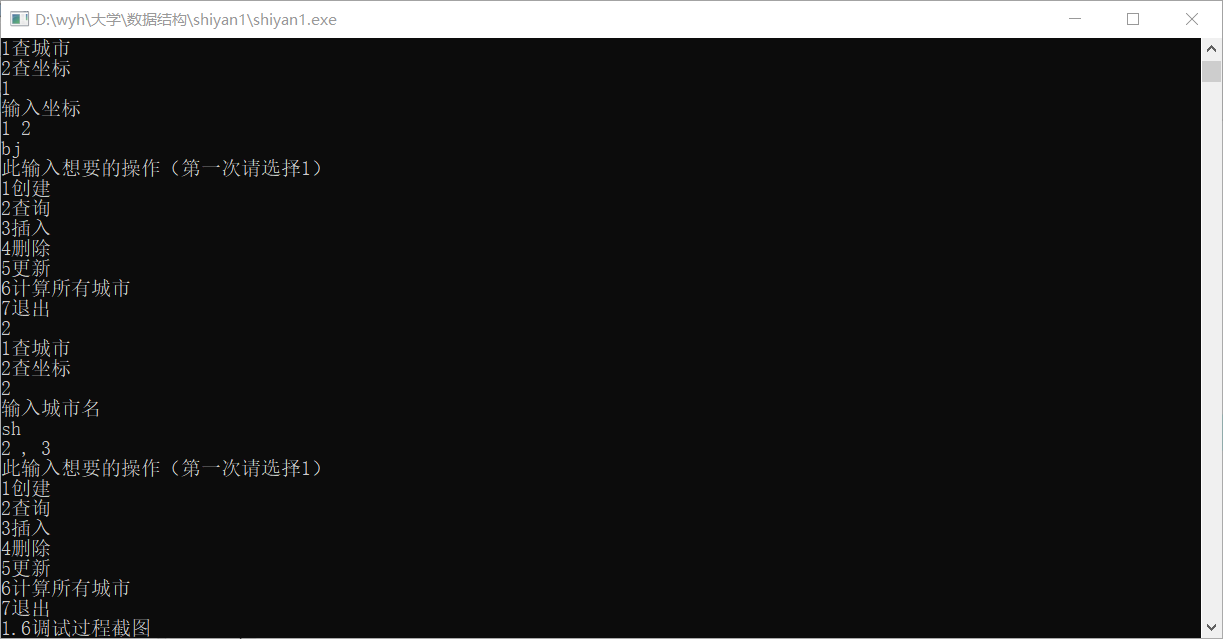
查找城市有误：是逻辑判断写错了

**1.7结果截图**

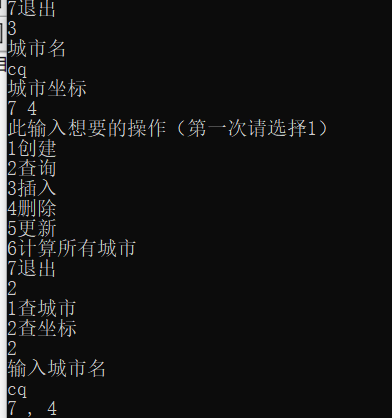
1. 创建



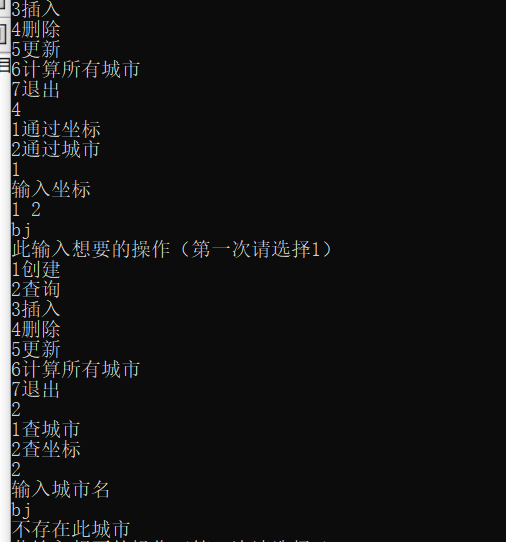
1. 查询



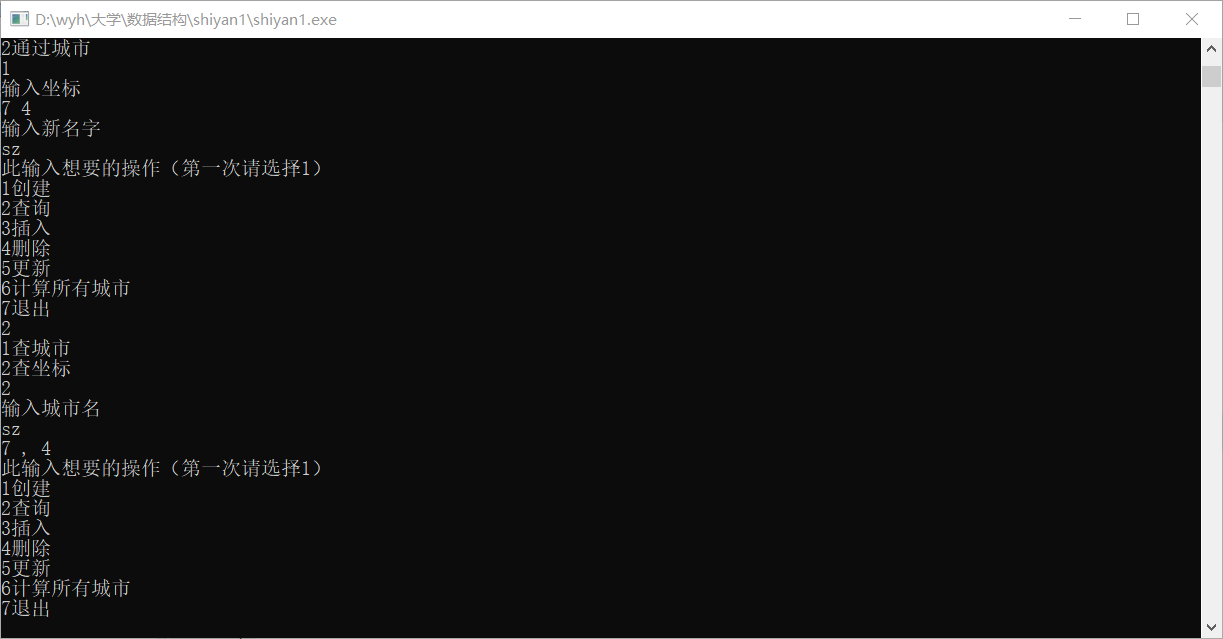
1. 插入



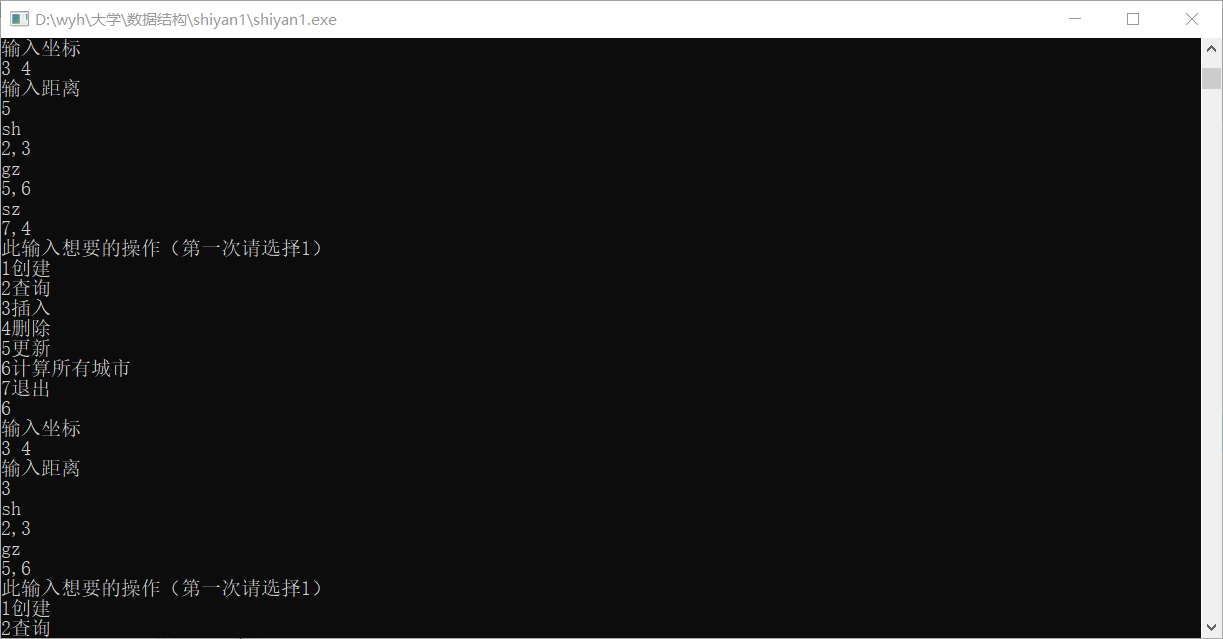
1. 删除



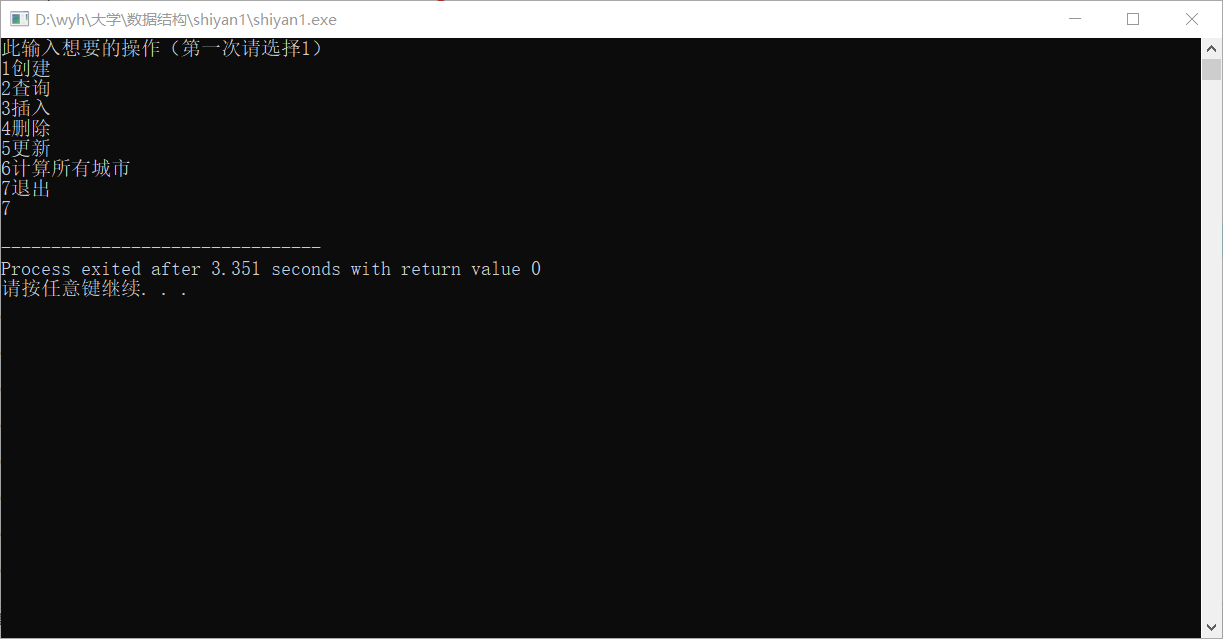
1. 更新



1. 计算



1. 退出



**2.约瑟夫环**

**2.1问题分析**

要使用循环链表查找结点，先创建一个结构体lnode,其中包含指向下一个结点的指针。编号为1,2,…,n的n个人按顺时针方向围坐一圈，每人持有一个密码（正整数）。人的编号，以及密码，注意要将尾节点指向头结点。找到每次报数的同学并更新m的值。

**2.2设计方案**

1. 一个主函数，一个删除函数
2. 在主函数中创建好循环链表，调用删除函数让同学出列
3. 删除函数中找到报数同学，循环m次，找到该同学后输出
4. 输出同学序号，更新m

**2.3算法**

1）//创建新链表

l=(linklist)malloc(sizeof(struct lnode));

cin>>l->key;

l->num=1;

la=l;//标记头结点

l->next=NULL;

for(int i=2;i<=renshu;i++)

{

lnode\* q=(linklist)malloc(sizeof(struct lnode));

l->next=q;

q->num=i;

cin>>q->key;

l=q;

}

l->next=la;

l=la;//形成循环列表

1. 循环调用删除

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

//调用人数的次数 ，表示处理完了

{

ldelete(l);

}

3）删除

bool ldelete(linklist& l)

{

//注意边界

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

{

l=l->next; //找到要删除的节点前一个结点

}

m=l->key;

//将该节点输出

//cout<<m<<;

cout<<l->num<<" ";

//删除该节点(因为如果只找到m前一个节点删除当前结点的话m=1会出问题，只能将后面一个节点填到前一个再删除后一个结点)

lnode\* p;

p=l->next;

l->num=p->num;

l->key=p->key;

l->next=p->next;

