项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 伊啸

学 号： 1951220

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](#_Toc60086807)

[1.1 项目简介 1](#_Toc60086808)

[1.2 功能分析 1](#_Toc60086809)

[2 设计 1](#_Toc60086810)

[2.1 数据结构设计 1](#_Toc60086811)

[2.2 类结构设计 1](#_Toc60086812)

[2.3 成员与操作设计 1](#_Toc60086813)

[2.4 系统设计 3](#_Toc60086815)

[3 实现 4](#_Toc60086816)

[3.1 插入功能的实现 4](#_Toc60086817)

[3.1.1 插入功能流程图 4](#_Toc60086818)

[3.1.2 插入功能核心代码 6](#_Toc60086819)

[3.2 银行业务功能的实现 7](#_Toc60086820)

[3.2.1 银行业务功能流程图 7](#_Toc60086821)

[3.2.2 银行业务功能核心代码 9](#_Toc60086822)

[3.2.3 银行业务功能截屏示例 10](#_Toc60086823)

[3.3 总体系统的实现 11](#_Toc60086824)

[3.3.1 总体系统流程图 11](#_Toc60086825)

[3.3.2 总体系统核心代码 12](#_Toc60086826)

[3.3.3 总体系统截屏示例 13](#_Toc60086828)

[4 测试 13](#_Toc60086829)

[4.1 功能测试 13](#_Toc60086830)

[4.2 出错测试 14](#_Toc60086831)

[4.2.1 输入客户人数不是正整数 14](#_Toc60086832)

[4.2.2 输入顾客编号错误 14](#_Toc60086833)

# 1 分析

## 1.1 项目简介

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 功能分析

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。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，采用链表来存储数据，便于插入删除等操作。

## 2.2 类结构设计

由于是先到窗口的顾客先进行输出，自然想到用队列来实现项目功能。

## 2.3 成员与操作设计

**队列（class queue）**

**公有成员：**

int rear, front;// 队列头和尾指针

int\* element;// 队列数组

int maxsize;// 队列可容纳的最大元素数

**公有操作：**

queue(int sz);//构造函数

~queue() {};//析构函数

bool Enqueue(const int& x);

//如果队列未满，则入队x并返回true，否则返回false

bool Dequeue(int& x);

// 如果队列不为空，则使head元素出队并返回true，否则返回false

bool getFront(int& x);// 通过x返回队列的head元素

void makeEmpty() { front = rear = 0; }//使队列为空

bool IsEmpty()const { return (rear == front) ? true : false; }

//确定队列是否为空

bool IsFull()const { return rear == maxsize ? true : false; }

//确定队列是否已满

int getSize()const { return rear - front; }//返回队列的大小

**链表结点类（node）**

**公有成员：**

int data;//结点中的数据

node\* link;//指向下一个结点的指针

**公有操作：**

node():data(0),link(NULL){}//构造函数

node(int number):data(number),link(NULL){}//构造函数

**链表类（LinkList）**

**私有成员：**

node\* head;//头结点

**公有操作：**

linklist();//构造函数

~linklist() {};//析构函数

void Insert(int position, int number);// 在该位置插入number

void Bank\_business(int );// 实现银行业务功能

## void display();

## 2.4 系统设计

系统首先实现对屏幕的初始化，完成对链表的创建和输入数据工作，然后根据数据为奇数还是偶数进入A、B队列，实现银行业务功能

# 3 实现

## 3.1 插入功能的实现

### 3.1.1 插入功能流程图



### 3.1.2 插入功能核心代码

void linklist::Insert(int position, int number) {

node\* p = head; //head为头结点，不存储输入数据

for (int i = 0; i < position; i++)//使p指向插入位置的前一个节点

p = p->link;

node\* newnode = new node(number);

newnode->link = p->link;

p->link = newnode;

}

## 3.2 银行业务功能的实现

### 3.2.1 银行业务功能流程图



### 3.2.2 银行业务功能核心代码

node\* p = head->link;

node\* q = list->head;

while (p) {

if (p->data % 2 == 1)

bank\_queue\_A.Enqueue(p->data);//奇数进队列A

else

bank\_queue\_B.Enqueue(p->data);//偶数进队列B

p = p->link;

