**项目说明文档**

**数据结构课程设计**

**——银行业务**

### 作 者 姓 名 尹诚成

学 号 2351279

指 导 教 师 张 颖

学 院 专 业 计算机科学与技术学院 软件工程



二〇二三 年 十二 月 十三 日

**目录**

[1.项目分析 1](#_bookmark0)

* 1. .[项目背景分析 1](#_bookmark1)
  2. .[项目需求分析 1](#_bookmark2)
  3. .[项目功能分析 1](#_bookmark3)

1.3.1.构建表达式二叉树功能 1

1.3.2.表达式转换与计算功能 1

1.3.3.异常处理功能 2

[2.项目设计 2](#_bookmark12)

* 1. .[数据结构设计 2](#_bookmark13)
  2. .[结构体与类设计 2](#_bookmark14)

2.2.1.Expression 联合体的设计 2

2.2.2 LinkNode 结构体的设计 3

2.2.3 Stack 类的设计 3

2.2.4 LinkedQueue类的设计 3

2.2.5 BinTreeNode 结构体的设计 3

2.2.6 BinaryTree 类的设计 3

2.2.7 ExpressionTree类的设计 3

* 1. .[项目主体架构设计 4](#_bookmark32)

[3.项目功能实现 4](#_bookmark33)

3.1.项目主体架构实现 4

3.2.构建表达式二叉树功能的实现 5

3.3.表达式计算功能的实现 5

3.4.表达式输入非法的处理功能的实现 5

4.项目测试 7

4.1.表达式输入合法性验证功能测试

4.2.构建表达式二叉树和表达式计算功能测试

1. **项目分析**
   1. 项目背景分析

在银行日常运营中，顾客到达并排队等待办理业务是一个常见的场景。为了提高服务效率，银行通常会设置多个业务窗口，并且这些窗口的处理速度可能因各种因素（如工作人员的熟练程度、窗口的配置等）而有所不同。

本项目旨在模拟一个具有两个不同处理速度业务窗口（A窗口和B窗口）的银行服务场景。其中，A窗口的处理速度是B窗口的两倍，即当A窗口处理完两个顾客时，B窗口只能处理完一个顾客。项目目标是给定一个到达银行的顾客序列，根据每个窗口的处理速度以及顾客的选择（奇数编号顾客去A窗口，偶数编号顾客去B窗口），计算出顾客业务完成的顺序，并按此顺序输出顾客的编号。

* 1. 项目需求分析

基于以上背景分析，本项目需要实现需求如下：

(1)用户应能通过命令行界面输入顾客的总数和每位顾客的标号；

(2)设计简单直观的控制台界面。

(3)根据A、B两个窗口的不同处理速度（A窗口是B窗口的2倍），模拟顾客在窗口办理业务的过程。

(4)根据顾客的编号（奇数去A窗口，偶数去B窗口）进行分类，并分别进入对应的窗口服务队列。

(5)根据窗口处理能力和顾客需求，智能调度顾客办理业务，并按照业务完成的顺序输出顾客序列。当不同窗口同时处理完顾客时，A窗口的顾客优先输出。

(6)实现异常处理机制，确保系统稳定性和安全性，避免因用户输入错误导致系统崩溃或信息丢失。

* 1. 项目功能分析

本项目旨在通过模拟计算机处理和计算表达式的过程，实现表达式二叉树的构建，并通过前序遍历，中序遍历和后序遍历，分别输出对应的前缀表达式，中缀表达式和后缀表达式，最后进行表达式的计算。下面对项目的功能进行详细分析。

* + 1. 顾客信息录入功能

读取输入的顾客总数和顾客编号序列，将顾客按照编号分类，并初始化两个窗口的服务队列。

* + 1. 窗口服务模拟功能

此功能模拟窗口的服务流程，使用合适的数据结构来模拟窗口的服务队列，记录每个窗口当前正在服务的顾客和已完成的顾客。并根据窗口的处理速度，依次从队列中取出顾客进行服务，并更新队列状态。

