数据结构课程设计

项目说明文档

约瑟夫问题

|  |  |
| --- | --- |
| 作者姓名： | 高逸轩 |
| 学 号： | 2053385 |
| 指导教师： | 张 颖 |
| 学院专业： | 软件学院 软件工程 |



同济大学

Tongji University

# 1项目分析

## 1.1 项目需求分析

本程序完成了以下需求：

* 功能完善

实现了建立、增加、查询、修改、删除的基本功能。

* 执行效率高

能够存储的数据条数应该足够多，使用的数据结构和算法应该具有较低的时间和空间复杂度。

* 健壮性

保证了输入的数据类型和范围合理时才会进行程序的运行

* 代码复用性

在本题中，链表的代码大部分基于上一题目中复用，为方便老师批改，将其整合到一个.cpp文件中

# 1.2 项目要求

### 1.2.1 功能要求

本项目的实质是完成使用循环链表模拟约瑟夫问题的求解过程。

### 1.2.2 输入格式

依次游戏总人数、游戏开始的位置、死亡数字（每次报到这个数字则杀死一名玩家）、剩余人数。

### 1.2.3 输出格式

每一次杀死玩家的位置、游戏结束后存活玩家的位置。

### 1.2.4 项目示例

# 2 项目设计

## 2.1 数据结构设计

本题目要求使用循环单链表来实现约瑟夫问题的模拟。

本系统实现的链表结构各种操作的时间复杂度如下：

* 插入操作：O(n)
* 删除操作：O(n)
* 查询操作：O(n)
* 遍历操作：O(n)

## 2.2 类设计

经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（ListNode）与链表类（List），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。以下为本程序的设计。

### 2.2.1 结点类（LinkedListNode）

// 节点

struct LinkedListNode

{

// 构造函数，初始化编号

LinkedListNode(int i = 0) { num = i; next = NULL; }

// 输出编号信息

void Display() { cout << num << endl; }

int num; // 编号

LinkedListNode\* next; // 指向下个位置

};

### 2.2.2 循环链表类（CircledLinkedList）

// 循环链表类

class CircleLinkedList

{

friend void JosephusSolve(CircleLinkedList& circleJosephus, int n, int s, int m, int k);

public:

// 构造函数，为头结点申请空间，同时构造循环条件

CircleLinkedList()

{}

// 析构函数，释放空间

~CircleLinkedList()

// 计算循环链表长度

int Length()

// 定位函数，返回表中第i个元素的地址

LinkedListNode\* Locate(int i)

// 寻找链表中是否含有x编号的人，搜索成功返回该节点位置，否则返回-1

int Search(int x)

// 删除第i个元素,x返回该元素的值

bool Remove(int i, LinkedListNode& x)

// 将新元素x插入到第i个节点后

bool Insert(int i, LinkedListNode& x)

// 将链表中信息依次输出

void Display()

private:

LinkedListNode\* head; // 链表头指针

};

## } 2.3 项目功能流程图

# 3. 核心代码介绍

本题采用的数据结构为带空头节点的单循环链表，其所用代码大部分来源于上一题中单链表部分，主要修改内容为**判断链表遍历结束的标志**。单链表判断结束的标准是：当前节点地址为NULL，即已经循环到链表尾部。在单循环链表中，由于最后一个节点的next指针指向head节点，所以应该对应将判断条件更改。以下为代码：

