项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 张诚睿

学 号： 2150998

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](#_Toc122644615)

[1.1 项目背景分析 1](#_Toc122644616)

[1.2 功能分析 1](#_Toc122644617)

[2 设计 1](#_Toc122644618)

[2.1 数据结构设计 1](#_Toc122644619)

[2.2 类结构设计 2](#_Toc122644620)

[2.3 类具体成员与操作设计 2](#_Toc122644621)

[2.4 main函数设计 2](#_Toc122644622)

[3 实现 3](#_Toc122644623)

[3.1 链式队列-进队功能的实现 3](#_Toc122644624)

[3.1.1 功能流程图 3](#_Toc122644625)

[3.1.2 核心代码 3](#_Toc122644626)

[3.2 链式队列-出队功能实现 4](#_Toc122644627)

[3.2.1 功能流程图 4](#_Toc122644628)

[3.2.2 核心代码 5](#_Toc122644629)

[3.3 业务处理顺序结果输出功能实现 5](#_Toc122644630)

[3.3.1 功能流程图 5](#_Toc122644631)

[3.3.2 核心代码 6](#_Toc122644632)

[3.4 总体功能的实现 6](#_Toc122644633)

[3.4.1 总体流程 6](#_Toc122644634)

[3.4.2 总体功能核心代码 6](#_Toc122644635)

[3.5.3 总体功能截屏示例 7](#_Toc122644636)

[4 测试 7](#_Toc122644637)

[4.1 功能测试 7](#_Toc122644638)

[4.1.1一般顾客序列处理结果输出功能测试 7](#_Toc122644639)

[4.2 边界测试 8](#_Toc122644640)

[4.2.1 仅有一个顾客 8](#_Toc122644641)

[4.2.2 所有顾客都是奇数 8](#_Toc122644642)

[4.2.3 所有顾客都是偶数 8](#_Toc122644643)

[4.3 出错测试 9](#_Toc122644644)

[4.3.1 顾客总数不是正整数 9](#_Toc122644645)

[4.3.2 顾客总数是字符 9](#_Toc122644646)

[4.3.3 顾客编号是字符 10](#_Toc122644647)

# 1 分析

## 1.1 项目背景分析

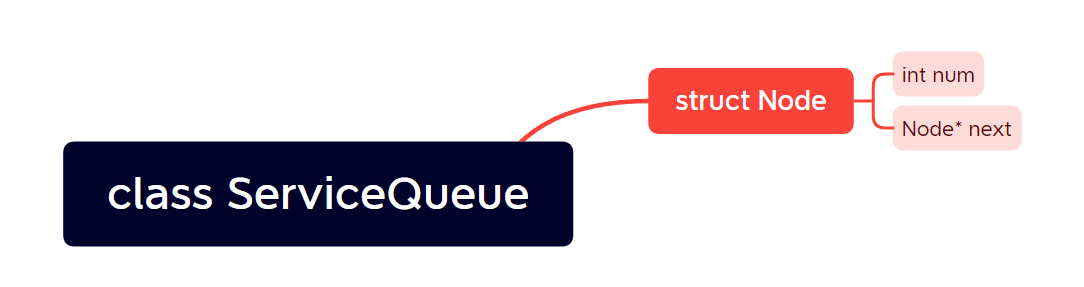
银行有两个窗口，处理业务的速度不一致，按照不同编号将顾客分成两组，分别在两个窗口办理业务，这样的情景模拟了真实银行的工作状态。一般地，顾客在到达银行后会获得取号码，按顺序前往窗口办理业务，本项目用输入的先后模拟顾客领取号码的顺序，用奇偶数给顾客划定窗口，是对真实银行工作情景的抽象模拟

## 1.2 功能分析

本项的目的在于实现对输入的顾客序列进行安排，输出办理业务的先后顺序。其中，奇数号顾客在A窗口办理业务，偶数号顾客在B窗口办理，A窗口办理业务的速度是B窗口的两倍，按照如上规则模拟后，将时间顺序上的结果序列输出，每个顾客序号用空格隔开，且最后一个顾客号后没有多余的空格。

# 2 设计

## 2.1 数据结构设计



核心数据结构设计如上，通过类-ServiceQueue实现同时控制两个队列，队列均是利用struct Node构成的链表模拟的，结点中用int num来存放顾客号码，输出时，不再建立新的输出队列，直接对奇偶两个队列做出队、输出等操作

## 2.2 类结构设计

如上所述，本项目采用struct Node描述链表结点类（LNode），这样使得链表结点类ServiceQueue（LinkList）可以访问链表结点。

所有计算数据放入哪个队列、输出队列的操作均在ServiceQueue类完成，它的私有成员还包括记录两个队列队头队尾的指针和一些细分的函数等。

## 2.3 类具体成员与操作设计

**链表结点类**

struct Node {

int num;