* + 1. 顾客服务顺序输出功能

此功能旨在维护一个全局的顾客服务完成序列。根据需求，当A、B窗口同时完成服务时，优先将A窗口的顾客添加到输出序列中。最后，按照输出规范，格式化输出顾客的服务完成序列。

* + 1. 异常处理功能

实现异常处理机制，处理用户可能输入的非法信息，确保系统的稳定性和安全性。

1. **项目设计**

2.1.数据结构设计

基于项目分析，选择使用数据结构作为数据结构，主要基于以下几个考虑：

(1)先进先出原则：队列遵循先进先出（FIFO）的原则，即先到达的顾客先被服务。这与银行业务窗口的服务规则高度契合，确保了公平性；

(2)处理速度差异适应性：项目中明确提到A窗口的处理速度是B窗口的两倍。队列可以灵活地处理这种速度差异，通过维护两个队列来模拟不同窗口的处理速度。在实际操作中，可以根据窗口的处理能力动态地从各自队列中分配顾客；

(3)避免资源竞争与冲突：使用队列可以避免多个窗口同时尝试服务同一顾客的情况，从而减少了资源竞争和冲突的可能性。每个窗口可以从其对应的队列中按顺序获取顾客，确保服务的连续性和有序性；

(4)易于实现与管理：队列数据结构相对简单，易于实现和管理。在编程中，可以使用数组、链表等数据结构来实现队列，根据具体需求选择合适的实现方式。

2.2.结构体与类设计

2.2.1.LinkNode结构体的设计

LinkNode 结构体是一个用于构建链表节点的模板结构体。该结构体用于表示链表中的每个节点，其中包括节点存储的数据以及指向下一个节点的指针。本项目希望链表结点类可以直接访问链表结点，所以使用 struct 而不是 class 描述链表结点类。其数据成员和构造函数定义及含义如下：

T data：数据域，存储节点的数据。

LinkNode<T>\* link：指针域，指向下一个节点的指针。

LinkNode(LinkNode\* ptr = NULL)：构造函数，初始化指针域。

LinkNode(const T& item, LinkNode\* ptr = NULL)：构造函数，初始化数据域和指针域。

2.2.2.LinkedQueue 类的设计

该通用模板类LinkedQueue用于存储和管理数据元素，遵循先进先出的原则。该队列是基于链表数据结构实现的，链表节点由MyLinkNode 结构体表示，其中包含数据和指针域。此模板类LinkedQueue允许在队列的前端添加元素，以及从队列的前端移除元素。其数据成员、构造函数、析构函数、公有成员函数，运算符重载定义及含义如下：

