项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 赵子昱

学 号： 1951459

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](#_Toc59454516)

[1.1 项目内容 1](#_Toc59454517)

[1.2 功能要求 1](#_Toc59454518)

[2 设计 1](#_Toc59454519)

[2.1 数据结构设计 1](#_Toc59454520)

[2.2 类结构设计 1](#_Toc59454521)

[2.3 成员与操作设计 2](#_Toc59454522)

[**3.Bank()** 函数的实现 2](#_Toc59454523)

[3.1 **Bank()** 函数的流程图 3](#_Toc59454524)

[3.2 **Bank()** 函数的代码 5](#_Toc59454525)

[4 测试 5](#_Toc59454526)

[4.1 功能测试 5](#_Toc59454527)

[4.1.1 正常测试，A窗口人多 5](#_Toc59454528)

[4.1.2 正常测试，B窗口人多 6](#_Toc59454529)

[4.1.3 最小N 6](#_Toc59454530)

[4.1.4 测试4 6](#_Toc59454531)

[4.1.5 测试5 7](#_Toc59454532)

[4.1.6 测试6 7](#_Toc59454533)

# 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。

## 功能要求

1. 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。
2. 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。
3. 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 |

# 2 设计

## 2.1 数据结构设计

银行窗口前，顾客需要排队，先来的先办业务并且先完成，基于这个特点，考虑用队列来实现两个窗口顾客信息的保存。为了节约存储空间，选择基于数组的循环队列。

## 2.2 类结构设计

队列中的函数包括基本的入队、出队、判空、判满函数。

## 2.3 成员与操作设计

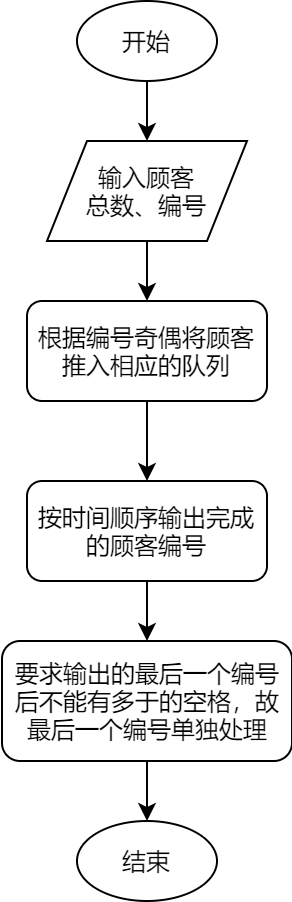
1. **template** <**class** T>
2. **class** Queue {
3. **public**:
4. Queue() :front(0),rear(0), maxSize(0) {}
5. Queue(**int** size) :elements(**new** T[size]), front(0), rear(0), maxSize(size) {}
6. ~Queue() { **delete**[]elements; };
8. **void** Push(T &newT);
9. //入队
11. T Pop();
12. //出队
14. **int** IsFull() { **return** (rear + 1) % maxSize == front; }
15. //判断队列是否满
17. **int** IsEmpty(){ **return** rear == front; }
18. //判断队列是否空
20. **int** getSize() { **return** maxSize; }
21. //返回最多元素个数
23. **private**:
24. T \*elements;
25. //队列的元素
27. **int** rear;
28. //队尾
30. **int** front;
31. //队头
33. **int** maxSize;
34. //最多元素个数
35. };

# **3.Bank()** 函数的实现

**Bank()** 函数为整个项目的主要函数，能实现顾客信息的输入，将用户分配到相应的队列，输出完成顺序编号。

主函数中直接调用**Bank()**。

## 3.1 **Bank()** 函数的流程图



## 3.2 **Bank()** 函数的代码

1. **void** Bank() {
2. **int** Num, count, countA = 0, countB;
3. //输入的顾客的编号，顾客总数，A、B窗口的顾客数
4. cin >> count;
5. Queue<**int**> QueueA(count + 1), QueueB(count + 1);  //创建A、B窗口的队列
6. //输入编号并将用户编号入相应的队列
7. **for** (**int** i = 0; i < count; i++) {
8. cin >> Num;
9. **if** (Num % 2 == 1) {
10. QueueA.Push(Num);
11. countA++;
12. }
13. **else** { QueueB.Push(Num); }
14. }
15. countB = count - countA;
16. //按时间顺序输出完成的顾客编号（A窗口处理完一个顾客的时间为时间单位）
17. **int** Time = countA < 2 \* countB ? 2 \* countB - 1 : countA - 1;
18. **for** (**int** i = 0; i < Time; i++) {
19. **if** (i % 2 == 0) {
20. **if** (countA-- > 0) { cout << QueueA.Pop() << " "; }
21. }
22. **else** {
23. **if** (countA-- > 0) { cout << QueueA.Pop() << " "; }
24. **if** (countB-- > 0) { cout << QueueB.Pop() << " "; }
25. }
26. }
27. //要求输出的最后一个编号后不能有多于的空格，故最后一个编号单独处理
28. **if** (countA >0 && countB >0) {
29. cout << QueueA.Pop() << " " << QueueB.Pop();
30. }
31. **else** {
32. **if** (countA > 0) { cout << QueueA.Pop(); }
33. **if** (countB > 0) { cout << QueueB.Pop(); }
34. }
35. }

# 4 功能测试

## 4.1 正常测试，A窗口人多

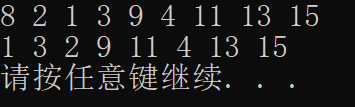
**测试用例**：

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**



## 4.2 正常测试，B窗口人多

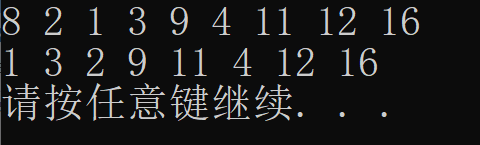
**测试用例：**

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：**



## 4.3 最小N

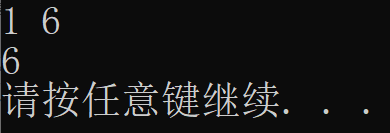
**测试用例：**

1 6

**预期结果：**

6

**实验结果：**



## 4.4 测试4

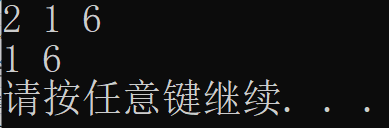
**测试用例：**

2 1 6

**预期结果：**

1 6

**实验结果：**



## 4.5 测试5

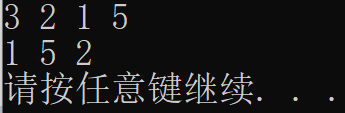
**测试用例：**

3 2 1 5

**预期结果：**

1 5 2

**实验结果：**



## 4.6 测试6

**测试用例：**

4 2 1 4 5

**预期结果：**

1 5 2 4

**实验结果：**

