## 实验内容

**Joseph问题求解算法的设计与实现**

**问题描述：**

约瑟夫（Joseph）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）。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；n=7，7个人的密码依次为：3，1，7，2，4，8，4，首先m值为6（正确的出列顺序应为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。可设n≤30。此题所用的循环链表中不需要“头结点”，请注意空表和非空表的界限。

**选作内容：**

向上述程序中添加在顺序结构上实现的部分

## 实验目的

掌握链表的基本操作：插入、删除、查找等运算，能够灵活应用链表这种数据结构。

## 程序清单

### 链表实现

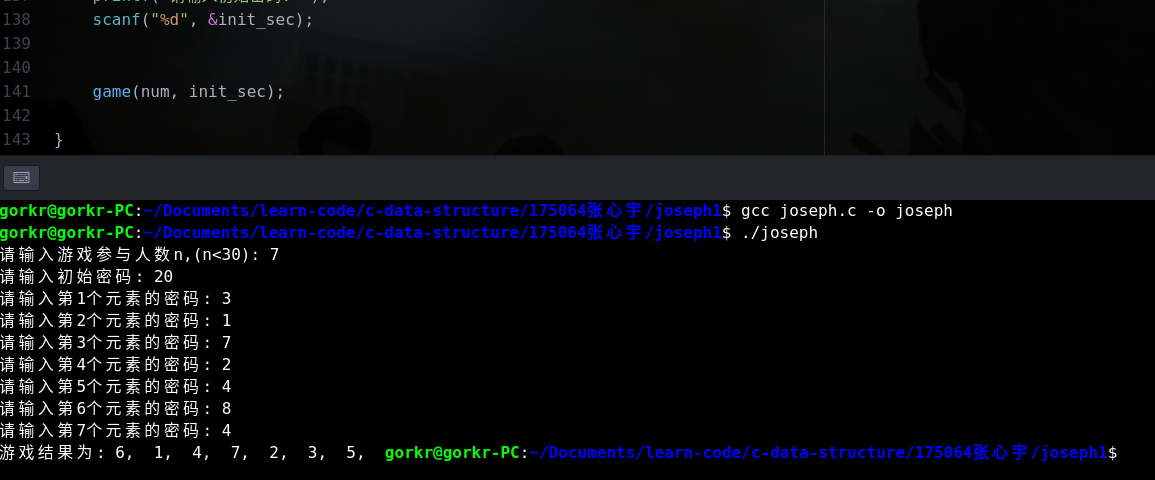
// joseph.c  
// 用链表实现joseph环  
/\*\*思路描述:  
 \* 1. 一个节点,要同时保存他的 原始位置, 和 密码.  
 \* head节点的密码属性存储表长度, 并随删除而变化.  
 \* 2. 每次出列, 都从环中删除出列节点. 并 记录出列元素位置信息.  
 \* 链表中仅剩一个节点, 可不删除.  
 \* 3. 循环次数是确定的, 使用for循环即可.  
 \* 4. 删除head->next节点时,注意head丢失.  
 \*/  
  
#include<stdio.h>  
#include<stdlib.h>  
  
// 节点结构.  
typedef struct LNode{  
 int seceret; // 存储密码  
 int pos; // 初始位置  
 struct LNode \* next;  
}LinkList;  
  
  
/\*\*  
 \* 创建带头结点的单向循环链表,长度为length. 头结点不参与循环.  
 \* @param length 循环链表中节点个数(不含头结点)  
 \* @return 头结点指针  
 \*/  
LinkList \* init\_circle(int length){  
 LinkList \*head, \*node, \*p; // p负责向后移动  
  
 head = (LinkList \*)malloc(sizeof(LinkList));  
 if(!head)exit(EXIT\_FAILURE);  
 head->seceret = length; // 记录循环链表中节点个数.  
 head->pos = 0;  
  
 p = head;  
 for(int i=1; i<=length; i++){  
 node = (LinkList \*)malloc(sizeof(LinkList));  
 if(!node)exit(EXIT\_FAILURE);  
 printf("请输入第%d个元素的密码: ", i);  
 scanf("%d", &node->seceret);  
 node->pos = i;  
 p->next = node;  
 p = node;  
 }  
 p->next = head->next; // 闭合循环.  
 return head;  
}  
  
  
/\*\*  
 \* 删除now节点  
 \* @param head 循环链表头结点  
 \* @param now 要删除的节点  
 \* @return 0 删除, 1 不需要执行删除.  
 \*/  
int del\_node(LinkList \* head, LinkList \* now){  
 if(head->seceret<=1)return 1; // 如果除头结点外只剩一个节点, 返回1.  
  
 if(now==head->next){head->next=now->next;} // 如果删除的节点恰好是head的next节点,让head指向下一节点  
 // 给我找崩溃了.......这个错.  
  