LinkNode<T>\* front：指针指向队列的前端，即队列中的第一个元素

LinkNode<T>\* rear：指针指向队列的末端，即队列中的最后一个元素

LinkedQueue()：默认构造函数，创建一个空的队列

~LinkedQueue()：析构函数，清空队列并释放内存。

bool EnQueue(const T item)：将元素添加到队列的末尾

bool DeQueue(T& item)：移除队列的前端元素并将其值通过引用返回

bool getFront(T& item) const：获取队列前端元素的值

void makeEmpty()：清空队列，释放所有元素占用的内存

bool IsEmpty() const：检查队列是否为空

int getSize() const：获取队列中元素的数量

void outputNode(LinkNode<T>\* p,bool lastCunstomer = false):输出当前队列的第一个元素。

2.3.项目主体架构设计

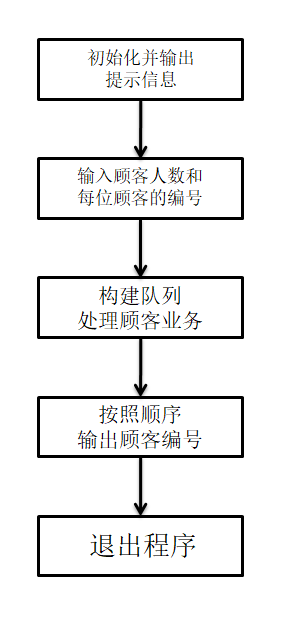


图 2.3 项目主体架构设计流程图

项目主体架构设计为：

(1)输入顾客总人数和每位顾客的编号；

(2)构建队列处理顾客业务；

(3)按照业务处理完成的顺序输出顾客的编号；

(4)退出程序。

**3.项目功能实现**

3.1.项目主体架构的实现

项目主体架构实现思路为：

(1)输入顾客总人数和每位顾客的编号：程序开始时，首先通过outputStart函数展示用户界面和输入要求，包括输入要求说明，输入的格式要求，输入数据的大小要求，确保输入的数据符合程序处理的格式；

(2)构建队列处理顾客业务：创建两个队列 Queue<int> A和 Queue<int> B，根据每位顾客的编号将其分配到不同的队列中，如果顾客的编号为奇数，则将其分配到A队列；如果顾客的编号为偶数，则将其分配到B队列。

(3)按照业务处理完成的顺序输出顾客的编号：一旦输入有效，程序调用outputCustomerNo 函数来按照业务处理完成的顺序输出顾客的编号。当 A 窗口每处理完两个顾客时，B 窗口处理完一个顾客，并且当不同窗口同时处理完两个顾客时，A 窗口的顾客优先输出。

(4)等待用户退出：最后，程序等待用户按下回车键退出。

3.2.顾客信息录入功能

录入顾客信息的功能通过一个do-while循环实现。每次录入信息前，程序调用makeEmpty方法将A、B两个队列清空，再调用inputCustomerInfo函数录入顾客信息，如果在录入信息时输入了非法数据，函数会立即返回false，重新进行循环和顾客信息录入。

3.3.顾客服务顺序输出功能

顾客服务顺序输出功能通过函数outputCustomerNo实现，其实现思路如下：

(1)初始化：函数接受三个参数：两个队列（用于存储顾客的信息）和一个整数 N，表示我们关注的顾客编号。在函数开始时，定义一些变量用于跟踪当前处理的顾客和相关信息，例如当前顾客的指针、计数器（用于记录输出的顾客数量）和一个布尔值（用于标记是否是最后一个被处理的顾客）。

(2)输出顾客的服务顺序

①同时处理两个队列的顾客：首先，函数进入一个循环，只要两个队列都不为空，就不断地从队列中取出顾客进行处理。在这个过程中，优先处理队列 A：从队列 A 中取出一个顾客，输出顾客的信息，并将该顾客从队列中移除。然后检查队列 A 是否还有下一个顾客，如果有，则继续处理下一个顾客，并进行相同的操作。接着，从队列 B 中取出一个顾客，输出其信息并移除，同时更新计数器，标记当前输出的顾客数量。

②处理剩余顾客：在同时处理两个队列的过程中，可能会出现一个队列为空而另一个队列仍有顾客的情况。因此，进入下一个阶段：首先，检查队列 A 是否为空，如果不为空，就继续输出队列 A 中剩余顾客的信息。然后，同样地检查队列 B，并输出其中的顾客信息。

通过以上的逻辑，函数确保了所有顾客按他们的到达顺序被逐个输出，不会遗漏。在服务过程中，系统会保持顾客的顺序处理，使得顾客的业务完成情况清晰可见。

3.4. 异常处理功能

在进行LinkedQuene类中私有数据成员等的动态内存申请时，程序使用new(std::nothrow)来尝试分配内存。new(std::nothrow)在分配内存失败时不会引发异常，而是返回一个空指针（NULL或nullptr），代码检查指针是否为空指针，如果为空指针，意味着内存分配失败，这时程序将执行以下操作：