Node\* next;

};

**链式栈类**

class ServiceQueue {

private:

int TotalNum;

Node\* frontEven, \* rearEven; // 偶数队头队尾

Node\* frontOdd, \* rearOdd; // 奇数队头队尾

bool EnQueue(int num, Node\*& rear, Node\*& front); // 入队

bool DeQueue(int& num, Node\*& front); // 出队

void makeEmpty();

bool GetNumber(int& number); // 获取客户编号

bool IsEmpty(Node\*& front);

void EvenOut();

void OddOut();

public:

ServiceQueue() :TotalNum(0), frontEven(NULL), rearEven(NULL), frontOdd(NULL), rearOdd(NULL) {};

~ServiceQueue() { makeEmpty(); };

bool GetCustom(); // 获取客户信息

void Out(); // 输出服务顺序结果

};

## 2.4 main函数设计

首先建立ServiceQueue对象Bank，之后获取顾客序号的输入，如输入错误，则提示重新输入直至输入正确，最后计算结果并输出，主函数返回0。

# 3 实现

此处仅介绍核心功能的实现

## 3.1 链式队列-进队功能的实现

### 3.1.1 功能流程图

### 3.1.2 核心代码

if (num < 0) // 错误编号

{

return false;

}

// 正确编号时，入队

if (front == NULL) // 空队

{

Node\* tmp = new Node;

if (tmp == NULL)

{

cout << "Memory Wrong" << endl;

exit(EXIT\_FAILURE);

}

tmp->num = num;

tmp->next = NULL;

front = tmp;

rear = tmp;

}

else // 非空队

{

Node\* tmp = new Node;

if (tmp == NULL)

{

cout << "Memory Wrong" << endl;

exit(EXIT\_FAILURE);

}

tmp->num = num;

tmp->next = NULL;

rear->next = tmp;

rear = tmp;

}

return true;

## 3.2 链式队列-出队功能实现

### 3.2.1 功能流程图

### 3.2.2 核心代码

if (front == NULL) // 空队

{

return false;

}

else

{

num = front->num;

Node\* tmp = front;

front = front->next;

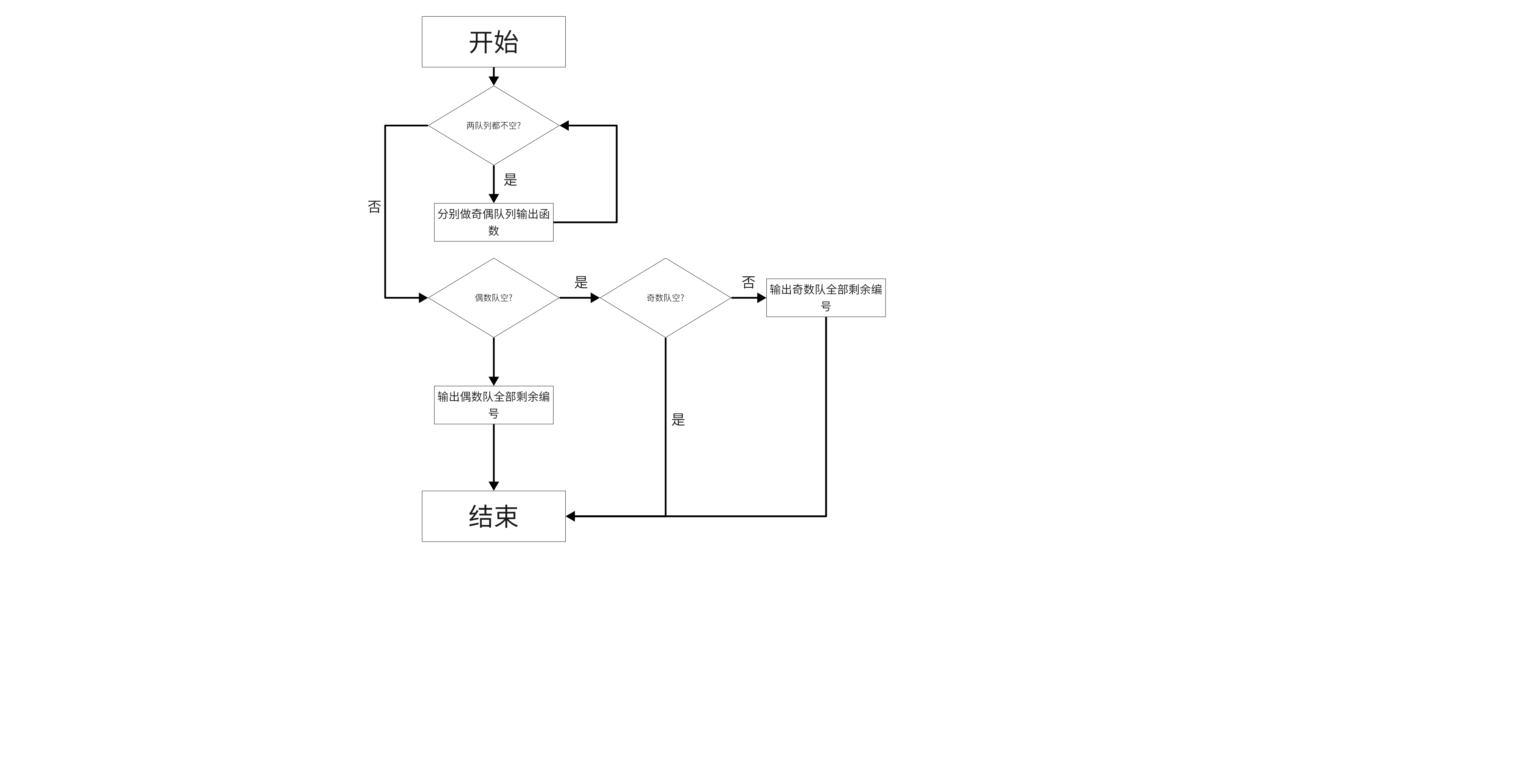
delete tmp;

