



同濟大學

TONGJI UNIVERSITY

数据结构课程设计

项目说明文档

银行业务

姓 名： 马小龙

学 号： 2353814

学 院： 计算机科学与技术学院（软件学院）

专 业： 软件工程

指导教师： 张颖

二〇二三年十一月五日

目录

第 1 章 项目分析.....	1
1.1 项目背景分析.....	1
1.2 项目需求分析.....	1
1.3 项目功能分析	1
1.3.1 顾客人数及编号录入功能	1
1.3.2 顾客分流功能	2
1.3.3 顾客序列输出功能	2
1.3.4 异常处理功能	2
第 2 章 项目设计.....	3
2.1 数据结构设计.....	3
2.2 结构体与类设计.....	3
2.2.1 链式队列结点 (QueueNode) 设计	3
2.2.1.1 概述.....	3
2.2.1.2 结构体定义.....	3
2.2.1.3 数据成员.....	4
2.2.1.4 构造函数.....	4
2.2.2 链式队列 (Queue) 设计	4
2.2.2.1 概述.....	4
2.2.2.2 Queue 类定义	4
2.2.2.3 私有数据成员.....	4
2.2.2.4 构造函数与析构函数.....	5
2.2.2.5 公有成员函数.....	5
2.3 项目框架设计.....	5
2.3.1 项目框架流程图	5
2.3.2 项目框架流程	6
第 3 章 项目功能实现.....	7
3.1 项目主体架构.....	7
3.1.1 实现思路.....	7
3.1.2 流程图.....	7
3.1.3 核心代码.....	7
3.1.4 示例.....	8
3.2 顾客人数、编号录入与分流功能.....	8
3.2.1 实现思路.....	8
3.2.2 流程图	9
3.2.3 核心代码	9

3.2.4 示例	11
3.3 顾客序列输出功能	11
3.3.1 实现思路	11
3.3.2 流程图	11
3.3.3 核心代码	12
3.3.4 示例	13
3.4 异常处理功能	13
3.4.1 输入非法的异常处理	13
3.4.2 动态内存申请失败的异常处理	13
3.4.3 队列的异常处理	14
第4章 项目测试	15
4.1 输入功能测试	15
4.1.1 输入顾客人数功能测试	15
4.1.2 顾客编号输入功能测试	15
4.2 分流及输出功能测试	16
第5章 相关说明	17
5.1 编程语言	17
5.2 Windows 环境	17
5.3 Linux 环境	17

第 1 章 项目分析

1.1 项目背景分析

随着银行业务量的持续增长，如何高效管理多个业务窗口，确保顾客能够及时获得服务，成为提升银行运营效率和顾客满意度的重要因素。银行通常设置多个窗口来分担业务，但不同窗口的处理速度差异往往会影响顾客的等待时间与整体服务效率。

在这种情况下，如何合理安排顾客的处理顺序，确保顾客按照业务完成的先后顺序得到处理，成为优化银行服务的重要课题。为了满足这一需求，本项目旨在实现根据特定信息将顾客安排到不同的窗口，根据客户的业务处理完成顺序打印顾客信息，方便工作人员进行分析，改进分流条件，提高业务处理顺序及客户满意的。

1.2 项目需求分析

项目实现的系统需要录入的顾客的编号，根据编号的奇偶将顾客分配到 A 或 B 窗口，最后应按照业务完成顺序输出顾客编号。

技术需求方面，系统需要支持 Windows 操作系统，使用 C++ 等主流编程语言，采用合适的数据结构，如队列，确保顾客数据的高效处理。系统应能动态处理不同数量的顾客数据，且不对顾客数量做严格限制。