}

for (int i = 0; i < numbers; i++) {

int Afirst, Asecond, Bfirst;

if (!bank\_queue\_A.IsEmpty()) {

//A窗口优先输出且A窗口输出两个顾客时B窗口输出一个顾客 bank\_queue\_A.Dequeue(Afirst);

node\* newnode = new node(Afirst);

newnode->link = q->link; //数据插入新链表

q->link = newnode;

q = q->link;

}

if (!bank\_queue\_A.IsEmpty()) {

bank\_queue\_A.Dequeue(Asecond);

node\* newnode = new node(Asecond);

newnode->link = q->link;

q->link = newnode;

q = q->link;

}

if (!bank\_queue\_B.IsEmpty()) {

bank\_queue\_B.Dequeue(Bfirst);

node\* newnode = new node(Bfirst);

newnode->link = q->link;

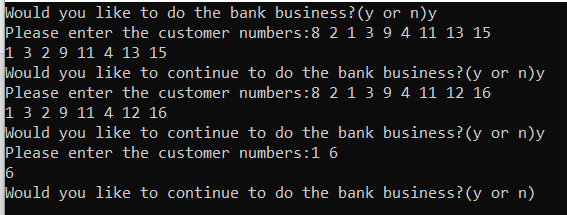
q->link = newnode;

q = q->link;

}

}

### 3.2.3 银行业务功能截屏示例



## 3.3 总体系统的实现

### 3.3.1 总体系统流程图



### 3.3.2 总体系统核心代码

linklist cust\_link;

int target; //标记是否存在负数

for (int i = 0; i < cust\_numbers; i++) {

target = 0;

int number = 0;

cin >> number;

if (cin.good() == false|| number <= 0) {

cout << "The number should be a positive integer, please enter valid data!" << endl;

target = 1;

break;

}

else

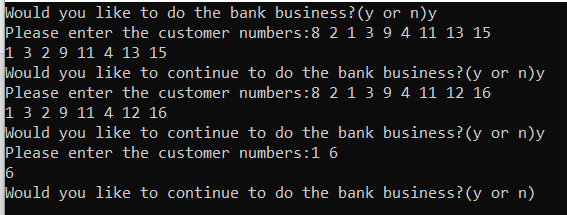
cust\_link.Insert(i, number);//将数据插入链表

}

if (target == 0)

### cust\_link.Bank\_business(cust\_numbers);//进行银行业务功能

### 3.3.3 总体系统截屏示例



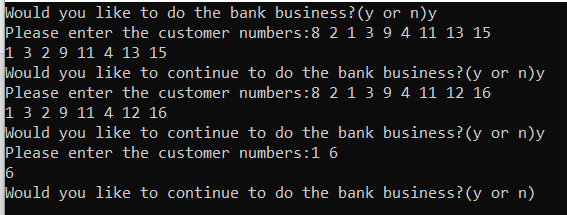
# 4 测试

## 4.1 功能测试

**测试用例： 预期结果：**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

**实验结果：**



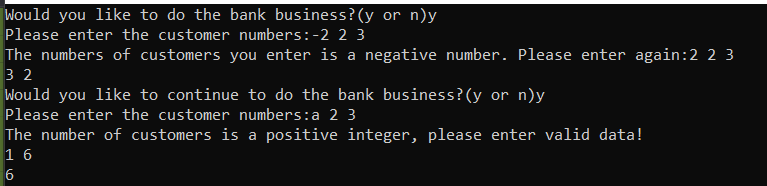
## 4.2 出错测试

### 4.2.1 输入客户人数不是正整数

**测试用例：**输入客户人数为负数或其他字符如abc，%&等

**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：**



**C:\Users\yixiao\AppData\Roaming\Tencent\Users\846507684\QQ\WinTemp\RichOle\I$O_M]A32]XL[%M20%1YHZH.png**

### 4.2.2 输入顾客编号错误

**测试用例：**输入客户编号为负数或其他字符如abc，%&等

**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：**