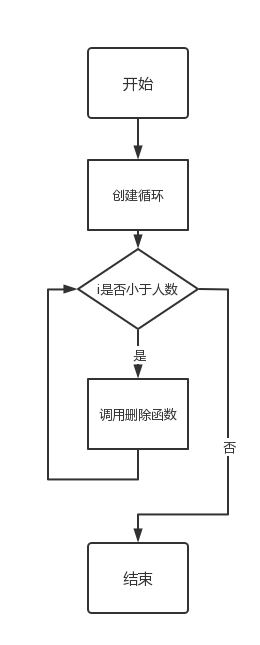
free(p);

//l=l->next;

return true;

}

**2.4设计图**

****

**2.5程序**

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<iostream>

#include<cmath>

using namespace std;

struct lnode

{

int key;//密码

int num;//编号

struct lnode\* next;

};

typedef struct lnode\* linklist;

int m;//出列值

int renshu;//测试人数

linklist l,la;

bool ldelete(linklist& l)

{

//注意边界

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

{

l=l->next; //找到要删除的节点前一个结点

}

m=l->key;

//将该节点输出

//cout<<m<<;

cout<<l->num<<" ";

//删除该节点

lnode\* p;

p=l->next;

//将后一个节点的值移到前一个结点

l->num=p->num;

l->key=p->key;

l->next=p->next;

free(p);//删除后一个节点

//l=l->next;

return true;

}

int main()

{

cin>>m;

cin>>renshu;

//创建新链表

l=(linklist)malloc(sizeof(struct lnode));

cin>>l->key;

l->num=1;

la=l;//标记头结点

l->next=NULL;

for(int i=2;i<=renshu;i++)

{

lnode\* q=(linklist)malloc(sizeof(struct lnode));

l->next=q;

q->num=i;

cin>>q->key;

l=q;

}

l->next=la;

l=la;//形成循环列表

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

//调用人数的次数 ，表示处理完了

{

ldelete(l);

