

# 实习报告

实验题目：约瑟夫环

计算机学院3班，雷雨轩，PB18111791

完成日期：2019年10月27日

## 一. 实验要求

1. 约瑟夫问题的一种描述是：编号为  $1, 2, \dots, n$  ( $n \leq 30$ ) 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。
2. 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个人的编号。
3. 基本要求：
  - (1) 使用命令行进行传参  
(命令行格式为<可执行程序名><人数  $n$ ><初始的 报数上限><密码 1>...<密码  $n$ >)，
  - (2) 当除可执行程序名外没有参数时，将继续执行程序并提示用户输入这些参数。
  - (3) 输入正确的参数后，将输出结果导出到文件中（文件保存路径为 D:\Microsoft Visual Studio\MyProjects\wf\_expe\_1\result1.txt，可根据需要修改）
4. 选做要求：当命令行参数不全或输入不正确时，会报错，并要求重新输入全部参数。
5. 选做要求：用顺序表实现约瑟夫环。
6. 测试数据：  
    输入：7 20 3 1 7 2 4 8 4  
    输出 6 1 4 7 2 3 5

## 二. 设计思路

### (一) 循环链表

#### 1. 抽象数据类型：

**ADT** CircularList{

数据对象：  $D = \{a_i \mid a_i \in \text{ElemSet}, i=1, 2, \dots, n, n \geq 0\}$

数据关系：  $R_1 = \{ \langle a_{i-1}, a_i \rangle \mid a_{i-1}, a_i \in D, i=2, \dots, n, n \geq 0 \}$

基本操作：

    InitList(LinkList& L, LinkList& P, char\* b[], int& n, int& m)

    操作结果：构造一个循环链表，每个节点为一个人的信息

    ListDelete(LinkList& L, LinkList& P, int i)

    操作结果：删去链表中编号为  $i$  的人

#### 2. 设计思路：

利用循环链表模拟人报数并出列（即删除对应的节点）

#### 3. 模块：

##### 1). 主程序模块：

调用链表初始化函数；

模拟报数过程

将所得结果（用数组存放）输入到文件中

##### 2) 循环链表模块：

完成初始化链表及删去节点两个基本操作

## (二) 顺序表实现

### 1. 抽象数据类型:

**ADT List{**

数据对象:  $D=\{a_i \mid a_i \in \text{ElemSet}, i=1,2,\dots,n, n \geq 0\}$

数据关系:  $R_1=\{\langle a_{i-1}, a_i \rangle \mid a_{i-1}, a_i \in D, i=2,\dots,n, n \geq 0\}$

基本操作:

InitList(SqList& L, char\* b[], int& m, int s)

操作结果: 构造一个顺序表, 每个节点为一个人的信息

### 2. 设计思路:

利用顺序链表模拟人报数并出列 (用打标记的方法代表删除对应的节点), 并通过求余来保证在圈内循环

### 3. 模块:

#### 1). 主程序模块:

调用链表初始化函数;

模拟报数过程

将所得结果 (用数组存放) 输入到文件中

#### 2) 顺序表模块:

判断命令行参数是否出错, 解决后完成初始化链表

## 三. 关键代码讲解

### (一) 循环链表实现

#### 1. 链表节点定义:

```
typedef struct LNode {  
    int password; //密码  
    int serial;   //编号  
    struct LNode* next;  
} LNode, * LinkList;
```

#### 2. InitList函数

```
void InitList(LinkList& L, LinkList& P, char* b[], int& n, int& m) {  
    //n: 人数; m: 初始报数上限; L, P分别为头指针, 尾指针; b为指针数组, 传过来的是命令行参数的地址
```

功能:

1) 对命令行是否传来参数判断, 若没有参数, 提示用户输入所有参数, 再继续执行初始化链表操作; 若有参数, 则将参数 (字符串) 转化为int型

2) 初始化

```

//初始化循环链表, 每个节点装密码及编号
L = new LNode;
L->password = p[0];
L->serial = 1;
L->next = L;
LinkedList o = L;
for (int i = 1; i < n; i++) {
    LinkedList q = new LNode;
    q->password = p[i];
    q->serial = i + 1;
    q->next = L; //尾结点next指针始终指回头结点
    o->next = q;
    o = q;
}
P = o;
}

