数据结构上机报告

题目：约瑟夫环

班级：04班 姓名： 学号： 完成日期：2020.10.08

一、需求分析

1.问题描述

约瑟夫环（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）。一开始人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，从它在顺时针方向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。

2.问题输入

输入数据第一行为两个正整数n和m，分别表示人的个数及初始随机数，每组数据的第二行为n个整数，表示每个人持有的密码。假定输入数据范围小于整型变量取值范围。

3.问题输出

用一行输出n个整数表示依次出圈人的编号，整数之间用空格分隔。

1. 程序功能

按照给定的n、m，根据出圈顺序依次输出出圈人的编号，直到输出所有参与者的的编号为止。程序的时间限制为2S，内存限制为10000Kb。

1. 测试数据

输入样例

7 20

3 1 7 2 4 8 4

输出样例

6 1 4 7 2 3 5

二、概要设计:

程序用不带头结点的循环链表作为存储数据的数据结构类型，利用链表的遍历和删除操作依次获取出圈人的编号并输出。

循环链表实现

1. 定义一个链表结点类型

typedef struct Seccode {

int num; //定义参与者的m值

int order; //定义参与者的编号

struct Seccode \*next;

}Code;

typedef struct Seccode \*LinkList;

2、遍历（获取）和删除函数：

int GetElem(LinkList L,int n,int \*order); //找到链表结点L后面的第n个结点，并获取该节点的编号，保存在地址order中。

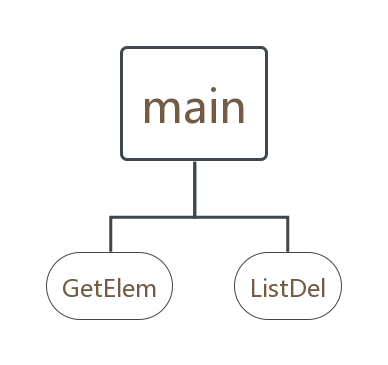
int ListDel(LinkList \*L,int n); // 删除链表结点L后面的第n个节点。

3、主程序流程

a.将给定的数据读入到单链表中，并将链表的尾节点指向头结点。

b.进行循环，按照给定的m遍历链表，找到下一个出圈人的的编号，输出该编号，删除已出圈节点，并将出圈人的m值作为下一个m值进行下一次遍历。直到输出所有的编号。

函数的调用关系图



三、详细设计

链表实现伪码

遍历（获取）函数：

int GetElem(LinkList L,int n,int \*order) { /\*遍历链表获取第n个结点，输出其编号，并返回该节点的m值供程序使用\*/

int i;

LinkList p;

p = L->next;

if(n == 1) {

for(i = 0;i < length;i++) {

p = p->next;

}

}

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

p = p->next;

}

\*order = p->order;

return p->num;

}

删除函数：

int ListDel(LinkList \*L,int n) { //删除链表L结点后的第n个结点

int i;

LinkList p,s;

p = \*L;

if(n == 1) {

for(i = 0;i < length;i++) {

p = p->next;

}

}

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

p = p->next;

}

s = p->next;

\*L = s;

p->next = s->next;

return 1;

}

主程序处理部分：

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

if(i == 0) {

if(m == 1) { /\*若初始m为1，则首编号者出圈，这

里因p已指向首节点，无法直接取值和删除，故遍历整个链表进行取值删除\*/

t = GetElem(p,n,order);

q = &p;

ListDel(q,n);

length--;

}

else {

t = GetElem(p,m-1,order); /\*若初始m不为1，则删除第m个报数者，并返回他的编号存在t变量中，将其m作为新m值\*/

q = &p;

ListDel(q,m-1);

length--;

}

}

else { //根据第一位出圈者输出后的m值删除其后第m个参与者

t = GetElem(p,m,order);

q = &p;

ListDel(q,m);

length--;

}

m = t; //将出圈者的m值作为新的m值

printf("%d ",\*order); //输出出圈者编号

}

return 0;