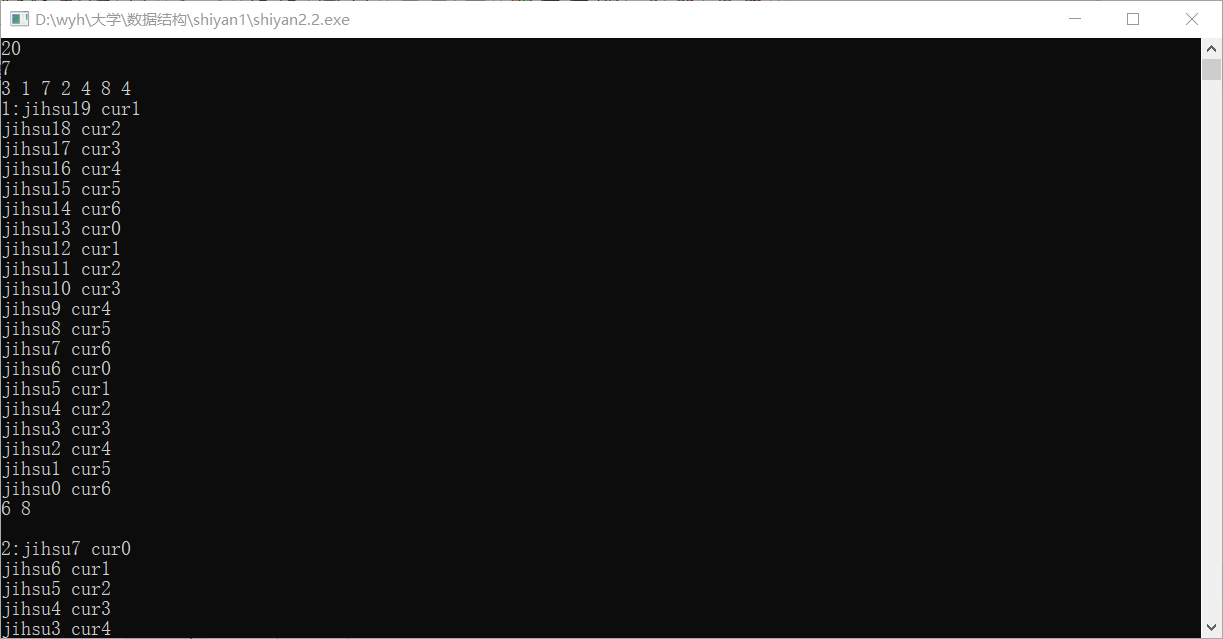
}

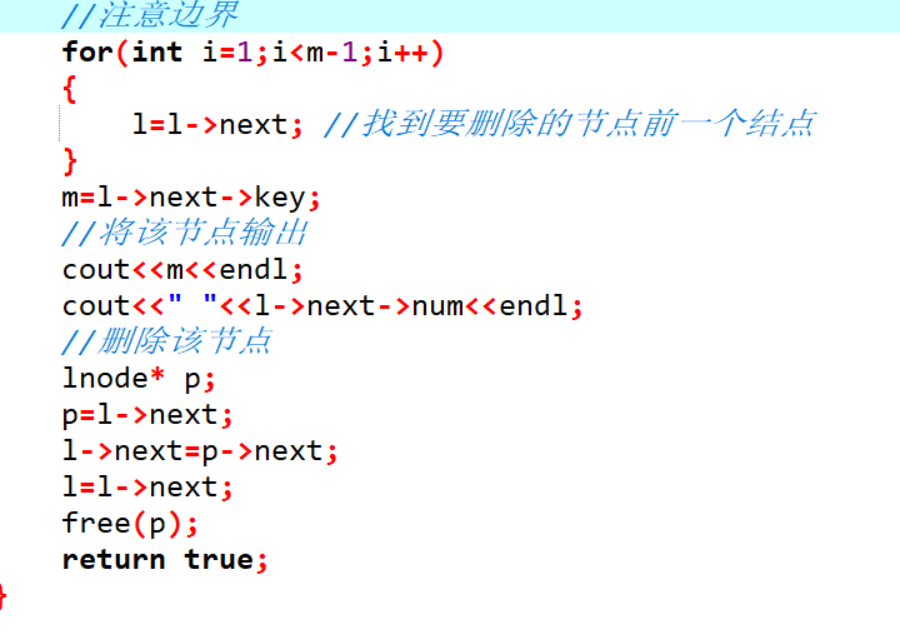
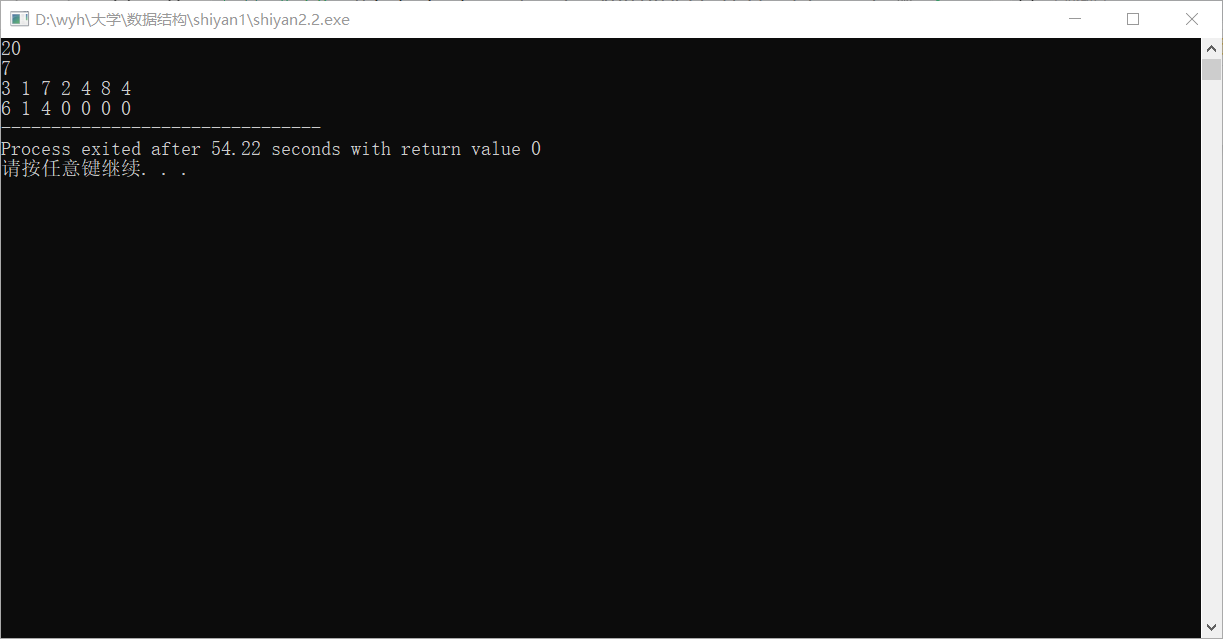
return 0;