## 3.1 建立链表

在构造函数中，为头结点申请空间。同时将head的next指针指向自己，构造循环条件。

## 3.2 销毁链表

析构函数中，我们从头结点开始向后遍历，在向后遍历的过程中同时释放经过节点的空间。由于采取了空表头链表的写法，最后我们还需要释放头节点的空间。

## 3.3 搜索数据

以num为关键字，对链表进行遍历。若搜索成功返回节点位置，反之则返回-1。

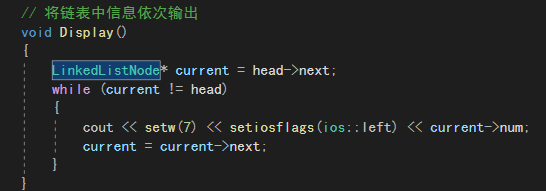
## 3.4 定位数据

在定位函数中，由于循环链表判断结尾的特殊性，遍历时需要从第一个非空节点开始遍历。所以在对位置进行合法性判断时，需要对头结点进行特判。

## 3.5 删除数据

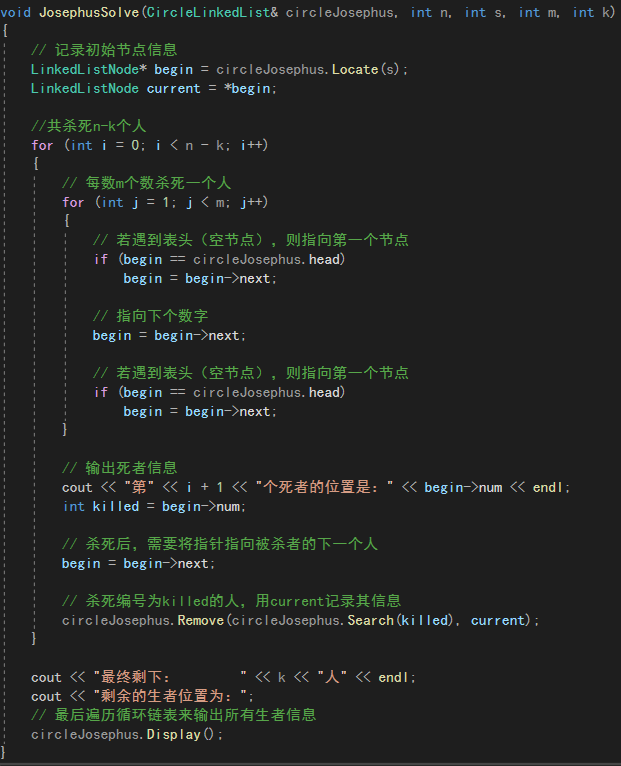
## 3.6 插入元素

## 3.7 展示链表内容



遍历链表，根据要求的格式输出各个节点内容。

## 3.8 约瑟夫问题

在利用循环链表进行约瑟夫问题求解前，我们需要先将编号为[0,n-1]的人按顺序插入链表中，然后调用JosephusSolve函数进行模拟求解：

在约瑟夫问题求解函数中，用begin记录初始节点的信息，用current记录每次当前被删除的节点信息。

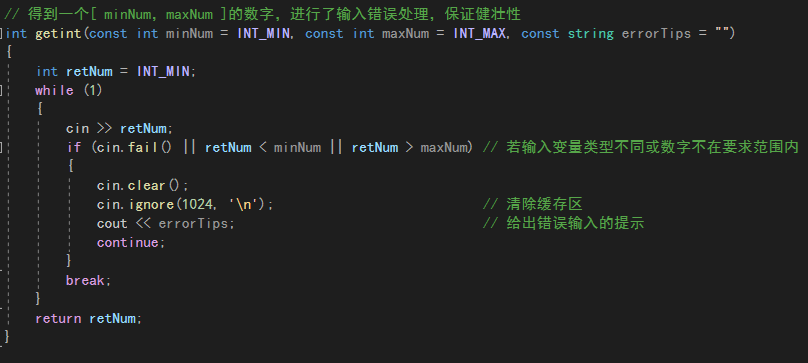
共循环n-k次，删除n-k个节点。每次循环m次（每数m个数杀死一个人），循环遍历链表。由于头结点为空，在每次遍历结束后，需要判断当前位置是否是head节点，若是，需要将指针向下移动一位，指向head的下一个非空位置。每次删除节点后，需要借助current记录下被删除的节点信息，并输出。