1.3 项目功能分析

本项目旨在开发一套银行业务管理系统，需要实现顾客人数和编号录入，根据编号将顾客安排到不同窗口，最后根据业务处理顺序输出顾客编号。以下是项目主要功能的详细信息。

1.3.1 顾客人数及编号录入功能

允许工作人员根据输入顾客人数，并根据顾客人数输入一定数目的顾客编号；其中，顾客人数应该为 $[1 \sim 1000]$ 内的正整数，顾客编号应为不超过 int 型上限的正整数，不同数字应该用空格隔开。

1.3.2 顾客分流功能

能够按照顾客编号的奇偶性将顾客分流到 A、B 两个窗口。

1.3.3 顾客序列输出功能

根据顾客业务完成顺序输出顾客序列。A 窗口处理速度为 B 窗口的两倍；当 A、B 同时输出时，A 窗口的顾客优先输出；不考虑顾客到达时间间隔。

1.3.4 异常处理功能

程序对各种异常进行了基本处理，以提升系统稳定性。

第2章 项目设计

2.1 数据结构设计

本项目采用链式队列作为主要数据结构，旨在高效地处理顾客数据的存储与管理。链式队列能够灵活地适应顾客数量的变化，因为它使用动态内存分配，避免了固定大小数组的内存浪费。在顾客数量不确定的情况下，链式队列能够根据实际需求扩展内存空间，确保了系统能够处理不同规模的数据。

链式队列还具有高效的入队和出队操作，能够在常数时间内完成顾客数据的增删，这对于模拟银行窗口的业务处理非常重要。当顾客被分配到不同窗口时，队列能够按顺序管理顾客，确保在业务完成时按照正确的顺序输出顾客编号，同时，顾客的处理过程也不会受到不必要的排序操作影响。

此外，链式队列的结构使得内存管理更加灵活，能够高效地处理顾客数据的动态变化，避免了不必要的内存浪费和性能瓶颈。它简化了数据操作，确保了顾客按顺序排队，且能够适应窗口处理负载的变化，从而优化了整个银行业务的处理效率。

2.2 结构体与类设计

为了使链式队列更加通用，本链表设计为链表模板设计。

2.2.1 链式队列结点（QueueNode）设计

2.2.1.1 概述

QueueNode 用于储存信息，包括结点储存的具体信息内容和下一结点的地址。

2.2.1.2 结构体定义

```
template <typename T>
struct QueueNode {
    T data;
    QueueNode<T>* link;
    QueueNode(T d = 0, QueueNode* l = nullptr) :data(d), link(l) {}
};
```

2.2.1.3 数据成员

T data: 数据域, 存储结点的数据

Node<T>* link: 指针域, 指向下一个结点的指针

2.2.1.4 构造函数

QueueNode(T d = 0, QueueNode* l = nullptr)

初始化 data 与 link, 不含参默认为 data=0, l=nullptr;

2.2.2 链式队列 (Queue) 设计

2.2.2.1 概述

该通用模板类 Queue 用于表示链式队列。链式队列结点由 QueueNode 结构体表示, 其中包含数据和指向下一个结点的指针。该链式队列提供了一系列基本操作函数, 包括入队、出队、获取队首元素等, 以及链式队列的构造和析构, 满足了常见的链式队列操作需求。

2.2.2.2 Queue 类定义

```
template <typename T>
class Queue {
public:
    Queue():rear(nullptr), front(nullptr) {}
    ~Queue() { makeEmpty(); }
    void EnQueue(const T& x);
    T DeQueue();
    T GetFront();
    QueueNode<T>* FrontAddress() { return front; }
    QueueNode<T>* RearAddress() { return rear; }
    void makeEmpty();
    bool IsEmpty()const { return front == nullptr; }
private:
    QueueNode<T>* rear, *front;
};
```

2.2.2.3 私有数据成员

QueueNode<T>* rear, *front;

