项目说明文档

数据结构课程设计

——约瑟夫生者死者游戏

作 者 姓 名： 刘畅

学 号： 2054164

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目 录

[1 分析 1](#_Toc84669565)

[1.1 背景分析 1](#_Toc84669566)

[1.2 功能分析 1](#_Toc84669567)

[2 设计 2](#_Toc84669568)

[2.1 数据结构设计 2](#_Toc84669569)

[2.2 类结构设计 2](#_Toc84669570)

[2.3 成员与操作设计 2](#_Toc84669571)

[3 实现 3](#_Toc84669572)

[3.1 插入功能的实现 3](#_Toc84669573)

[3.1.1 实现方法 3](#_Toc84669574)

[3.1.2 插入功能核心代码 3](#_Toc84669575)

[3.2 删除功能的实现 3](#_Toc84669576)

[3.2.1 实现方法 3](#_Toc84669577)

[3.2.2 删除功能核心代码 3](#_Toc84669578)

[3.3 清空功能的实现 4](#_Toc84669579)

[3.3.1 实现方法 4](#_Toc84669580)

[3.3.2 清空功能核心代码 4](#_Toc84669581)

[3.4 main.cpp代码逻辑 4](#_Toc84669582)

[3.4.1 具体实现过程 4](#_Toc84669583)

[3.4.2 总体系统核心代码 4](#_Toc84669584)

[4 测试 5](#_Toc84669585)

[4.1 测试1 5](#_Toc84669586)

[4.2 测试2 6](#_Toc84669587)

[4.3 测试3 6](#_Toc84669588)

# 1 分析

## 1.1 背景分析

约瑟夫生者死者游戏的大意是：30个旅客同乘一条船，因为严重超载，加上风高浪大危险万分；因此船长告诉乘客，只有将全船一半的旅客投入海中，其余人才能幸免于难。无奈，大家只得统一这种方法，并议定30个人围成一圈，由第一个人开始，依次报数，数到第9人，便将他投入大海中，然后从他的下一个人数起，数到第9人，再将他投入大海，如此循环，直到剩下15个乘客为止。问哪些位置是将被扔下大海的位置。

## 1.2 功能分析

在此我们使用单循环链表建立一个简单的模型，假如N个旅客排成一个环形，依次顺序编号1, 2, …, N。从某个指定的第S号开始。沿环计数，每数到第M个人就让器出列，且从下一个人开始重新计数，继续进行下去。这个过程一直进行到剩下K个旅客为止。其中N，S，M，K由用户输入。

由于N个旅客是站成一个圈，因此本题可以使用循环链表。首先需要建立一个含N个元素的链表，采用一个个插入的方式，要注意这里最后一个结点指向的不是NULL而是附加头结点。

在实现过程中主要需要进行向前移动M个单位以及删除某个人的操作，前者只需令某个结点指向下一个结点，后者的删除操作与普通链表相似，只需让target结点的前一个结点指向target所指向的结点，再将target删除，并输出target的数据。

最后还需要遍历一遍链表输出所有剩下的结点。

为了使链表为空和不为空时添加元素保持一致，添加一个附加尾结点，在初始化的时候同时指向对方。这样在插入新元素的时候就能统一操作。

循环链表在遍历的时候会遇到附加头尾结点的情况，由于附加头尾结点并不存储有效的数据，因此在循环到它们的时候需要跳过，因此设置一个成员函数CircList::CircLinkNode<T>\* Next(CircLinkNode<T>\* cur)，用来封装指向下一结点的操作。

# 2 设计

## 2.1 数据结构设计

由于N个旅客围成一个圈之后，没有明确的前后序号顺序，因此采用循环链表。

## 2.2 类结构设计

在CircList.hpp中由于采用template模板编程，声明和实现无法分文件编写，故两者放在同一文件并命名为.hpp文件，其中包含两个类，即循环链表结点类（CircLinkNode）和循环链表类（CircList）。

## 2.3 成员与操作设计

**循环链表结点类（CircLinkNode）**

**public：**

CircLinkNode() :\_data(T()), \_next(NULL) { }

T \_data;

CircLinkNode<T>\* \_next;