}

2.6调试过程截图

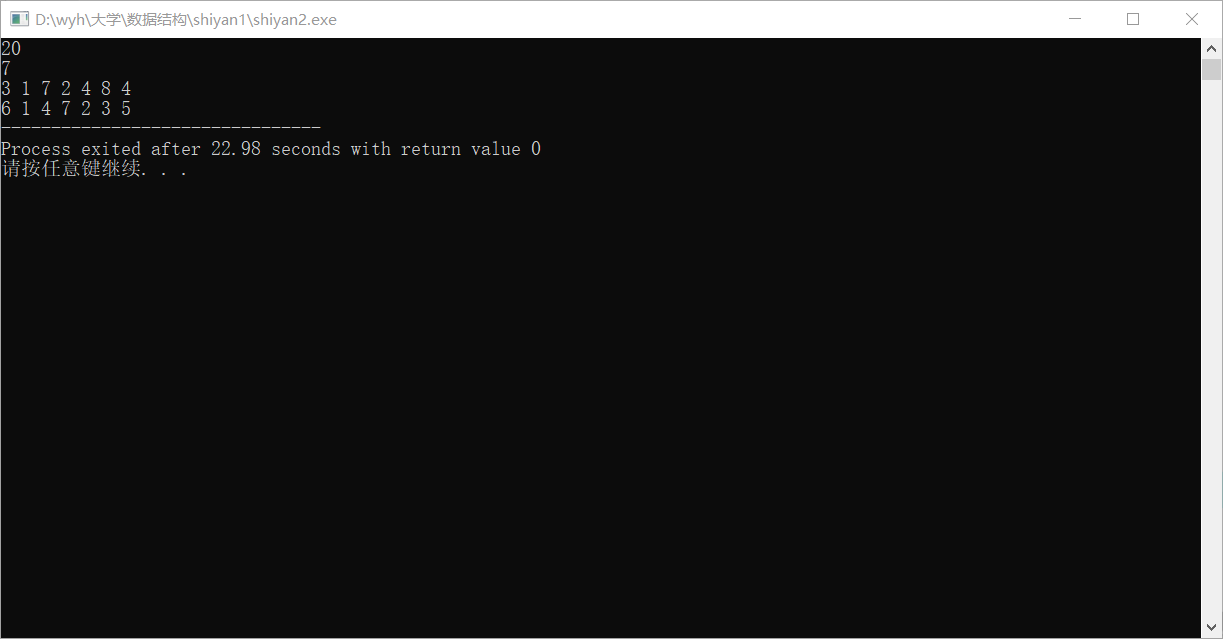
输出每次出列的人，查看出错在哪





如果只找到m前一个节点删除当前结点的话m=1会出问题，只能将后面一个节点填到前一个。

**2.7运行结果截图**

****

**3选做**

1. **城市（思路大致一致）**

**#include<cstdio>**

**#include<cstdlib>**

**#include<cstring>**

**#include<iostream>**

**#include<cmath>**

**using namespace std;**

**struct lnode**

**{**

**double x,y;//坐标**

**char ncity[100];//城市名**

**}city[10086];**

**typedef struct lnode\* linklist;**

**int caozuo;//选择操作类型**

**linklist la,lb,lc;**

**int chengshishu=0;**

**//插入**

**void linsert()**

**{**

**chengshishu++;**

**cout<<"城市名"<<endl;**

**cin>>city[chengshishu].ncity;**

**cout<<"城市坐标"<<endl;**

**cin>>city[chengshishu].x>>city[chengshishu].y;**

**}**

**//创建新链表**

**bool lcreate()**

**{**

**int k;**

**cout<<"输入创建城市数："<<endl;**

**cin>>k;**

**while(k)**

**{**

**linsert();//插入节点**

**k--;**

**//cout<<k<<endl;**

**}**

**return true;**

**}**

**bool lfind()**

**{**

**int cha;**

**double xx,yy;**

**char ch[20];**

**int flag=0;**

**cout<<"1查城市"<<endl;**

**cout<<"2查坐标"<<endl;**

**cin>>cha;**

**if(cha==1)**

**{**

**cout<<"输入坐标"<<endl;**

**cin>>xx>>yy;**

**for(int i=1;i<=chengshishu;i++)**

**{**

**if(city[i].x==xx&&city[i].y==yy)**

**{**

**cout<<city[i].ncity<<endl;**

**flag=1;**

**break;**

**}**

**}**

**if(!flag)//没找到该城市**

**{**

**cout<<"不存在此城市"<<endl;**

**}**

**}**

**else if(cha==2)**

**{**

**cout<<"输入城市名"<<endl;**

**cin>>ch;**

**for(int i=1;i<=chengshishu;i++)**

**{**

**if(strcmp(city[i].ncity,ch)==0)//strcmp比较两个字符串**

**{**

**cout<<city[i].x<<","<<city[i].y<<endl;**

**flag=1;**

**break;**

**}**

**}**

**if(!flag)**

**{**

**cout<<"不存在此城市"<<endl;**

**}**

**}**

**return true;**

**}**

**bool ldelete()**

**{**

**int cha;**

**int flag=0;**

**double xx,yy;**

**char ch[20];**

**int cur=0;**

**cout<<"1通过坐标"<<endl;**

**cout<<"2通过城市"<<endl;**

**cin>>cha;**

**if(cha==1)**

**{**

**for(int i=1;i<=chengshishu;i++)**

**{**

**if(city[i].x==xx&&city[i].y==yy)**

**{**

**cout<<city[i].ncity<<endl;**

**flag=1;**

**cur=i;**

**break;**

**}**

**}**

**if(!flag)//没找到该城市**

**{**

**cout<<"不存在此城市"<<endl;**

**}**

**else**

**{**

**strncpy ( city[cur].ncity,city[chengshishu].ncity , sizeof(city[chengshishu].ncity) );**

**city[cur].x=city[chengshishu].x;**