队伍首指针和队尾指针, 分别指向队伍首元素和尾元素。

2.2.2.4 构造函数与析构函数

`Queue();`

默认构造函数，创建一个空队列。

`~Queue();`

析构函数，释放队列的内存资源，包括所有结点的内存。

2.2.2.5 公有成员函数

`void EnQueue(const T& x);`

入队操作。

`T DeQueue();`

出队操作

`T GetFront();`

获取队头元素。

`QueueNode<T>* FrontAddress();`

获取队列头指针。

`QueueNode<T>* RearAddress();`

获取队列尾指针。

`void makeEmpty();`

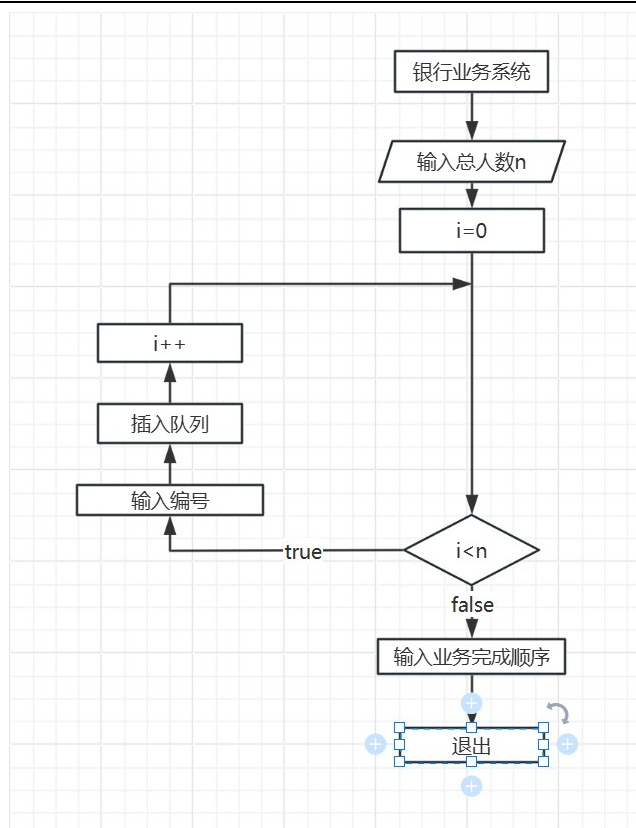
清空队列。

`bool IsEmpty()const;`

判断队列是否为空。

2.3 项目框架设计

2.3.1 项目框架流程图



2.3.2 项目框架流程

1. 进入银行业务系统；
2. 输入顾客人数和相应数目的顾客编号；
3. 根据顾客编号不同将其放入不同队列；
4. 通过循环结构输出顾客编号，实现业务处理情况的模拟；
5. 退出银行业务系统。

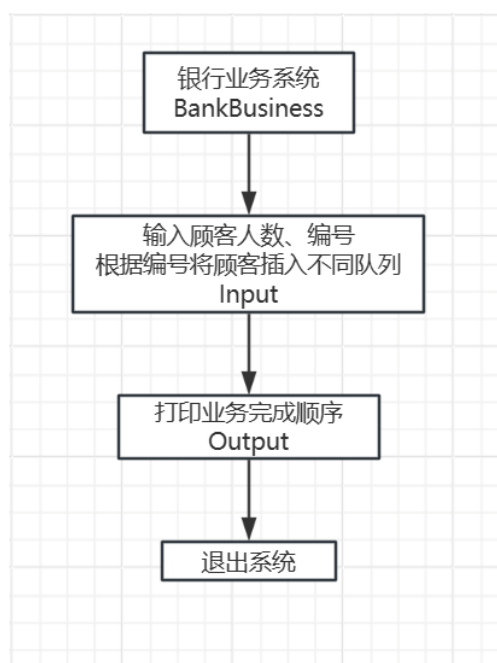
第3章 项目功能实现

3.1 项目主体架构

3.1.1 实现思路

1. 进入 BankBusiness 函数以进入银行业务系统。
2. 调用 Input 函数开始输入顾客人数、顾客编号，同时根据编号不同将顾客放入不同队列
3. 调用 Output 函数，借助循环结构按照业务完成顺序输入顾客编号。
4. 退出银行业务系统。

