项目说明文档

数据结构课程设计

——银行业务办理

作 者 姓 名： 杨晶

学 号： 1854025

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](#_Toc495668153)

[1.1 背景分析](#_Toc495668154)

[1.2 功能分析](#_Toc495668155)

[2 设计](#_Toc495668156)

[2.1 数据结构设计](#_Toc495668157)

[2.2 类结构设计](#_Toc495668158)

[2.3 成员与操作设计](#_Toc495668159)

[2.4 系统设计](#_Toc495668160)

[3 实现](#_Toc495668161)

[3.1 搜索功能的实现](#_Toc495668162)

[3.1.1 搜索功能流程图](#_Toc495668163)

[3.1.2 搜索功能核心代码](#_Toc495668164)

[3.2 打印功能的实现](#_Toc495668166)

[3.2.1 打印功能流程图](#_Toc495668167)

[3.2.2 打印功能核心代码](#_Toc495668168)

[3.3 输入功能的实现](#_Toc495668170)

[3.3.1 输入功能流程图](#_Toc495668171)

[3.3.2 输入功能核心代码](#_Toc495668172)

[3.3.3 输入功能截图示例](#_Toc495668173)

[4 测试](#_Toc495668186)

[4.1 功能测试](#_Toc495668187)

[4.2 边界测试](#_Toc495668193)

[4.2.1 某个链表无输入数据](#_Toc495668194)

[4.2.2 交集数据在头/尾节点的情况](#_Toc495668195)

[4.2.3 交集为空](#_Toc495668196)

[4.3 出错测试](#_Toc495668197)

[4.3.1 异常输入](#_Toc495668198)

[4.3.2 不满足非降序正整数序列](#_Toc495668199)

# 1 分析

## 1.1 背景分析

众所周知，一个银行拥有优秀的银行业务办理是十分重要的，要想打造一支优质文明的优质专业人才服务队伍，除了要求员工具备扎实的专业功底、娴熟的职业技能、高度的责任意识、自始至终准确地提供服务等这些员工基本素质条件之外，还要强调一种“以客户为中心”的服务理念、服务意识。

今天的银行服务已经不再简单的归结为“办理某某业务”这么简单，银行业原本就是一项高负债的行业，现如今这种负债更增加了“服务资本”。服务的不慎足以导致银行经营满盘皆输，银行如何全方位服务使自己成为客户的优质财务管家，是银行业发展任重而道远的目标；是值得我们每一位银行从业人员努力的方向。对于银行已知客流量赫尔处理速度的业务窗口，倘若我们能够预知顾客的出入顺序，那么想必对于一个银行打造昂扬向上的服务文化，塑造良好的窗口形象是十分必要的。

因此，为了促进企业的发展，对于两个已知客户处理速度的窗口，我们可以试着写出一个小程序预判出客人流出的顺序。

## 1.2 功能分析

已知银行的A，B两个业务窗口处理业务的速度不一样，A窗口处理速度是B窗口的2倍。给定到达银行的顾客序列，我们的程序可以输出顾客序列。假定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。

该程序输入为一行正整数，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。

程序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

如上功能分析所述，改程序需要先进入的顾客先出来。综合考虑，队列这种数据结构适合进行这种操作。队列的数据元素又称为队列元素。在队列中插入一个队列元素称为入队，从队列中删除一个队列元素称为出队。因为队列只允许在一端插入，在另一端删除，所以只有最早进入队列的元素才能最先从队列中删除，故队列又称为先进先出（FIFO—first in first out）[线性表](https://baike.baidu.com/item/%E7%BA%BF%E6%80%A7%E8%A1%A8)。

## 2.2 类结构设计

建立顺序队列结构必须为其静态分配或动态申请一片连续的存储空间，并设置两个指针进行管理。一个是队头指针front，它指向队头元素；另一个是队尾指针rear，它指向下一个入队元素的存储位置。

每次在队尾插入一个元素是，rear增1；每次在队头删除一个元素时，front增1。随着插入和删除操作的进行，队列元素的个数不断变化，队列所占的存储空间也在为队列结构所分配的连续空间中移动。当front=rear时，队列中没有任何元素，称为空队列。当rear增加到指向分配的连续空间之外时，队列无法再插入新元素，但这时往往还有大量可用空间未被占用，这些空间是已经出队的队列元素曾经占用过得存储单元。

