项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 何慧琳

学 号： 2152343

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 分析

## 1.1 背景分析

银行业务系统是一个银行重要的工具，它对于银行的管理者和顾客来说都至关重要，一个基本的银行业务系统应该能够统计各窗口的人流及预计各窗口顾客办理、办结时间或顺序，以便为顾客提供更高效的服务。

随着计算机科学技术的不断成熟，使用计算机对银行业务系统进行管理，具有手工管理所无法比拟的优势。这些优点能够极大地提高银行的效率提高顾客的体验感，也是银行智能化的重要条件。因此，开发一套银行业务系统具有十分重要的意义。

## 1.2 功能分析

作为一个最简易的银行业务系统，首先应该有的功能就是输入顾客的编号序列。其次应当根据编号将每位顾客分配到相应窗口，根据不同窗口不同处理速度，预计每位顾客办结顺序。

综上所述，一个银行业务系统应当能够输入一段序列，将序列分组按顺序存储，按一定顺序输出序列中的编号。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求能够对输入的序列分组存储，并且同一组中先到的顾客先办结，所以需要一个先进先出的存储结构，于是定义了一个队列类（Queue），队列由一个个队列结点组成，所以另外定义了一个队列结点类（QueueNode），用两个队列来储存两个窗口尚未办结的顾客。此外因为两个窗口办理用时不同，需要同时计算每个时刻两窗口办结的顾客，即实现主要的银行业务功能，于是定义了银行业务类（banking\_bussiness），以实现序列的输入、业务办理的输出功能。

## 2.2 类结构设计

**队列结点类（QueueNode）**

一个队列结点应当包含自身的数值和链接到下一个结点的指针。同时它是队列的基本组成结构，为了在队列类中能访问结点的各成员，在该类中将队列类声明为友元。

**私有成员及操作：**

Type data;//结点数据

QueueNode\* link;//链接指针

QueueNode(Type d = 0, QueueNode\* l = NULL) :data(d), link(l) {};//构造函数

**队列类（Queue）**

队列由一个个相链接的结点组成，于是需要两个QueueNode类型的指针，一个指向队列头，以实现先进先出特点的出队头操作，另一个指向队尾，负责结点从后面入队操作。因此还需要结点入队、取队头、删队头的函数。此外为防止删出操作时的访问溢出还需要判断队列是否为空的函数。

**私有成员：**

QueueNode<Type>\* first, \* rear;//队列指针

**公有操作：**

Queue():first(NULL),rear(NULL) {};//构造函数

~Queue();//队列析构函数

void push(const Type item);//结点入队

Type top();//取队头结点

Type pop();//队头出队，返回队头元素值

int IsEmpty()const {return first == NULL;};//判断队列是否为空，空返回1，非空返回0

## void MakeEmpty();//清空队列

**银行业务类（banking\_bussiness）**

队列负责序列的存储，具体的读入、分配顾客序列和顾客出队列的控制操作还需由银行业务类来实现。每次进行业务操作前都要输入本次顾客总数和一段顾客编号序列，这就要求银行业务类有两个私有成员分别存储顾客总数和顾客队列，通过一个其内部的读入函数对这两个成员进行初始化。初始化完成后由业务操作函数对顾客队列进行处理并输出顾客办结序列。

**私有成员：**

int total;//顾客总数

Queue<int> queue;//顾客队列

**公有操作：**

banking\_bussiness() :total(0) {};//银行业务构造函数

void read();//输入顾客队列

void operation();//按照业务处理完成的顺序输出顾客的编号

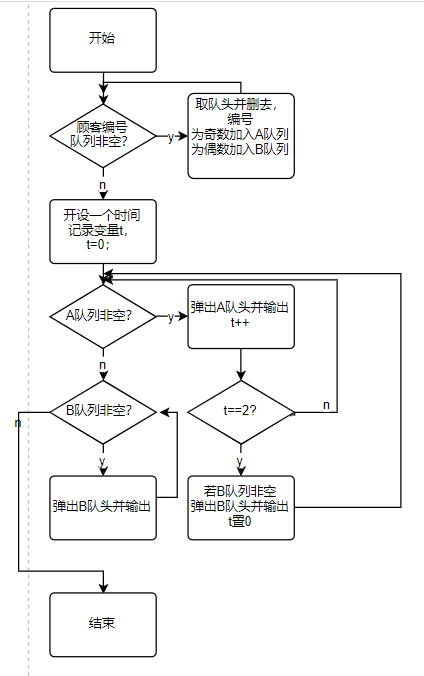
## 2.4 系统设计

系统首先调用main()函数实现对屏幕的初始化，调用banking\_bussiness类中的read()函数输入顾客序列，调用其中的operation()函数进行计算和结果输出。

