《实验五——约瑟夫环问题仿真》

（一）问题描述

设编号为 1，2，…，n(n>0)个人按顺时针方向围坐一圈，每人持有一个正整数密码。开始时任意给出一个报数上限 m，从第一个人开始顺时针方向自 1 起顺序报数，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起重新自 1 报数；如此下去直到所有人全部出列为止。

（二）解决思路

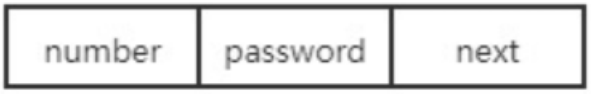
通过问题描述，很自然地想到使用不带头结点的单向循环链表，用以表示 n 个人顺时针围坐形成的圈。一方面，从循环链表任一结点出发均可访问到表中其它结点，这使得报数操作容易实现；另一方面，删除（即报数到 m 的人出列）操作中不需要区分尾结点还是中间结点，使操作简化。

（三）数据结构

循环链表：循环链表是线性链表的一种变形。在线性链表中，每个结点的指针都指向它的下一个结点，最后一个结点的指针域为空，表示链表结束。而循环链表则将表中最后一个结点的指针域指向首元结点，使得整个链表形成一个环。由此，从表中任一结点出发均可找到表中其它结点。如下图所示为单向循环链表：

（四）算法分析

① 建立一个不带头结点的具有 n 个结点的约瑟夫环问题循环链表，表中的结点定义如下：



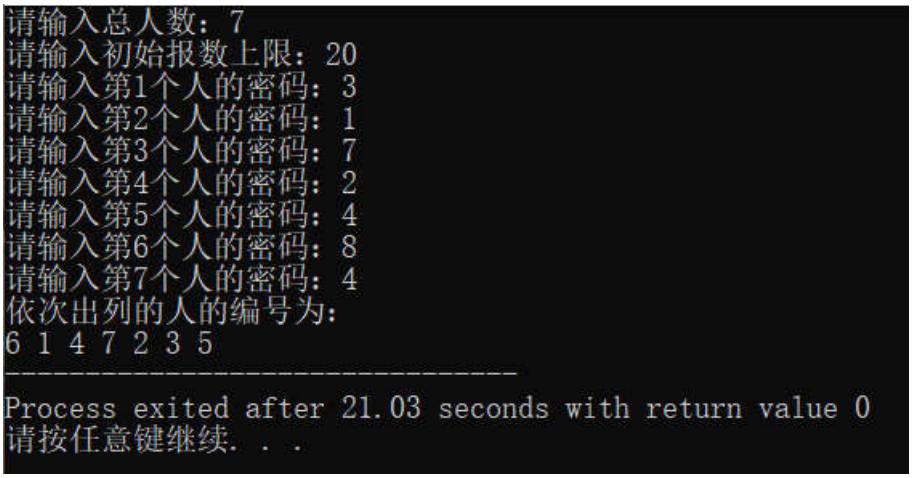
② 在循环链表中，循环查找密码为 m 的结点，将其输出，并取出该结点的密码赋给 m，最后将该结点从链表中删除，直到输出循环链表中的所有元素为止。

（五）运行结果

测试数据：N=7，七个人的密码依次为 3，1，7，2，4，8，4。
初始报数上限值 m=20。

预期的出列编号：6，1，4，7，2，3，5。

结果截图：



结论：程序运行结果与预期相符，成功实现了约瑟夫环问题的仿真。

（六）反思总结

通过本次实验，我进一步了解了循环链表这一数据结构，熟练掌握了链表的插入、删除操作，同时也提高了数学建模与编程能力。

（七）源程序

```
//约瑟夫环问题仿真
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    unsigned int number;            //编号
    unsigned int password;          //密码
    struct Node* next;              //指向下一个结点的指针
} node;

//构建具有n个结点的循环链表
node* CreateList(int n) {
    node* head = (node*)malloc(sizeof(node));
    printf("请输入第1个人的密码：");
    scanf("%d", &head->password);
    head->number = 1;
    head->next = head;
    node* temp = head;
    for (int i = 2; i <= n; i++) {
        temp->next = (node*)malloc(sizeof(node));
        printf("请输入第%d个人的密码：", i);
        scanf("%d", &temp->next->password);
        temp->next->number = i;
        temp->next->next = head;
        temp = temp->next;
    }
    return head;
}

//打印 p->next 的编号，并返回其密码
int OutList(node** Head, node* p) {
    int num, password;
    //删除首元结点
    if (p->next == (*Head)) {
        num = (*Head)->number;
        password = (*Head)->password;
        (*Head) = (*Head)->next;
        free(p->next);
        p->next = (*Head);
    }
```

```
    //删除其它结点
    else {
        node* temp = p->next;
        num = temp->number;
        password = temp->password;
        p->next = temp->next;
        free(temp);
    }
    printf("%d ", num);
    return password;
}

//仿真
void simulation(node* head, int n, int m) {
    int state = 1;        //用于标志第一次报数
    node* p;
    for (int k = 0; k < n; k++) {
        for (int i = 0; i < m - 1; i++) {
            if (state) {
                p = head;
                state = 0;
            }
            else
                p = p->next;
        }
        m = OutList(&head, p);
    }
}

int main(void) {
    int N, m;
    printf("请输入总人数：");
    scanf("%d", &N);
    printf("请输入初始报数上限：");
    scanf("%d", &m);
    node* head = CreateList(N);
    printf("依次出列的人的编号为：\n");
    simulation(head, N, m);

    return 0;
}
```