**2020秋季学期**

**数据结构实验报告**

**实验一 线性表实验**

**班级：\_\_计实验19\_**

**学号：19101130120**

**姓名：\_\_\_黄周杰\_\_**

|  |
| --- |
| **评语：**  **日期：** |

**实验目的：**

熟悉线性表的基本运算在顺序存储结构和链式存储结构上的实现，其中重点掌握线性表的链式表示时各种操作的实现。

**1、**约瑟夫

**需求分析：**（包括对问题的理解，解决问题的策略、方法描述）

根据题目描述，是n个人围成一圈，然后报数，由此想到建立一个循环链表，来实现报数的循环。又由于是要输出出列顺序，我们需要对每一个人进行编号，也就意味着在数据结构中，我们要存储这个人的序号，当这个人出列的时候，我们把这个人的序号输出，再讲m变为这个人的数。出列的时候，需要实现链表的删除。

**系统设计：**（包括数据结构定义、抽象出基本操作描述、主程序模块处理过程描述）

**typedef** struct person**{**

int code**;** //这个人的手中的上限值

int num**;** //这个人的编号

person**\*** next**;**

**}**person**;**

数据结构结构定义

基本操作:

input**(**int **&**m**,** int **&**n**,** person**\*** head**)**

初始条件：head头指针已存在.

操作结果: m，n附上了对应的值，head指向了存储了信息的循环链表的第一个结构 体空间

launch**(**int m**,** int n**,** person**\*** head**)**

初始条件:head头指针已存在

操作结果:输出出列顺序

过程描述：

整体上分为两个模块，一是进行输入，二是开始报数.

在第一个模块（**输入**），要将每个人之间的关系用循环链表储存。

在第二个模块（**报数**），就需要进行循环报数，然后对满足要求的person，输出相应的CODE，然后再将这个人移除,报数变为1，更新m值。而这个循环判断的标志就是，这个循环链表中没有人存在。

**基本操作的实现：**（对各基本操作实现的描述）

void input**(**int **&**m**,** int **&**n**,** person**\*** head**)**

**{**

cin**>>**m**>>**n**;** //先输入m和n

int i **=** n**;**

person**\*** fp **=** head**;**

person**\*** p**;**

**while(**i**--)**

**{**

p **=** **new** person**;** //创建列表

cin**>>**p**->**code**;** //输入对应的上限值

p**->**num **=** n **-** i**;** //输入对应的编号

fp**->**next **=** p**;** //将指针指向的位置修改

fp **=** p**;**

**}**

p**->**next **=** head**->**next**;** //将最后一个人的next指向第一个人

**}**

输入

**描述**：先输入m和n，然后用n的值进行循环n次。在每一次，都要创建相应的person结构，然后将code输入相应空间，同时上一个person结构中的next指针要指向现在的person结构。

报数

void launch**(**int m**,** int n**,** person**\*** head**)**

**{**

int i **=** 1**,** count **=** n**;**

person**\*** p **=** head**->**next**;** //p指向当前节点的下一个节点

person**\*** fp **=** head**;**

**while(**count**)**

**{**

**if(**i **==** m**)** //判断这个人是否报到m的时候

**{**

i **=** 1**;**

m **=** p**->**code**;** //更新m

cout**<<**p**->**num**<<**' '**;** //输出对应的序号

fp**->**next **=** p**->**next**;** //删除节点

p **=** p**->**next**;**

count**--;** //将还存在的人数减一

**continue;**

**}**

i**++;** //报数加一

fp **=** fp**->**next**;** //更新指针

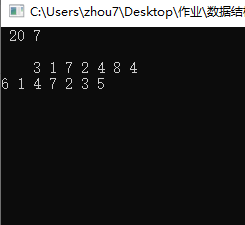
p **=** p**->**next**;**

**}**

**}**

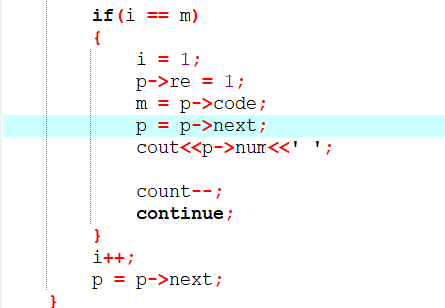
**描述**：在循环中，先进行判断这个人是否出去了，如果出去了，就直接下一个;如果没出去，判断这个人报的数是否满足条件，满足:将该人的状态置1，输出num，m取为code，i置1。不满足：i加1，p指向下一个.