# 3 实现

## 银行业务功能的实现

### 银行业务功能流程图



### 银行业务功能核心代码

void banking\_bussiness::operation()

{//按照业务处理完成的顺序输出顾客的编号

Queue<int> A\_queue,B\_queue;//A、B窗口还在排队的顾客队列

int first = 1;//是否是第一个输出的编号

while (!queue.IsEmpty()) {//将所有顾客分到A、B窗口

int num = queue.pop();//弹出队头编号并返回给num

if (num % 2 == 1) //编号为奇数进入A队列

A\_queue.push(num);

else//编号为偶数进入B队列

B\_queue.push(num);

}

int t = 0;//记录时间

while (!A\_queue.IsEmpty()) {

if (!first)//不是第一个被输出的顾客编号则在编号前输出空格

cout << " ";

cout << A\_queue.pop();//弹出出A队头并输出其编号

t++;//A窗口处理完一个顾客，时间+1

if (t == 2) {//A处理完两个顾客，B处理完一个顾客

if (!B\_queue.IsEmpty()) {

cout << " " << B\_queue.pop();//弹出B队头并输出其编号

}

t = 0;//一个周期结束时间，清零。

}

first = 0;//第一个输出标志清零

}

while (!B\_queue.IsEmpty()) {

if (!first)//不是第一个被输出的顾客编号则在编号前输出空格

cout << " ";

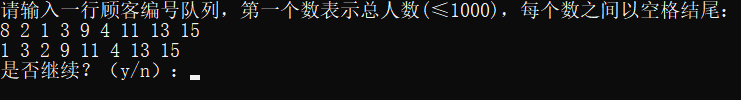
cout << B\_queue.pop();

first = 0;

}

}

### 银行业务功能截屏示例



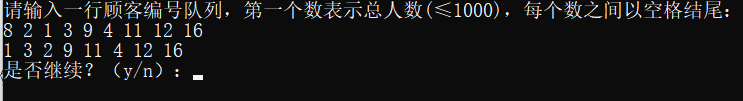
# 4 测试

## 4.1 功能测试

**测试用例**：8 2 1 3 9 4 11 12 16（正常测试，B窗口人多）

**预期结果**：1 3 2 9 11 4 12 16

**实验结果**



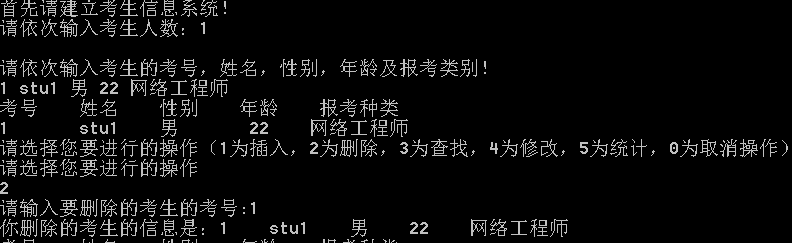
## 4.2 边界测试

### 4.2.1 初始化最小顾客数

**测试用例：**1 6

**预期结果：**6。

**实验结果：**



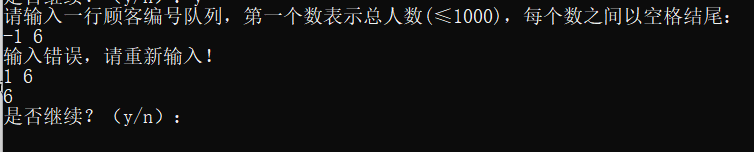
## 4.3 出错测试

### 4.3.1 顾客数输入错误

**测试用例：**-1 6

**预期结果：**程序给出错误提示，不崩溃。

**实验结果：**

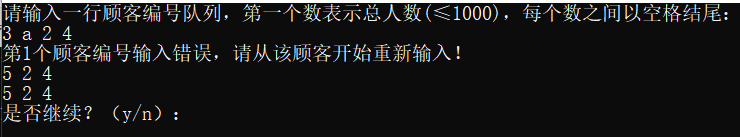


### 4.3.2 顾客编号输入错误

**测试用例：**3 a 2 5

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

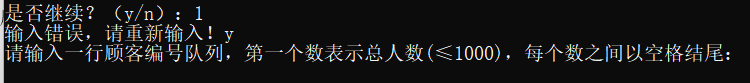
****

### 4.3.3 是否继续操作码输入错误

**测试用例：**输入y、n以外的字符

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

****