3.1.2 流程图



3.1.3 核心代码

```
void BankBusiness()
{
    std::cout << "+-----+" << std::endl;
    std::cout << "|                银行业务                |" <<
std::endl;
```

```

std::cout << "|                Bank Business                |" <<
std::endl;
std::cout << "+-----+" << std::endl;

Queue<int> A, B;
std::cout << "**顾客总数以及顾客编号输入输出规则：\n"
    << "1.顾客总数以及顾客编号应该为正整数；\n"
    << "2.顾客总数应在范围[1-<< MAX_CUSTOMER<<"]内，顾客编
号不大于 2147483647；\n"
    << "3.A 窗口的顾客优先输出。\\n";
while (!Input(A, B))
    ;

Output(A, B);
}

```

3.1.4 示例

```

+-----+
|                银行业务                |
|                Bank Business            |
+-----+
**顾客总数以及顾客编号输入输出规则：
1. 顾客总数以及顾客编号应该为正整数；
2. 顾客总数应在范围[1-1000]内，顾客编号不大于2147483647；
3. A窗口的顾客优先输出。

请依次输入顾客总数N以及N个顾客的编号；
2 2 3
业务处理完成顺序为：
3 2
进程已结束，退出代码为 0

```

3.2 顾客人数、编号录入与分流功能

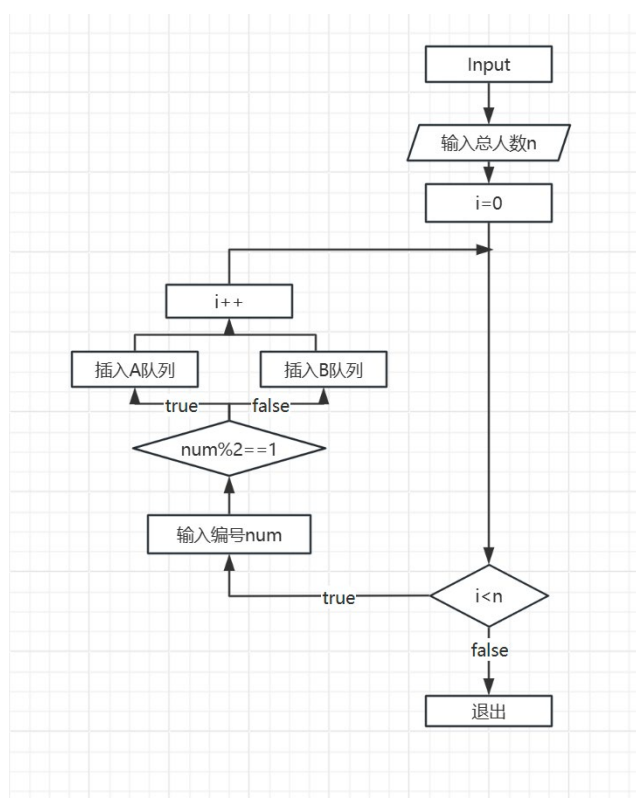
3.2.1 实现思路

顾客人数、编号录入与分流功能的函数名为 Input，该功能实现的思路为：

1. 首先输入顾客人数，输入不符合规范则需重新输入；

2. 利用 for 循环，循环次数为考生人数；
3. 在每一次循环中，输入顾客的编号，若编号不为正整数或者与已有编号重复，则需要重新输入；
4. 每输入成功一个顾客的编号，检查其奇偶性，并依据此将其放入 A 队列或 B 队列；
6. 所有顾客的编号输入完成后，退出循环，并退出 Input 函数。