**city[cur].y=city[chengshishu].y;**

**chengshishu--;**

**}**

**}**

**else if(cha==2)**

**{**

**cout<<"输入城市名"<<endl;**

**cin>>ch;**

**for(int i=1;i<=chengshishu;i++)**

**{**

**if(strcmp(city[i].ncity,ch)==0)//strcmp比较两个字符串**

**{**

**cout<<city[i].x<<","<<city[i].y<<endl;**

**flag=1;**

**cur=i;**

**break;**

**}**

**}**

**if(!flag)**

**{**

**cout<<"不存在此城市"<<endl;**

**}**

**else**

**{**

**strncpy ( city[cur].ncity,city[chengshishu].ncity , sizeof(city[chengshishu].ncity) );**

**city[cur].x=city[chengshishu].x;**

**city[cur].y=city[chengshishu].y;**

**chengshishu--;**

**}**

**}**

**return true;**

**}**

**bool lupdate()**

**{**

**int cha,flag=0,cur=0;**

**double xx,yy;**

**char ch[20];**

**cout<<"1通过坐标"<<endl;**

**cout<<"2通过城市"<<endl;**

**cin>>cha;**

**if(cha==1)**

**{**

**cout<<"输入坐标"<<endl;**

**cin>>xx>>yy;**

**for(int i=1;i<=chengshishu;i++)**

**{**

**if(city[i].x==xx&&city[i].y==yy)**

**{**

**flag=1;**

**cur=i;**

**break;**

**}**

**}**

**if(!flag)//没找到该城市**

**{**

**cout<<"不存在此城市"<<endl;**

**}**

**else**

**{**

**cout<<"输入新名字"<<endl;**

**cin>>city[cur].ncity;**

**}**

**}**

**else if(cha==2)**

**{**

**cout<<"输入城市名"<<endl;**

**cin>>ch;**

**for(int i=1;i<=chengshishu;i++)**

**{**

**if(strcmp(city[i].ncity,ch)==0)//strcmp比较两个字符串**

**{**

**cout<<city[i].x<<","<<city[i].y<<endl;**

**flag=1;**

**cur=i;**

**break;**

**}**

**}**

**if(!flag)**

**{**

**cout<<"不存在此城市"<<endl;**

**}**

**else**

**{**

**cout<<"输入新坐标"<<endl;**

**cin>>city[cur].x>>city[cur].y;**

**}**

**}**

**return true;**

**}**

**bool pan(double xx,double yy,double dis,int i)**

**{**

**double dx=(city[i].x)-xx;**

**double dy=(city[i].y)-yy;**

**if(((dx\*dx+dy\*dy)<=dis\*dis)&&(!((dx==0)&&(dy==0))))**

**{**

**// cout<<l->x<<l->y<<endl;**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool lcompute()**

**{**

**double xx,yy;**

**double dis;**

**cout<<"输入坐标"<<endl;**

**cin>>xx>>yy;**

**cout<<"输入距离"<<endl;**

**cin>>dis;**

**for(int i=1;i<=chengshishu;i++)**

**{**

**if(pan(xx,yy,dis,i)==true)**

**{**

**cout<<city[i].ncity<<endl;**

**cout<<city[i].x<<","<<city[i].y<<endl;**

**}**

**}**

**return true;**

**}**

**int main()**

**{**

**while(1)**

**{**

**cout<<"此输入想要的操作（第一次请选择1）"<<endl;**

**cout<<"1创建"<<endl;**

