项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 曹峰源

学 号： 1951328

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 3](#_Toc58873480)

[1.1 背景分析 3](#_Toc58873481)

[1.2 功能分析 3](#_Toc58873482)

[2. 设计 3](#_Toc58873483)

[2.1 数据结构设计 3](#_Toc58873484)

[2.2 类结构设计 3](#_Toc58873485)

[2.3 成员内容 3](#_Toc58873486)

[2.3.1 顾客结点类 3](#_Toc58873487)

[2.3.2 窗口队列类 3](#_Toc58873488)

[3. 功能 4](#_Toc58873489)

[3.1 流程图 4](#_Toc58873490)

[3.2 代码实现 4](#_Toc58873491)

[3.3 结果显示 6](#_Toc58873492)

[4. 测试 6](#_Toc58873493)

[4.1 总人数错误测试 6](#_Toc58873494)

[4.2 边界情况测试 6](#_Toc58873495)

1. 分析
   1. 背景分析

对于当今社会，各行各业都越来越离不开计算机领域，对于银行窗口业务的处理也不例外，同样可以使用数据结构的知识来帮助简化数据存储计算，这段程序便是依据银行业务处理的简单 。

* 1. 功能分析

对于银行业务处理的问题，这段程序应能够满足客户编号数据的输入，并对于这些数据进行奇数进入A窗口、偶数进入B窗口的操作，最后应当能实现根据A窗口与B窗口的业务处理速度输出客户处理顺序。

1. 设计
   1. 数据结构设计

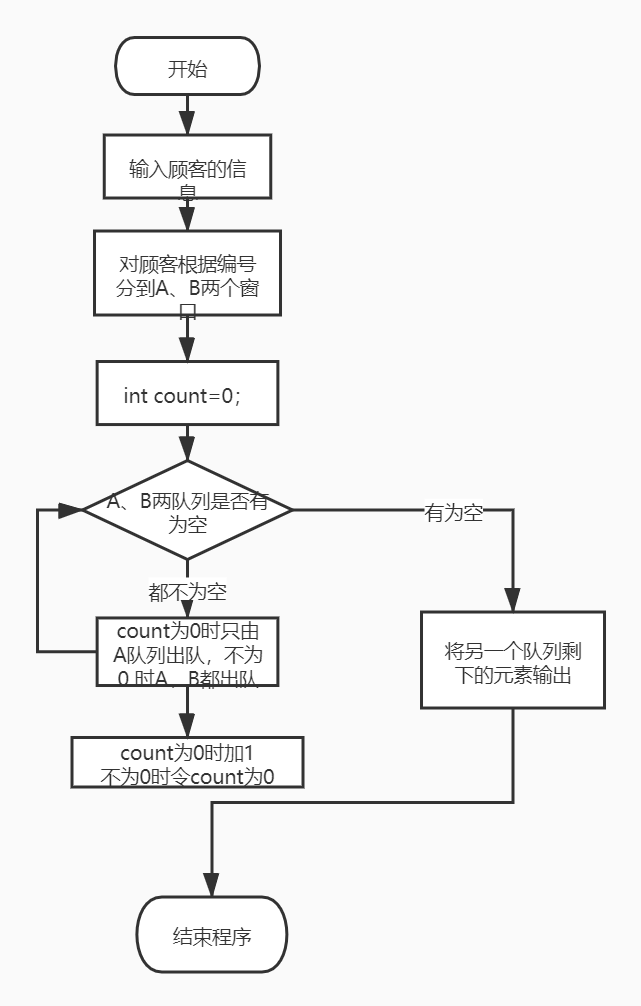
因为银行应当是先处理完先到的顾客，所以应当是先来先出的，这样便选择使用队列这样的数据结构来实现功能。

* 1. 类结构设计

这段程序选择的是使用链表来实现队列，并创建一个结点类来存放顾客的编号数据，并让链表队列类成为结点类的友元类。

* 1. 成员内容
     1. 顾客结点类

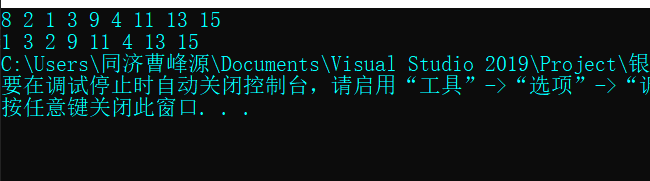
1. **class** node//创建用于存放数据的结点类
2. {
3. **friend** **class** queue;
4. **private**:
5. **int** data;
6. node\* next=NULL;
7. };
   * 1. 窗口队列类
8. **class** queue//创建队列类
9. {
10. **private**:
11. node\* front;
12. node\* rear;
13. **public**:
14. queue()//构造函数
15. {
16. front = **new** node;
17. rear = front;
18. }
19. ~queue() //析构函数
20. {
21. **while** (front!=rear)//将队列中当前每一个结点都释放掉
22. {
23. **int** x = 0;
24. dequeue(x);
25. }
26. }
27. **bool** isempty();//判断队列是否为空
28. **void** inqueue(**int** p);//进队列的函数
29. **bool** dequeue(**int** &x);//出队列的函数
30. };
31. 功能
    1. 流程图



* 1. 代码实现

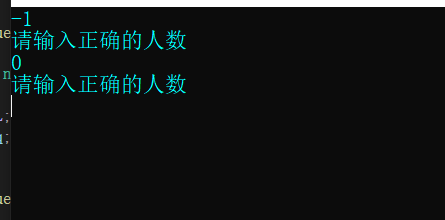
1. **int** main()
2. {
3. queue a, b;//建立两个队列
4. **int** N;
5. **while** (cin >> N)//输入总顾客的人数
6. {
7. **if** (N > 0) **break**;
8. **else**
9. {
10. cout << "请输入正确的人数"<<endl;
11. }
12. }
13. **int** num=0,data;
14. **while** (num<N)//输入每个顾客的编号，编号为奇数的进入a队列，为偶数的进入b队列
15. {
16. cin >> data;
17. **if** (data % 2 == 0)
18. {
19. b.inqueue(data);
20. }
21. **else** a.inqueue(data);
22. num++;
23. }
24. **int** count = 0;//记录已经输出的次数
25. **while** (a.isempty()!=0&&b.isempty()!=0)//a队列与b队列都不为0时
26. {
27. **if** (count==0)//count为0说明b队列还在处理业务，将a队列的输出
28. {
29. **int** x=0;
30. a.dequeue(x);
31. cout << x << ' ';
32. count++;
33. }
34. **else**//不为0说明a与b队列都处理完当前业务了，都进行输出
35. {
36. **int** x = 0;
37. a.dequeue(x);
38. cout << x << ' ';
39. b.dequeue(x);
40. cout << x << ' ';
41. count = 0;
42. }
43. }
44. **while** (a.isempty() == 0 && b.isempty() != 0)//a队列为空后，将b队列中剩余元素输出
45. {
46. **int** x = 0;
47. b.dequeue(x);
48. cout << x << ' ';
49. }
50. **while** (a.isempty() != 0 && b.isempty() == 0)//b队列为空后，将a队列中剩余元素输出
51. {
52. **int** x = 0;
53. a.dequeue(x);
54. cout << x << ' ';
55. }
56. **return** 0;
57. }
    1. 结果显示

以2 1 3 9 4 11 13 15为例：



1. 测试
   1. 总人数错误测试

期望：总人数输入小于等于0的时候程序不崩溃



* 1. 边界情况测试

以只输入6为例

期望：程序正常运行，只输出6

