项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 谢宇翔

学 号： 1951708

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](#_Toc28797)

[1.1 背景分析 1](#_Toc23744)

[1.2 功能分析 1](#_Toc21998)

[2 设计 1](#_Toc31602)

[2.1 数据结构设计 1](#_Toc14060)

[2.2 类结构设计 1](#_Toc8926)

[2.3 成员与操作设计 1](#_Toc23707)

[2.4 系统设计 2](#_Toc27798)

[3 实现 3](#_Toc9025)

[3.1 读取顾客序号的实现 3](#_Toc12510)

[3.1.1 功能流程图 3](#_Toc727)

[3.1.2 核心代码 3](#_Toc11631)

[3.2 输出排列好的顾客序列 8](#_Toc22864)

[3.2.1 功能流程图 8](#_Toc27420)

[3.2.2 核心代码 9](#_Toc27679)

[3.2.3 输出功能截屏示例 10](#_Toc8658)

[4 测试 11](#_Toc690)

[4.1 边界测试 11](#_Toc5528)

[4.1.1 只有一个用户 11](#_Toc18943)

[4.3 出错测试 11](#_Toc31856)

[4.3.1 客户人数输入错误 11](#_Toc29022)

[4.3.2 客户序号错误 12](#_Toc15426)

[4.3.3 客户数量大于1000 12](#_Toc13118)

[4.3.5 查找考号不存在 12](#_Toc2371)

[4.3.6 修改考号不存在 12](#_Toc20601)

# 1 分析

## 1.1 背景分析

对于银行每日接待的那么多客户，每个窗口的对不同业务的处理时间也不同，设计一个程序对不同业务的人们实行有效的窗口指引是个切实有用的方法，能大大节约时间充分利用时间，对人流的疏导也有很大作用。

## 1.2 功能分析

银行有2个窗口，2个窗口的业务处理速度不同，对于许多办理不同业务的人，可以做到按照业务完成的顺序输出顾客序列，达到充分利用时间的结果。

# 2 设计

## 2.1 数据结构设计

本程序主要用了2个链表类，分别存储2个银行窗口的顾客序列，再按照顺序依次输出2个链表里的数据，直到其中一个链表空为止，将另一个链表的所有顾客序号一起输出。每个顾客可以当做一个节点。

## 2.2 类结构设计

经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（node）与链表类（list），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描述链表结点类（node），这样使得链表结点类（list）可以访问链表结点。

## 2.3 成员与操作设计

**链表结点类（node）**

struct node

{

int data = 0;

node\* next=NULL;

};

**链表类（list）**

class list

{

public:

node\* head = NULL;//表头空节点

node\*current = NULL;//当前节点

};

## 2.4 系统设计

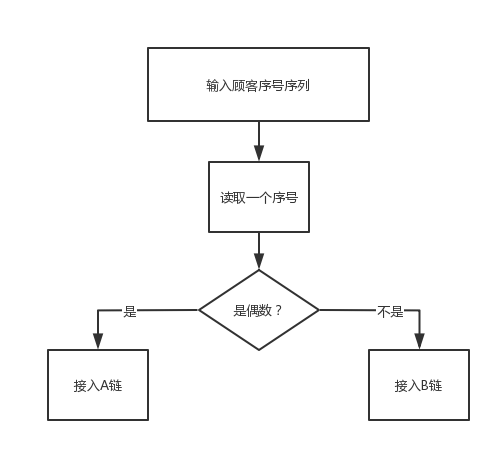
首先输入一段客户序号，然后对客户的序号逐个判断是否为偶数，分往2个链表。

当全部数据存储进2个链表后，再对2个链表按顺序输出，A链输出2个，B链输出一个，直到所有顾客都输出完毕。

# 3 实现

## 3.1 读取顾客序号的实现

### 3.1.1 功能流程图



### 3.1.2 核心代码

list L1, L2;//建立2个链表

int num;

bool isfirst1 = 1, isfirst2 = 1;

cout << "请先输入客户数量:(小于1000)" << endl;

while (1)

{

cin >> num;

if (cin.fail())

{

cout << "请输入正确数值!请重新输入客户数量:" << endl;

cin.clear();

cin.ignore(1000, '\n');

}

else if (num <= 0)

{

cout << "请输入正整数！请重新输入客户数量:" << endl;

cin.ignore(1000, '\n');

}

else if (num > 1000)

{

cout << "请输入小于1000的正整数！请重新输入客户数量:" << endl;

cin.ignore(1000, '\n');

}

else

break;

}

cout << "请输入所有客户的编号:" << endl;

int i = 0;

for (; i < num; i++)

{

int no;

while (1)

{

cin >> no;

if (cin.fail())

{

cout << "请输入正确数值!请从错误处开始继续输入客户序号:" << endl;

cin.clear();

cin.ignore(1);