## 2.3 成员与操作设计

1. **class** Queue {
2. **public**:
3. **void** push(**const** **int**& x);//入队
4. **bool** pop();//出队
5. **int** getTop() { **return** people[top]; };//返回队首元素
6. **bool** isEmpty() **const** { **return** top == rear; }
7. **bool** isFulll() **const** { **return** rear == maxSize ; }
8. Queue() { top = rear = 0; };
9. Queue(**int** size);
10. ~Queue(){**delete**[]people;}
12. **private**:
13. **int**\* people;//存储数组
14. **int** top;//头结点
15. **int** rear;//尾结点
16. **int** maxSize;//最大容量
17. };
19. //含参构造函数
20. Queue::Queue(**int** size): top(0),rear(0),maxSize(size){
21. people = **new** **int**[maxSize];
22. assert(people != nullptr);
23. }
25. //入队
26. **void** Queue::push(**const** **int**& x){
27. assert(!isFulll());
28. people[rear] = x;
29. rear++;
30. }
32. //出队
33. **bool** Queue::pop(){
34. **if** (isEmpty() == **true**) **return** **false**;
35. top++;
36. **return** **true**;
37. }

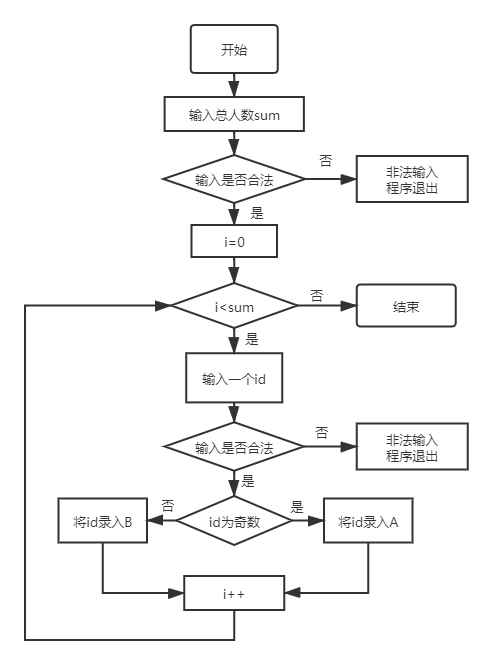
## 2.4 系统设计

在主函数中，用户首先输入顾客队列信息，再通过我们设计的函数将不同的信息写入两个队列中。之后我们通过循环结构的代码对输出顺序进行控制，对信息进行处理，最终输出结果。