 LinkList \* p;  
 p = head;  
 while(p->next!=now){ // 找到now节点的前一个节点.  
 p=p->next;  
 }  
  
 p->next=now->next;  
 free(now);  
 head->seceret--;  
  
  
 return 0;  
}  
  
  
/\*\*  
 \* 实现joseh游戏  
 \* @param num 参与joseph人数(节点数量)  
 \* @param m 初始密码  
 \* @return 0: 成功执行  
 \*/  
int game(int num, int m){  
 int result[num];  
 LinkList \*p, \*q, \*joseph\_circle = init\_circle(num);  
 p = joseph\_circle->next;  
 for(int i=0; i<num-1; i++){  
  
 //找到要删除的节点  
 while((m-1)%joseph\_circle->seceret){  
 p = p->next;  
 m--;  
 }  
 m = p->seceret;  
 q = p->next; // 记录下一个节点位置  
 result[i] = p->pos; // 记录位置  
 del\_node(joseph\_circle, p);  
 p = q;  
 }  
 result[num-1] = joseph\_circle->next->pos;  
 printf("游戏结果为: ");  
 for(int i=0; i<num; i++){  
 printf("%d, ", result[i]);  
 }  
}  
  
  
int main(){  
 int num , init\_sec; // num:游戏人数, init\_sec:初始密码  
  
 printf("请输入游戏参与人数n,(n<30): ");  
 scanf("%d", &num);  
  
 while(1){  
 if(0<num&&num<=30){break;}  
 printf("输入错误,请重新输入: ");  
 scanf("%d", &num);  
 }  
  
 printf("请输入初始密码: ");  
 scanf("%d", &init\_sec);  
  
  
 game(num, init\_sec);  
  
}

### 选做内容： 顺序表实现

// joseph2  
// 顺序表实现joesh环  
#include<stdio.h>  
#include<stdlib.h>  
  
// 个人节点结构  
typedef struct {  
 int seceret;  
 int pos; // 初始位置  
}Node;  
  
// 顺序表  
#define LIST\_INIT\_SIZE 30  
typedef struct {  
 Node \*array;  
 int length; // 当前元素含量  
}List;  
  
  
/\*\*  
 \* 初始化长度为num的顺序表  
 \* @param num 线性表长度  
 \* @return 顺序表指针  
 \*/  
List\* init\_List(int num){  
 // 创造一个空线性表  
 List \* l = (List\*)malloc(sizeof(List));  
 if(!l)exit(-1);  
 l->array = (Node\*)malloc(sizeof(Node)\*LIST\_INIT\_SIZE);  
 l->length = 0;  
  
 // 线性表插入元素  
 int seceret;  
 for(int i=1; i<=num; i++){  
 printf("请输入第%d个元素的密码: ", i);  
 scanf("%d", &seceret);  
 l->array[i-1].seceret = seceret;  
 l->array[i-1].pos = i;  
 l->length++;  
 }  
 return l;  
}  
  
  
/\*\*  
 \* 删除顺序表中下标为i的节点  
 \* @param l 被操作线性表  
 \* @param i 删除节点下标  
 \* @return 被删除节点  
 \*/  
Node del\_node\_sq(List \* l, int i){  
 if((i<0)||(i>=l->length))exit(-1);  
 Node deled\_node = l->array[i];  
 l->length--;  
 for(i;i<l->length;i++){  
 l->array[i]=l->array[i+1];  
 }  
 return deled\_node;  
}  
  
/\*\*  
 \* joseph循环  
 \* @param num 参与游戏人数  
 \* @param m 游戏初始密码  
 \* @return 0,游戏顺利执行  
 \*/  
int game(int num, int m){  
 int result[num], del=0;  
 List \* josph\_circle = init\_List(num);  
 for(int i=0; i<num; i++){  
  
 // 找到要删除的节点  
 while((m-1)%josph\_circle->length ){  
 del++;  
 del=del%josph\_circle->length;  
 m--;  
 }  
 del=del%josph\_circle->length; // 防止(m-1)=length的情况， 以至越界。  
  
 m = josph\_circle->array[del].seceret;  
 result[i]=josph\_circle->array[del].pos;  
 del\_node\_sq(josph\_circle, del);  
 }  
 for(int i=0; i<num; i++){  
 printf("%d, ", result[i]);  
 }  
 return 0;  
  
}  
  
int main(){  
 int num, init\_sec;  
 printf("请输入游戏参与人数n,(n<30): ");  
 scanf("%d", &num);  
  
 while(1){  
 if(0<num&&num<=30){break;}  
 printf("输入错误,请重新输入: ");  
 scanf("%d", &num);  
 }  
  
 printf("请输入初始密码: ");  
 scanf("%d", &init\_sec);  
  
 game(num, init\_sec);  
 return 0;  
}

## 运行结果

### 链表实现结果



### 顺序表实现结果



## 分析与思考

代码方面：

1. 代码太过冗长， 应尽力减少不必要的赋值语句。
2. 没有将输入单元独立出来。

数据结构方面：

1. 链表实现时，可不使用头结点。

其他：

1. 测试驱动开发， 每一单元写完后应适量测试， 尽可能保证单元正确。
2. 重点考虑特殊情况，如最后一个节点的删除情况。