}

四、调试分析

1、调试问题

a.初期实现程序时，创建指针未初始化，为悬空指针，导致程序报错，后利用malloc函数及相关变量进行了指针初始化，解决了该问题。

b.误以为指针形参在函数内进行赋值操作可以改变传入指针的指向，后发现结果与预期

不匹配，后通过分析函数内部及主函数指针指向的变化发现问题并解决。

1. 算法的时空复杂度分析和改进设想

算法的时间复杂度为O（n\*average(m)），空间复杂度约为16n个字节，目前存在空间复杂度较高的问题，后期通过优化遍历和删除函数，比如用一个函数来实现这两个功能，可以减少循环的次数，降低算法的时间复杂度。

1. 经验和体会

熟记指针操作，如悬空指针不能直接参与运算，指针形参传入不能改变指向等。对环形操作使用循环链表较为简易，实现更为清晰。

五、用户使用说明

用户只需运行所给文件夹下exe程序，按照n、参与者的m值的顺序输入数据，即可得到输岀结果。

六、测试结果

输入样例

7 20

3 1 7 2 4 8 4

输出样例

6 1 4 7 2 3 5

输入样例

5 1

1 20 9 5 7

输出样例

1 2 4 5 3

1. 附录

#include<stdio.h>

#include<stdlib.h>

int length = 0; //定义全局变量：现有的未出圈人员数量

typedef struct Seccode { //创建单链表，存储参与者的编号和m值

int num;

int order;

struct Seccode \*next;

}Code;

typedef struct Seccode \*LinkList;

int GetElem(LinkList L,int n,int \*order) { /\*遍历链表获取第n个结点，输出其编号，并返回该节点的m值供程序使用\*/

int i;

LinkList p;

p = L->next;

if(n == 1) {

for(i = 0;i < length;i++) {

p = p->next;

}

}

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

p = p->next;

}

\*order = p->order;

return p->num;

}

int ListDel(LinkList \*L,int n) { //删除链表L结点后的第n个结点

int i;

LinkList p,s;

p = \*L;

if(n == 1) {

for(i = 0;i < length;i++) {

p = p->next;

}

}

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

p = p->next;

}

s = p->next;

\*L = s;

p->next = s->next;

return 1;

}

int main() {

int i,n,m,t,temp = 1; /\*整型变量t用来存储当前出圈人的m值，temp

为首编号，i为循环变量，n为参与者数量\*/

int \*order; //定义位置指针，指向当前结点的编号。

LinkList head,p,\*q; /\*定义头结点head，指向当前参与者的指针p，以

及指向p的指针q\*/

order = &temp;

head = (LinkList)malloc(sizeof(Code));

p = (LinkList)malloc(sizeof(Code));

scanf("%d %d",&n,&m);

length = n;

scanf("%d",&head->num);

head->next = p;

for(i = 0;i < n-1;i++) { //依次输入各参与者的m值

scanf("%d",&p->num);

p->order = i+2;

p->next = (LinkList)malloc(sizeof(Code));

p = p->next;

}

p->num = head->num;

p->order = 1;

p->next = head->next; //将单链表首尾相连，构成循环链表

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

if(i == 0) {

if(m == 1) { /\*若初始m为1，则首编号者出圈，这

里因p已指向首节点，无法直接取值和删除，故遍历整个链表进行取值删除\*/

t = GetElem(p,n,order);

q = &p;

ListDel(q,n);

length--;

}

else {

t = GetElem(p,m-1,order); /\*若初始m不为1，则删除第m个报数者，并返回他的编号存在t变量中，将其m作为新m值\*/

q = &p;

ListDel(q,m-1);

length--;

}

}

else { //根据第一位出圈者输出后的m值删除其后第m个参与者

t = GetElem(p,m,order);

q = &p;

ListDel(q,m);

length--;

}

m = t; //将出圈者的m值作为新的m值

printf("%d ",\*order); //输出出圈者编号

}

return 0;

}