项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 祝新元

学 号： 1751629

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](#_Toc531889735)

[1.1 项目内容 1](#_Toc531889736)

[1.2项目功能要求 1](#_Toc531889737)

[2 设计 2](#_Toc531889738)

[2.1 数据结构设计 2](#_Toc531889739)

[2.2 类结构设计 2](#_Toc531889740)

[2.3 成员与操作设计 2](#_Toc531889741)

[2.4 系统设计 3](#_Toc531889742)

[3 实现 4](#_Toc531889743)

[3.1 将顾客分别加入两个队列的实现 4](#_Toc531889744)

[3.1.1核心代码 4](#_Toc531889745)

[3.2 有一个队列为空 4](#_Toc531889746)

[3.2.1核心代码 4](#_Toc531889747)

[3.3 办理业务 4](#_Toc531889748)

[3.3.1核心代码 4](#_Toc531889749)

[4 测试 5](#_Toc531889750)

[4.1 测试用例 5](#_Toc531889751)

[4.1.1正常测试，A窗口人多 5](#_Toc531889752)

[4.1.2 正常测试，B窗口人多 6](#_Toc531889753)

[4.1.3 最小N 7](#_Toc531889754)

[4.1.4 A窗口没有人 7](#_Toc531889755)

[4.1.5 B窗口没有人 8](#_Toc531889756)

# 1 分析

## 1.1 项目内容

设某银行有A，B两个业务窗口，且处理业务的速度不一样，其中A窗口处理速度是B窗口的2倍----即当A窗口每处理完2个顾客是，B窗口处理完1个顾客。给定到达银行的顾客序列，请按照业务完成的顺序输出顾客序列。假定不考虑顾客信后到达的时间间隔，并且当不同窗口同时处理完2个顾客时，A窗口的顾客优先输出。

## 1.2项目功能要求

输入说明：输入为一行正整数，其中第一数字N（N<=1000）为顾客总数，后面跟着N位顾客的编号。编号为奇数的顾客需要到A窗口办理业务，为偶数的顾客则去B窗口。数字间以空格分隔。

输出说明：按照业务处理完成的顺序输出顾客的编号。数字键以空格分隔，但是最后一个编号不能有多余的空格。

# 2 设计

## 2.1 数据结构设计

考虑到要实现银行的业务处理，其中涉及到来的早的客户先办理业务，两个窗口的处理业务的速度也不一样，所以我采用队列的数据结构。

## 2.2 类结构设计

经典的队列一般包括两个抽象数据类型（ADT）——结点类（LinkNode）与队列类（LinkedQueue），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。为方便处理，本系统采用struct描述结点类（LinkNode），这样使得结点类（LinkedQueue）可以访问队列。同时为体现面向对象编程的思想，在设计时所采用的数据结构均为模板类，template<class T>，之后不在重复。

## 2.3 成员与操作设计

**结点类（**LinkNode**）**

**公有成员：**

T data; //表示数据

LinkNode<T> \*link; //下一个LinkNode的指针域

**公有操作：**

LinkNode(LinkNode<T> \*next=NULL):link(next){}//构造函数

LinkNode(const T& d,LinkNode<T> \*next=NULL):data(d),link(next){} //具有默认参数的构造函数

**队列类（**LinkedQueue**）**

**私有成员：**

LinkNode<T> \*front;//链表的头指针

LinkNode<T> \*rear;//链表的尾指针

**公有操作：**

LinkedQueue<T>::LinkedQueue():rear(NULL),front(NULL){}

// LinkedQueue的构造函数，头指针和尾指针均指向NULL

LinkedQueue<T>::~LinkedQueue(){}

// LinkedQueue的析构函数，通过调用makeEmpty()实现对内存的回收

bool LinkedQueue<T>::EnQueue(const T& x)

//让数据x加入队列，若头指针为空，则新建一个结点；否则把他加入尾指针的后面

void LinkedQueue<T>::makeEmpty()

//收回链表所占的内存，防止内存泄漏

bool LinkedQueue<T>::DeQueue(T& x)

//使得队列第一个元素出队列，成功则返回true

bool LinkedQueue<T>::getFront(T& x)

//得到队列中第一个元素的值，返回在x中

int LinkedQueue<T>::getSize()const

//得到队列的大小

void LinkedQueue<T>::show()

//输出整个队列

## 2.4 系统设计

先读入顾客总数，新建队列q1和q2，读入所有的顾客编号，根据他们编号的奇偶将他们分别加入队列q1和q2.若是有一个队列是空，则直接输出另一个队列的所有顾客。若都不为空，根据q1处理两个顾客，q2处理一个顾客的速度依次输出所有顾客，若都要输出则先输出q1的。

# 3 实现

## 3.1 将顾客分别加入两个队列的实现

### 3.1.1核心代码

for(int i=0;i<N;i++)

{

cin>>cus;

//顾客的编号为偶数

if(cus%2==0)

{

flag=q2->EnQueue(cus);

}

//顾客的编号为奇数

else

{

q1->EnQueue(cus);

}

}

## 3.2 有一个队列为空

### 3.2.1核心代码

//q1为空

if(q1->getSize()==0)

{

q2->show();

return 0;

}

//q2为空

else if(q2->getSize()==0)

{

q1->show();

return 0;

}

## 3.3 办理业务

### 3.3.1核心代码

int count=1;

int deque=0;

int Maxcus = max(q1->getSize(), q2->getSize() \* 2);//最多办理业务时间

while(count<=Maxcus)

{

if (q1->DeQueue(deque))

{

cout << deque << ' ';

}

if(count%2==0)

{

if(q2->DeQueue(deque))

{

cout<<deque<<' ';

}

}

count++;

}

# 4 测试

## 4.1 测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 输入 | 输出 | 说明 |
| 1 | 8 2 1 3 9 4 11 13 15 | 1 3 2 9 11 4 13 15 | 正常测试，A窗口人多 |
| 2 | 8 2 1 3 9 4 11 12 16 | 1 3 2 9 11 4 12 16 | 正常测试，B窗口人多 |
| 3 | 1 6 | 6 | 最小N |

### 4.1.1正常测试，A窗口人多

**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**



### 4.1.2 正常测试，B窗口人多

**测试用例：**8 2 1 3 9 4 11 12 16

**预期结果：**

1 3 2 9 11 4 12 16

**实验结果：**



### 4.1.3 最小N

**测试用例：**1 6

**预期结果：**

6

**实验结果：**



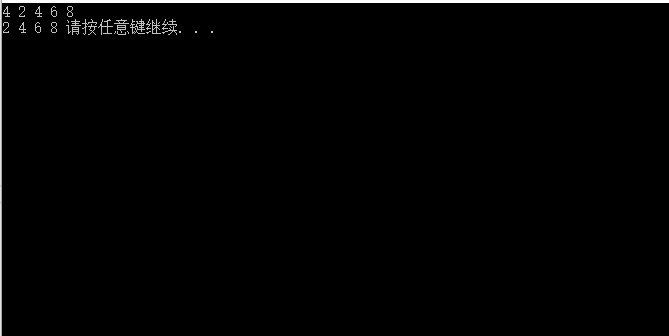
### 4.1.4 A窗口没有人

**测试用例：4 2 4 6 8**

**预期结果：**

2 4 6 8

**实验结果：**



### 4.1.5 B窗口没有人

**测试用例：4 1 3 5 7**

**预期结果：**

1 3 5 7

**实验结果：**