```

3. ListDelete函数:

```

void ListDelete(LinkedList& L, LinkedList& P, int i) { //删去编号为i的人
    LinkedList p = L;
    LinkedList q = L;
    if (p->serial == i) { //删去的是头结点
        if (p->next == L) { //只剩头结点则将头结点的next指针指向NULL, 表示头结点被删去
            p->next = NULL;
        }
        else {
            L = p->next;
            P->next = L;
            delete p;
        }
    }
    else {
        p = p->next;
        while (p->serial != i && p!=L) {
            p = p->next;
            q = q->next;
        }
        if (P == p) { //删去的是尾结点
            P = q;
        }
        q->next = p->next;
        delete p;
    }
}

```

4. main函数:

完成报数模拟, 并将结果, 即出列顺序输出到文件中

```

int main(int a, char* b[])
{
    //n: 人数; m: 初始报数上限; L, P分别为头指针, 尾指针
    int m, n;
    Linklist L, P;
    InitList(L, P, b, n, m);    //调用函数初始化链表
    Linklist p = L;
    Linklist q = L;
    int sequence[30] = { 0 };    //依次存放出列人的编号
    int j = 0;    //对应sequence中已存放人数
    int i = 0;    //当前报数值
    while (L->next != L) {
        for (i = 1; i < m; i++) { //报数
            p = p->next;
        }
        m = p->password;    //报数到出列人处, 更新报数上限值m
        sequence[j++] = p->serial;    //记录出列人编号
        q = p->next;    //指向下一个开始
        ListDelete(L, P, p->serial);    //删去出列人所在节点
        p = q;
    }
    sequence[j++] = L->serial;
    delete L;
    FILE* stream1;    //文件输入流, 将标准输出流中的数据全部传到文件中
    freopen_s(&stream1, "D:\\Microsoft Visual Studio\\MyProjects\\wf_expe_1\\result.txt",
        "w", stdout);
    for (int i = 0; i < n; i++) {
        cout << sequence[i] << endl;
    }
    fclose(stdout);
}

```

## (二) 顺序表实现

### 1. 链表及节点定义

```

typedef struct { //每个节点存储空间需要存密码以及标志符(判定此节点是否出列)
    int password;
    int flag;
} LNode;

typedef struct {
    LNode* elem;    //存储空间基址
    int length;    //表中人数
} SqList;

```

### 2. InitList函数

功能:

1) 对输入有无参数、参数有无错误、参数个数是否正确做判断, 出错则用catch捕捉后提示用户重新输入所有参数

2) 建立顺序表并将链表的每个节点赋值初始化

3. main函数:

```

int main(int a, char* b[]) {
    int m, n;    //m为报数上限, n为最初人数
    SqList L;
    InitList(L, b, m, a); //初始化顺序表
    n = L.length;
    int sequence[30] = { 0 };
    int j = 0;
    int i = 0; //当前报数值
    int p = 0; //指针
    while (n != 1) {
        for (int i = 1; i < m; i++) { //报数
            while (L.elem[p].flag != 1) { //判断当前元素是否已经被打了标记(已经出列)
                p = (p + 1) % L.length; //直到指向一个未出列的下一个元素
            }
            p = (p + 1) % L.length; //指向下一个元素(可能该元素已出列)
        }
        while (L.elem[p].flag != 1) { //确保报数上限的时候指向未出列的元素
            p = (p + 1) % L.length;
        }

        m = L.elem[p].password; //更新报数上限值
        sequence[j++] = p + 1; //存入出列人编号
        L.elem[p].flag = 0; //给该出列人打标记, 表示已出列
        n--; //记录当前还有多少人未出列
        p = (p + 1) % L.length;
    }
}

```

## 四。调试分析

### (一) 循环链表实现:

#### 1. 遇到的问题:

- (1) 命令行传入参数与否的判断
- (2) 将字符串转为int的函数
- (3) 初始化链表时要注意循环链表与一般链表的区别: 尾结点的next指针始终指向头结点
- (4) ListDelete函数: 删去节点分情况讨论要讨论完全, 头尾节点的情况要单独操作
- (5) 数据输入到文件的方法
- (6) 报数模拟过程中, 要理解到可以在列中只剩最后一人时就退出, 不用再报数了。