**测试结果：**（输入的测试数据及运行结果、正确性、在线测试情况）



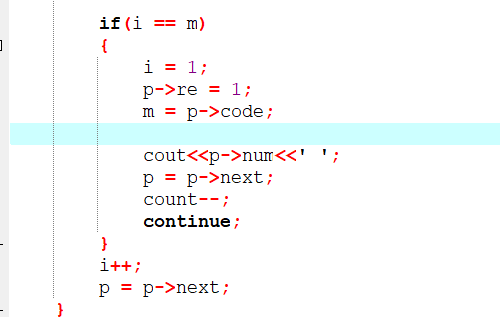
**调试分析：**（包括调试过程中对原设计的修改，以及遇到的问题和解决的方法）

**问题一：**



修改前

这样的源码，在输出的时候，会先到下一个再输出，这样输出的永远是下一个的序号，应该先输出之前的答案，再将执政移动到下一位。



修改后

**问题二：**

一开始是打算，新设置一个变量re来判断是否被移除，但是这样耗费了空间，还要另作判断，于是我打算用删除节点的方法来达到循环的目的。

2、窗口问题

**需求分析：**（包括对问题的理解，解决问题的策略、方法描述）

由题可知，窗口问题的输入是讲究后到在前的规则，那么我们在输入的时候，就需要把最新的输入放到链表的最前面，来形成要求的顺序。在点击的时候，我们依次判断这个点是否在框内，如果在，就输出对应的序号，并将这个框放在最前面，这就是需要我们把这个结构体，放在链表的最前面，如果我们最后没有点击到框，就返回ignored。

**系统设计：**（包括数据结构定义、抽象出基本操作描述、主程序模块处理过程描述）

数据结构定义:

**typedef** struct LR**{**

int id**;**

int point**[**4**];**

struct LR **\***next**;**

**}**LR**,** **\***LinkList**;**

基本操作:

update**(**LR **\***np**,** LR **\***p**,** LR **\***f**)**

初始条件:节点np , p , f存在

操作结果;将p对应的框放在最前面，然后更新链表的顺序

input**(**int**&** N**,** int**&** M**,**LR **\***f**)**

初始条件:N,M已存在节点 f已存在

操作结果:给N, M赋值，建立了存储了窗口信息的链表

click**(**LR **\***f**)**

初始条件:节点f已存在

操作结果:输入了点击的点，再输出了对应的框。并更新了f链表.

整体上分为两个板块：

1. 输入窗口，在这个版块中，输入一个窗口，便将其放在最前面，考虑到实现困难程度，便使用链表进行实现。
2. 输入点击，在这个版块中，我们输入点击位置，然后我们根据顺序，对点和框进行判断（在这个框以内），如果在里面，就输出相应框，然后把这个框放在最前面。

**基本操作的实现：**（对各基本操作实现的描述）

void input**(**int**&** N**,**int**&** M**,**LR **\***f**)**

**{**

int i**;**

LR **\***p**;** //创建一个框

p**->**id **=** 1**;** //输入对应的信息

cin**>>**N**>>**M**;**

cin**>>**p**->**point**[**0**]>>**p**->**point**[**1**]>>**p**->**point**[**2**]>>**p**->**point**[**3**];**

p**->**next **=NULL;**

LR **\***np**;**

**for(**i **=** 2**;** i **<=** N**;**i**++)**

**{**

np **=** p**;**

p **=** **new** LR**;** //创建一个框并输入对应的信息

p**->**next **=** np**;**

f**->**next **=** p**;** //将这个框放到最前面

p**->**id **=** i**;**

cin**>>**p**->**point**[**0**]>>**p**->**point**[**1**]>>**p**->**point**[**2**]>>**p**->**point**[**3**];**

**}**

**}**

实现输入

**描述**：创建一个空的LR结构体,然后将框的数据放入这个结构体内。

点击实现

void update**(**LR **\***np**,** LR **\***p**,** LR **\***f**)**

**{**

np**->**next **=** p**->**next**;** //将这个框放到最前面

p**->**next **=** f**->**next**;**

f**->**next **=** p**;**

**}**

void click**(**LR **\***f**)**

**{**

int x**,**y**,**j **=** 0**;**

cin**>>**x**>>**y**;** //输入点击的位置

LR **\***p **=** f **->**next**;**

LR **\***np **=** f**;**

**while(**p**)** //遍历这个链表，然后判断是否在框内

**{**

**if(**x **>=** p**->**point**[**0**]** **&&** y **>=** p**->**point**[**1**]** **&&** x **<=** p**->**point**[**2**]** **&&** y **<=** p**->**point**[**3**])**

**{**

j **=** 1**;** //用于判断是否点到了框，并结束循环

**break;**

**}**

np **=** np**->**next**;**

p **=** p**->**next**;**

**}**

**if(**j**)**

**{**

cout**<<**p**->**id**<<**endl**;** //输出

update**(**np**,**p**,**f**);**

**}**

**else**

**{**

cout**<<**"IGNORED"**<<**endl**;**

**}**

**}**

**描述**：先输入点的位置，遍历链表，只要这个点在一个框内，返回这个框的标号，于是将这个结构体放在链表最前面。如果没有返回一个标号，便输出IGNORED.

**测试结果：**（输入的测试数据及运行结果、正确性、在线测试情况）

输入数据

【样例输入】

3 4

0 0 4 4

1 1 5 5

2 2 6 6

1 1

0 0

4 4

0 5

结果

2

1

1

IGNORED

**调试分析：**

无.