**cout<<"2查询"<<endl;**

**cout<<"3插入"<<endl;**

**cout<<"4删除"<<endl;**

**cout<<"5更新"<<endl;**

**cout<<"6计算所有城市"<<endl;**

**cout<<"7退出"<<endl;**

**cin>>caozuo;**

**if(caozuo==7)**

**{**

**break;**

**}**

**else**

**{**

**switch(caozuo)**

**{**

**case 1:**

**lcreate();**

**break;**

**case 2:**

**lfind();**

**break;**

**case 3:**

**linsert();**

**break;**

**case 4:**

**ldelete();**

**break;**

**case 5:**

**lupdate();**

**break;**

**case 6:**

**lcompute();**

**break;**

**default:**

**break;**

**}**

**}**

**}**

**return 0;**

**}**

1. **约瑟夫环**
2. **因为顺序结构不能用指针链接，所以需要多加判断该人是否出列，出列则跳过。**
3. **其次不像链表一样，需要用模运算使其形成循环**

**#include<cstdio>**

**#include<cstdlib>**

**#include<cstring>**

**#include<iostream>**

**#include<cmath>**

**using namespace std;**

**struct lnode**

**{**

**int key;//密码**

**int num;//编号**

**int flag;//是否已经输出**

**}huan[10086];**

**typedef struct lnode\* linklist;**

**int m;//出列值**

**int renshu;//测试人数**

**linklist l,la;**

**int main()**

**{**

**cin>>m;**

**cin>>renshu;**

**for(int i=1;i<=renshu;i++)**

**{**

**cin>>huan[i].key;**

**huan[i].num=i;**

**huan[i].flag=0;**

**}**

**int dangqianren=renshu;**

**int cur;**

**for(int i=1;i<=renshu;i++)**

**{**

**int jishu=m;**

**//cout<<i<<":";**

**if(i==1)**

**{**

**cur=0;**

**}**

**while(jishu!=0)//找到要输出的**

**{**

**cur=(cur+1)%dangqianren;//使用mod运算**

**if(cur==0)cur=dangqianren;**

**if(huan[cur].flag==0)//还未出局**

**{**

**jishu--;**

**//cout<<"jihsu"<<jishu<<" ";**

**}**

**//cout<<"cur"<<cur<<endl;**

**}**

**cout<<huan[cur].num<<endl;**

**m=huan[cur].key;//更新m**

**huan[cur].flag=1;//已经出列**

**//cout<<endl;**

**}**

**return 0;**

**}**

1. **实验心得**

在第一个实验中遇到的问题较多，该实验所需要的操作与函数较多，前一个函数的操作可能影响到后一个函数。首先在创建函数的时候就遇到了问题，输入了一个城市之后程序直接结束了，后来发现是没有调用初始化。其次是再删除是没有删除成功，还有就是计算的时候发现少了要输出的城市，发现是一个逻辑判断写错了，应该是!((l->next->x==xx)&&(l->next->y==yy))，而我直接写成(l->next->x!=xx)&&(l->next->y!=yy)这两者逻辑是不对的。今后也要考虑清楚而不能冒然动手写。

在第二个实验中遇到的问题主要是一个逻辑问题，删除操作本来想找到前一个结点并且删除当前结点，但是这么做需要讨论m=2和m=1，且m=1会出问题，考虑了一下只能将后面一个节点填到前一个在删除了。所以说一定要考虑清楚所有情况。而顺序结构的方法则相对简单，在想到使用模运算后相对不容易出错。

通过这次实验，我对数据结构的应用有了一定了解，加强了对删除插入操作的理解，也明白了实际应用中顺序结构和链表的差异，明白这两种各有优缺点。单个城市信息的删除和插入变得异常轻易，不需要挪动大量的数据，只需要修改结点即可达成目的，相比顺序表确实方便许多。而约瑟夫环，循环链表的使用让我们不用思索怎么建立循环，只需要将尾指针连接到头指针就行。也知道在解决一个具体问题时要先想出一个模型，想好其数据类型，再通过过相应算法编写出程序。