}

else if (no <= 0)

{

cout << "请输入正整数！请从错误处开始继续输入客户序号:" << endl;

cin.ignore(1);

}

else

break;

}

if (no % 2 == 0)//偶数

{

if (isfirst2 == 1)//建立头结点

{

L2.head = new node;

L2.current = L2.head;

L2.head->data = no;

isfirst2 = 0;

}

else

{

L2.current->next = new node;

L2.current = L2.current->next;

L2.current->data = no;

}

}

else//奇数

{

if (isfirst1 == 1)//建立头结点

{

L1.head = new node;

L1.current = L1.head;

L1.head->data = no;

isfirst1 = 0;

}

else

{

L1.current->next = new node;

L1.current = L1.current->next;

L1.current->data = no;

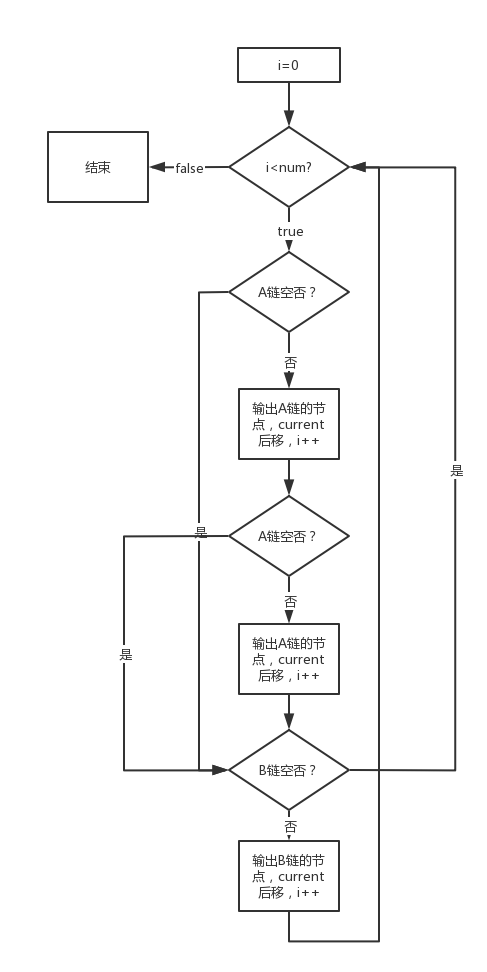
}

}

}

## 3.2 输出排列好的顾客序列

### 3.2.1 功能流程图



### 3.2.2 核心代码

i = 0;

L1.current = L1.head;

L2.current = L2.head;

for (; i < num; )

{

if (L1.current!= NULL)

{

if (i == 0)//输出头结点，不带空格

{

cout << L1.current->data;

L1.current = L1.current->next;

i++;

}

else//带空格

{

cout << " " << L1.current->data;

L1.current = L1.current->next;

i++;

}

}

if (L1.current != NULL)

{

cout << " " << L1.current->data;

L1.current = L1.current->next;

i++;

}

if (L2.current != NULL)

{

if (i == 0)

{

cout << L2.current->data;

L2.current = L2.current->next;

i++;

}

else

{

cout << " " << L2.current->data;

L2.current = L2.current->next;

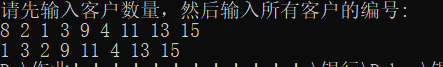
i++;

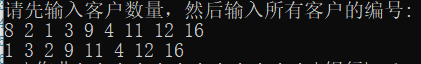
}

}

}

### 3.2.3 输出功能截屏示例





# 4 测试

## 4.1 边界测试

### 4.1.1 只有一个用户

**测试用例：**1 6

**预期结果：输出6**

**实验结果：**

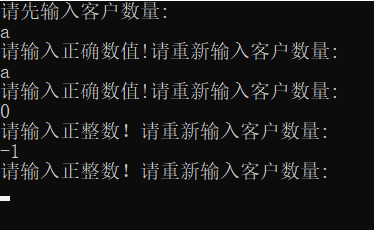
## 4.3 出错测试

### 4.3.1 客户人数输入错误

**测试用例：**输入客户人数错误

**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：**

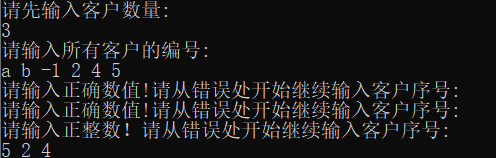


### 4.3.2 客户序号错误

**测试用例：**输入客户序号错误

**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：**



### 4.3.3 客户数量大于1000

**测试用例：**10000

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

### 4.3.5 查找考号不存在

**测试用例：**要查找的考号不存在

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

### 4.3.6 修改考号不存在

**测试用例：**要修改的考号不存在

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