(1)向标准错误流std::cerr输出一条错误消息"Error: Memory allocation failed."，指出内存分配失败；

(2)调用exit函数，返回错误码MEMORY\_ALLOCATION\_ERROR（通过宏定义方式定义为-1），用于指示内存分配错误，并导致程序退出。

**4.项目测试**

4.1.顾客信息录入功能测试

分别输入超过上下限的整数、偶数、浮点数、字符、字符串，可以验证程序对输入非法的情况进行了处理。

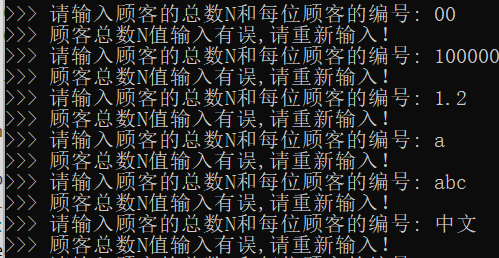


图 4.1.1 顾客信息录入功能验证（输入非法）

当输入合法时，程序继续运行。

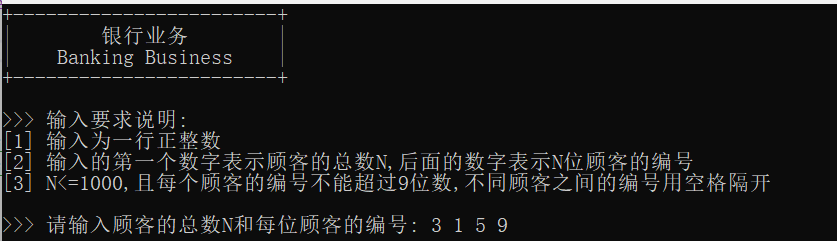


图 4.1.2 顾客信息录入功能验证（输入合法）

4.2.顾客服务顺序输出功能

合法输入顾客的总数和每位顾客的编号，可以验证程序能够正常计算并按照输出完成业务的顾客编号。

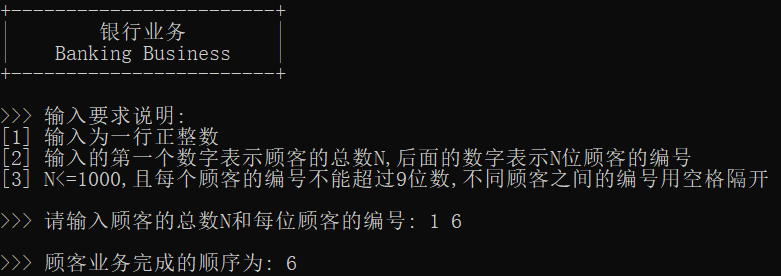


图 4.2.1 顾客服务顺序输出功能测试（测试一）

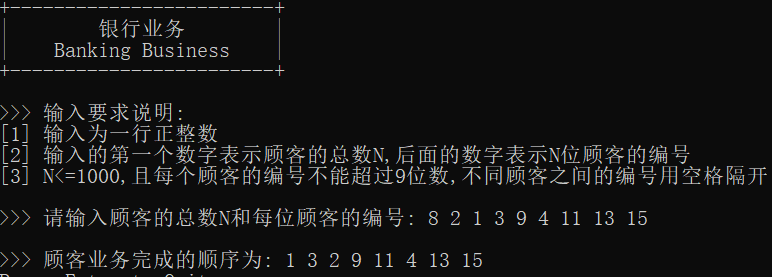


图 4.2.2 顾客服务顺序输出功能测试（测试二）

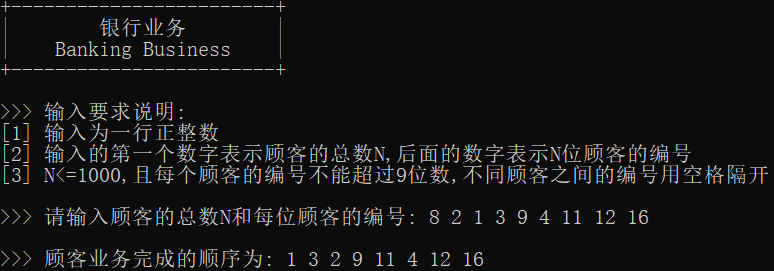


图 4.2.3 顾客服务顺序输出功能测试（测试三）