**实验报告**

题目：约瑟夫环

班级：电信1809

姓名：叶杰栋

学号：U201813497

完成日期：2019/3/26

1. 需求分析
2. 输入的形式：一组正整数；输入值的范围：一组任意正整数。
3. 输出的形式：一串编号。
4. 程序所能达到的功能：完成约瑟夫环的游戏，按顺序输出出局人的编号。
5. 测试数据：①m=20,n=7,密码：3,1,7,2,4,8,4，输出：3,2,1,5,4,6,7。（错误输出）

②m=20,n=7,密码：3,1,7,2,4,8,4，输出：6,1,4,7,2,3,5。（正确输出）

③m=13,n=5,密码：6,1,4,2,6，输出：3,2,4,1,5。（正确输出）

1. 概要设计

先定义一个循环单链表，每个节点中有三个域，分别储存玩家的编号、密码以及下一个节点的地址。游戏开始，利用循环找到要出列的玩家对应的节点，然后将该玩家的密码取出记录，并打印该玩家的编号，然后删除该节点（玩家出列），直到只剩两个玩家。最后两个玩家单独处理，按出列先后顺序打印两个玩家的编号。最后释放这两个节点空间。

1. 详细设计

typedef struct node

{

int data;

int num;

struct node\* next;

}node;//定义节点

游戏过程伪代码：

程序开始

for(i=1;i<n-1;i++)

{

for(j=1;j<m;j++)

{

p=p->next;

}

m=p->data;

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

q=p;

while(q->next!=p)

{

q=q->next;

}

q->next=p->next;

free(p);

p=q->next;

}

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

{

p=p->next;

}

q=p->next;

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

free(p);

free(q);

函数调用关系图：

main→create

1. 调试分析
2. 建立的循环链表的指向反了。解决方法：重新建立指向正确的循环链表；或者就地逆置该链表。
3. 整体时间复杂度O（nm）或O（n²）。
4. 经验体会：建立链表的时候应该注意节点建立的顺序，避免弄错数据输入的顺序。
5. 用户使用说明

按照输入游戏相关数值即可。

1. 测试结果

（1）m=20,n=7,密码：3,1,7,2,4,8,4，输出：6,1,4,7,2,3,5。

（2）m=13,n=5,密码：6,1,4,2,6，输出：3,2,4,1,5。

（3）m=20,n=10,密码：4,5,1,7,2,8,9,1,2,5，输出：10,5,7,9,2,1,8,3,4,6。

1. 附录

完整代码：

#include <stdio.h>

#include <stdlib.h>

typedef struct node

{

int data;

int num;

struct node\* next;

}node;

node\* create(int n)

{

int i,a;

node \*head,\*p,\*q,\*x,\*y,\*z;

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

head->next=NULL;

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

{

printf("请输入第%d个人的密码:",i+1);

scanf("%d",&a);

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

p->data=a;

p->num=i+1;

p->next=head->next;

head->next=p;

}

q=head;

while(q->next!=NULL)

{

q=q->next;

}

q->next=head->next;

free(head);

z=q;

x=z;

y=NULL;

z=NULL;

while(x)

{

y=x->next;

x->next=z;

z=x;

x=y;

}

return q;

}

void main()

{

int n,i,j;

int m=20;

node \*list,\*p,\*q;

printf("请输入参与游戏的人数:");

scanf("%d",&n);

list=create(n); //初始化单循环链表

p=list;

for(i=1;i<n-1;i++) //开始游戏

{

for(j=1;j<m;j++)

{

p=p->next;

}

m=p->data;

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

q=p;

while(q->next!=p)

{

q=q->next;

}

q->next=p->next;

free(p);

p=q->next;

}

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

{

p=p->next;

}

q=p->next;

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

free(p);

free(q);

}