}

return true;

## 3.3 业务处理顺序结果输出功能实现

### 3.3.1 功能流程图



其中，奇偶队列输出函数分别做相应出队、输出num的操作

### 3.3.2 核心代码

while (!IsEmpty(frontEven) && !IsEmpty(frontOdd)) // 只要都不空，就输出

{

OddOut();

EvenOut();

}

// 有空队，另一队全部输出

if (IsEmpty(frontEven)) // 偶数队空

{

if (IsEmpty(frontOdd))

{

return;

}

else

{

while (!IsEmpty(frontOdd))

{

OddOut();

}

}

}

else // 偶数队不空，奇数队必空

{

while (!IsEmpty(frontEven))

{

EvenOut();

}

}

## 3.4 总体功能的实现

### 3.4.1 总体流程

首先建立ServiceQueue对象Bank，之后获取顾客序号的输入，如输入错误，则提示重新输入直至输入正确，最后计算结果并输出，主函数返回0。

### 3.4.2 总体功能核心代码

int main()

{

ServiceQueue Bank; // 银行服务队列

cout << "开始输入顾客信息" << endl;

while (!Bank.GetCustom())

{

cout << "Costom number wrong, reinput from the beginning" << endl;

}

cout << "处理顺序如下:" << endl;

Bank.Out();

return 0;

}

### 3.5.3 总体功能截屏示例



# 4 测试

## 4.1 功能测试

### 4.1.1一般顾客序列处理结果输出功能测试

**测试用例**：

8 2 1 3 9 4 11 13 15

**预期结果**：

1 3 2 9 11 4 13 15

**实验结果**



输出结果末尾没有多余空格

## 4.2 边界测试

### 4.2.1 仅有一个顾客

**测试用例：**

1 6

**预期结果：**

6

**实验结果：**



输出结果末尾没有多余空格

### 4.2.2 所有顾客都是奇数

**测试用例：**

5 1 3 5 7 9

**预期结果：**

1 3 5 7 9

**实验结果：**



输出结果末尾没有多余空格

### 4.2.3 所有顾客都是偶数

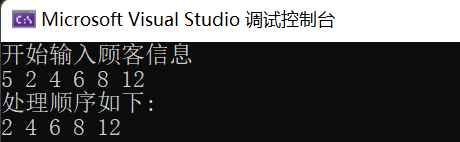
**测试用例：**

5 2 4 6 8 12

**预期结果：**

2 4 6 8 12

**实验结果：**



## 4.3 出错测试

### 4.3.1 顾客总数不是正整数

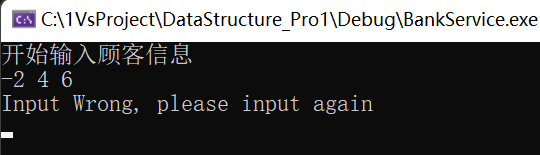
**测试用例：**

-2 4 6

**预期结果：**

输出提示信息，要求重新输入，程序不崩溃

**实验结果：**



### 4.3.2 顾客总数是字符

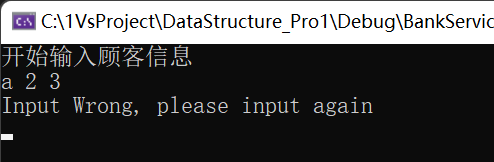
**测试用例：**

a 2 3

**预期结果：**

输出提示信息，要求重新输入，程序不崩溃

**实验结果：**



### 4.3.3 顾客编号是字符

**测试用例：**

4 a b c d

**预期结果：**

输出提示信息，要求重新输入，程序不崩溃

**实验结果：**