#### 2. 时空复杂度分析:

- (1) 空间复杂度:  $O(n)$ : 只在InitList函数中根据参与人数n来额外分配空间
- (2) 时间复杂度:
  - InitList函数: $O(n)$ , 只负责依次初始化n个节点
  - ListDelete函数: $O(n)$ : 最多遍历整个链表就可找到要删去的节点
  - main函数: $O(n^2)$ : n-1次while循环以及每次循环里的ListDelete操作

### (二) 顺序表实现:

#### 1. 遇到的问题:

- (1) 如何判断参数是否有缺漏 (利用argc)
- (2) 如何对错误进行处理 (利用try-catch块)
- (3) 报数模拟的逻辑一定要清晰, 循环时必须是指向下一个未出列的人


#### 2. 时空复杂度分析:

- (1) 空间复杂度:  $O(n)$ : 只在InitList函数中根据参与人数n来额外分配空间
- (2) 时间复杂度:
  - InitList函数: $O(n)$ , 只负责依次初始化n个节点
  - main函数: $O(n^2)$ : n-1次while循环以及每次循环里的报数操作

## 五.代码测试

### (一) 循环链表实现:

```
D:\Microsoft Visual Studio\MyProjects\wf_expe_1\Debug>wf_expe_1 7 20 3 1 7 2 4 8 4
6 1 4 7 2 3 5
D:\Microsoft Visual Studio\MyProjects\wf_expe_1\Debug>wf_expe_1
请输入人数及初始报数上限:
7 20
请依次输入每个人的密码:
3 1 7 2 4 8 4
6 1 4 7 2 3 5
```



命令行输入：先进入exe文件所在路径，再使用命令行格式运行程序，参数之间用空格隔开。图为正确输入，以及没有参数时的情况。

### (二) 顺序表实现

```
D:\Microsoft Visual Studio\MyProjects\wf_expe_1\Debug>wf_expe_1 7 20 3 1 7 2 4 8 4
6 1 4 7 2 3 5
D:\Microsoft Visual Studio\MyProjects\wf_expe_1\Debug>wf_expe_1
没有参数输入
请输入人数:
7
请输入初始报数上限:
20
请输入每个人的密码:
3
1 7 2 4 8 4
6 1 4 7 2 3 5
D:\Microsoft Visual Studio\MyProjects\wf_expe_1\Debug>wf_expe_1 7 20 1 2
少参数, 请重新输入
请输入人数:
7
请输入初始报数上限:
20
请输入每个人的密码:
3 1 7 2 4 8 4
6 1 4 7 2 3 5
D:\Microsoft Visual Studio\MyProjects\wf_expe_1\Debug>wf_expe_1 7 20 1 2 a b 2 23 6
Invalid argument
请输入人数:
7
请输入初始报数上限:
20
请输入每个人的密码:
3 1 7 2 4 8 4
6 1 4 7 2 3 5
```



文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

6  
1  
4  
7  
2  
3  
5

命令行输入：先进入exe文件所在路径，再使用命令行格式运行程序，参数之间用空格隔开。图为正确输入;没有参数;参数不够；参数值错误的四种情况

## 六. 实验总结

这次实验让我熟悉了约瑟夫环问题及其多种解法。报数模拟的不同复杂程度使我对顺序表和链式表的区别及优势更清楚，如链表只需指针依次动，而且节点可以真正删去；但在顺序表中，节点删去只能打标记，循环也要通过求余来模拟。此外通过检索以及调试代码，也学会了判断输入是否符合要求，如何使用命令行传参以及将字符串转化为数字等等。

写实验报告的过程也是对自己写代码时思路的一个更加全面细致、冷静地回顾，让我加深印象，收获颇丰。

## 七. 附录

wf\_expe\_1.cpp: 循环链表实现约瑟夫环

wf\_2.cpp:顺序表实现约瑟夫环