除结点

**循环链表类（CircList）**

**public:**

CircList();//构造函数，初始化头尾指针，让它们互相指向对方

~CircList();//析构函数

void Erase();//删除整个链表

CircLinkNode<T>\* getNode(int pos);//获得第pos个链表结点

bool Insert(T& t);//在链表末尾添加元素

bool Remove(CircLinkNode<T>\* pre);//移除pre结点下一个结点

CircLinkNode<T>\* Next(CircLinkNode<T>\* cur);//封装循环链表的指向下一结点操作，防止尾结点再次判断

void PrintList();//打印链表

**private:**

CircLinkNode<T>\* \_first, \* \_last;//附加头、尾结点

# 3 实现

## 3.1 插入功能的实现

### 3.1.1 实现方法

给定插入元素，将其添加在循环链表末尾：先根据该元素创建循环链表结点，通过遍历链表找到最后一个元素，将最后一个元素指向新结点，新结点指向附加尾结点。

若链表本身为空，即附加头尾结点互相指向对方，操作方法和上述一致，因此附加尾结点的设置可以简化代码，是之更为清晰。

插入方法主要使用于构建N个元素环绕站立的抽象结构，即通过循环语句依次插入15个元素。

### 3.1.2 插入功能核心代码

bool Insert(T& t) {

CircLinkNode<T>\* newCircNode = new CircLinkNode<T>;

if (newCircNode == NULL) {return false;}

CircLinkNode<T>\* leftNode = \_first;

while (leftNode->\_next != \_last) {leftNode = leftNode->\_next;}

newCircNode->\_data = t;

leftNode->\_next = newCircNode;

newCircNode->\_next = \_last;

return true;

}

## 3.2 删除功能的实现

### 3.2.1 实现方法

已知要删除的链表结点tar的前一个结点pre，删除下一个节点：先使用内嵌的Next()函数寻找pre下一个结点赋值给tar，然后判断pre是否为链表末尾，若是，则要删除的结点为第一个结点，将first指向第二个结点即可；若不是，则将pre指向tar下一个结点，然后删除目标节点tar。

### 3.2.2 删除功能核心代码

bool Remove(CircLinkNode<T>\* pre) {

if (pre == NULL || Next(pre) == NULL) {return false;}

CircLinkNode<T>\* tar = Next(pre);

if (pre->\_next == \_last) {\_first->\_next = tar->\_next;}

else {pre->\_next = tar->\_next;}

delete tar;

return true;

}

## 3.3 清空功能的实现

### 3.3.1 实现方法

清空链表不删除附加头尾结点，故只要取到附加头结点，然后依次遍历下去直到附加尾结点，将两者中间的所有结点删除，并且使附加头结点指向附加尾结点即可。

析构函数只需在此基础上删除附加头尾结点即可。

### 3.3.2 清空功能核心代码

void Erase() {

CircLinkNode<T>\* pos;

while (\_first->\_next != \_last) {

pos = \_first->\_next;

\_first->\_next = pos->\_next;

delete pos;

}

}

## 3.4 main.cpp代码逻辑

### 3.4.1 具体实现过程

先将N个人依次插入到循环链表之中，然后取得第S-1个人设为pre即前驱结点（因为删除结点需要传入其前一个结点），由于最后需要剩下K个人，所以需要循环N-k+1次，每次循环都将前驱结点pre往后移动M-1次（因为是每数M个人就删除，包含自身），再将pre下一结点删除即直接将pre传入删除函数。

等到循环结束，将链表打印即可。

### 3.4.2 总体系统核心代码

CircList<int> circlist;

CircLinkNode<int>\* pre;

int N, S, M, K;

cin >> N >> S >> M >> K;

for (int i = 1; i <= N; i++) {

circlist.Insert(i);

}

pre = circlist.getNode(S - 1);

for (int i = 1; i < N - K + 1; i++) {

for (int j = 1; j < M; j++) {

pre = circlist.Next(pre);

}

cout << "第" << i << "个：" << pre->\_next->\_data << endl;

circlist.Remove(pre);

}

circlist.PrintList();

# 4 测试

## 4.1 测试1

**测试用例：**30 1 9 15

**实验结果：**

第1个：9

第2个：18

第3个：27

第4个：6

第5个：16

第6个：26

第7个：7

第8个：19

第9个：30

第10个：12

第11个：24

第12个：8

第13个：22

第14个：5

第15个：23

剩下的人编号：1 2 3 4 10 11 13 14 15 17 20 21 25 28 29

## 4.2 测试2

**测试用例：**18 9 8 1

**实验结果：**

第1个：16

第2个：6

第3个：14

第4个：5

第5个：15

第6个：8

第7个：0

第8个：12

第9个：9

第10个：4

第11个：3

第12个：7

第13个：11

第14个：18

第15个：17

第16个：10

第17个：0

剩下的人编号：13

## 4.3 测试3

**测试用例：**24 24 3 5

**实验结果：**

第1个：2

第2个：5

第3个：8

第4个：11

第5个：14

第6个：17

第7个：20

第8个：23

第9个：3

第10个：7

第11个：12

第12个：16

第13个：21

第14个：0

第15个：9

第16个：15

第17个：22

第18个：6

第19个：18

剩下的人编号：4 10 13 19 24