# 3 实现

## 3.1数据录入功能的实现

### 3.1.1 数据录入功能流程图

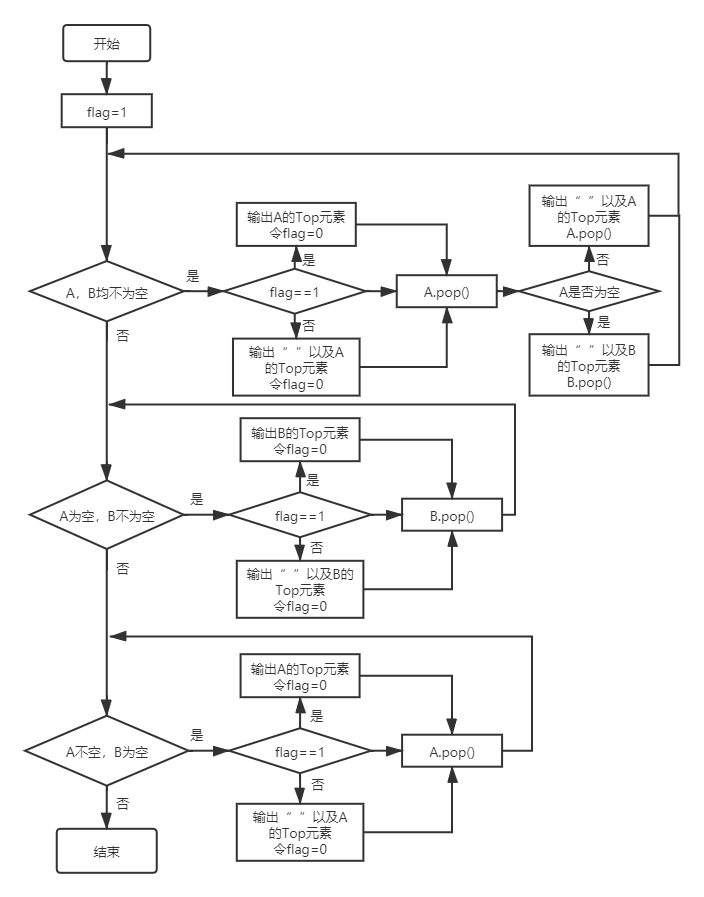


### 3.1.2 数据录入核心代码

1. **int** sum;
2. **int** id;
3. cin >> sum;
5. **if** (cin.fail() || sum <= 0)
6. cout << "非法输入！" << endl << "程序强制退出！" << endl;
8. Queue A(sum);
9. Queue B(sum);
11. **for** (**int** i = 0; i < sum; i++) {
12. cin >> id;
13. **if** (cin.fail()||id<=0) {
14. cout << "非法输入！" << endl << "程序强制退出！" << endl;
15. system("pause");
16. exit(0);
17. }
18. **if** (id % 2 == 1) {
19. A.push(id);
20. }
21. **else** **if** (id % 2 == 0) {
22. B.push(id);
23. }
24. }

## 3.2 输出功能的实现

### 3.2.1 输出功能流程图



### 3.2.2 输出功能核心代码

1. **int** flag = 1;//处理空格：判断当前是不是输出的第一个编号
2. //A，B都不为空
3. **while** (A.isEmpty() != 1 && B.isEmpty()!= 1 ) {
4. //输出一个数，先看A
5. **if** (flag == 1) {//如果是第一次输出就不输出空格
6. cout << A.getTop();
7. flag--;
8. }
9. **else** **if** (flag == 0) {//不是第一次输出就输出空格
10. cout << " " << A.getTop();
11. }
12. A.pop();
13. //如果A还有就输出A
14. **if** (A.isEmpty() != 1) {
15. cout << " " << A.getTop();
16. A.pop();
17. }
18. //输出B
19. cout << " " << B.getTop();
20. B.pop();
21. }
23. //A为空，B不为空
24. **while** (A.isEmpty() == 1 && B.isEmpty() != 1) {
25. **if** (flag == 1) {//如果是第一次输出就不输出空格
26. cout << B.getTop();
27. flag--;
28. }
29. **else** **if** (flag == 0) {//不是第一次输出就输出空格
30. cout << " " << B.getTop();
31. }
32. B.pop();
33. }
35. //B为空，A不为空
36. **while** (A.isEmpty() != 1 && B.isEmpty() == 1) {
37. **if** (flag == 1) {//如果是第一次输出就不输出空格
38. cout << A.getTop();
39. flag--;
40. }
41. **else** **if** (flag == 0) {//不是第一次输出就输出空格
42. cout << " " << A.getTop();
43. }
44. A.pop();
45. }

# 4 测试

## 4.1 功能测试

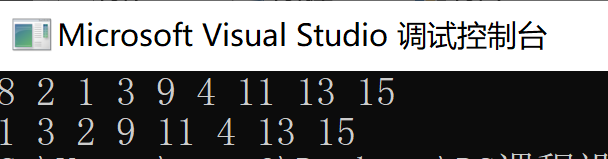
**测试用例**：

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**



## 4.2 边界测试

### 4.2.1 最小n

**测试用例：**1 6

**预期结果：6**。

**实验结果：**



## 4.3 出错测试

### 4.3.1 异常输入

**测试用例：**输入非数字符号，如；。？

**预期结果：**程序报错，启动退出。

**实验结果：**

****

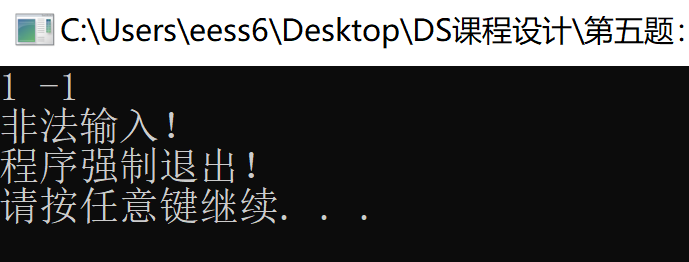
### 4.3.2 不满足输入的是一行正整数

**测试用例：**

1 -1

**预期结果：**程序报错，启动退出。

**实验结果：**

****