在结束n-k次循环后，展示循环链表中的剩余节点信息。

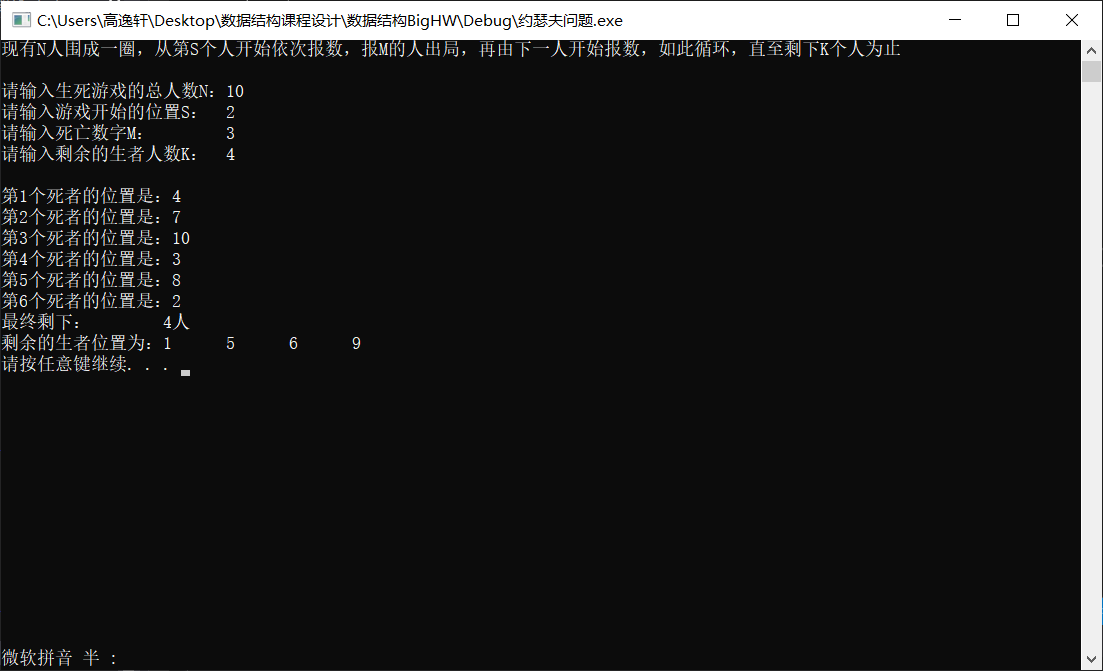
# 4 项目测试

## 4.1 健壮性测试

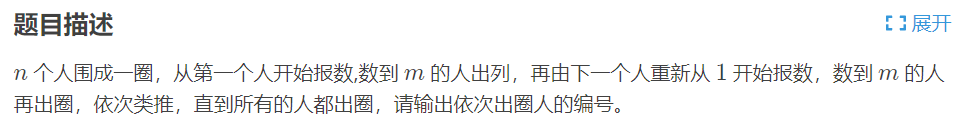
本次程序的错误输入处理主要分为两种：一类为输入的信息为字符等不符合所需数据类型；另外一类为输入的数据不合法，如剩余玩家大于初始人数、开始位置小于0等。这类通过自行编写的getint()函数进行处理。



## 4.2 功能测试

为方便老师测试，提供了文件2\_test.txt，内含一组数据，测试了本程序功能。以下为一组测试数据：

# 5 心得与体会

 约瑟夫问题是一个经典的问题，在高中竞赛经历中，我采用的是数组方式来解决：

在本次项目中，我采取题目要求的方式：循环链表，这简化了数组模拟方式中取模来定位的方式。同时，我更深刻的由单链表改进而来的单循环链表的过程，以及最重要的区别：判断遍历结束的标志的不同。

另外，健壮性的保证也是重要内容。在本次作业中，我采用了自己编写的getint，以及其他操作，处理了错误输入、空间异常等问题，保证了程序的稳定性。

由于篇幅原因，报告内还有很多内容与解释没有展示，请老师和助教老师再移步源程序，在其中的注释写了每一步过程的详解。