3.2.2 流程图



3.2.3 核心代码

```

bool Input(Queue<int>& A, Queue<int>& B)
{
    std::cout << "\n 请依次输入顾客总数 N 以及 N 个顾客的编号； \n";
    double dN;
    std::cin >> dN;
    int N= static_cast<int>(dN); // 顾客总数
    if (std::cin.fail()||dN!=N) {
        std::cout << "输入顾客总数非法！ \n";
    }
}

```

```
        ClearBuffer();
        return false;
    }
    if (N <= 0 || N > MAX_CUSTOMER) {
        std::cout << "输入顾客总数不在范围[1-" << MAX_CUSTOMER << "]"
        内! \n";
        ClearBuffer();
        return false;
    }
    for (int i = 0; i < N; ++i) {
        double d_customer;
        std::cin >> d_customer;
        int customer = static_cast<int>(d_customer);
        if (std::cin.fail() || d_customer != customer) {
            std::cout << "输入顾客编号非法! \n";
            ClearBuffer();
            return false;
        }
        if (customer <= 0) {
            std::cout << "输入顾客编号不是正整数! \n";
            ClearBuffer();
            return false;
        }
        if (IsSame(customer, A) || IsSame(customer, B)) {
            std::cout << "重复输入相同编号的顾客! \n";
            ClearBuffer();
            return false;
        }
        if (customer % 2 == 1) {
            A.Enqueue(customer); // 奇数编号到 A 窗口
        }
        else {
            B.Enqueue(customer); // 偶数编号到 B 窗口
        }
    }
}
```

```

    }
    return true;
}

```

3.2.4 示例



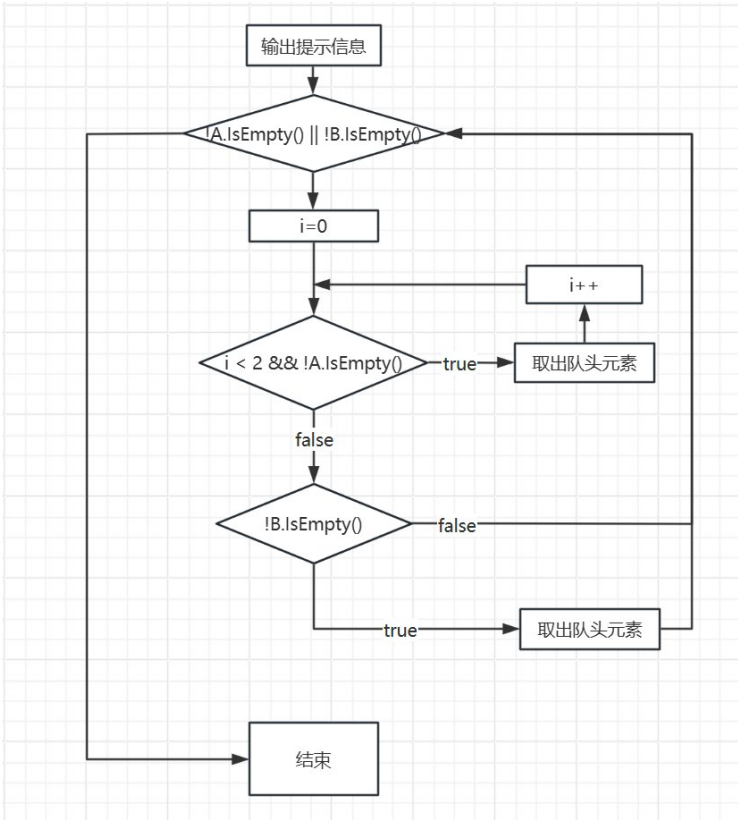
3.3 顾客序列输出功能

3.3.1 实现思路

输出顾客序列功能的函数名为 Output，输出顾客序列功能实现的思路为：

1. 打印提示信息；
2. 采用 While 循环的判断 A 队列或 B 队列是否为空，若不为空则进入循环；
3. 先取出 A 队列的两个编号并输出，直到 A 队列为空；再去除 B 队列的一个元素并输出，直到 B 队列为空；
4. 当 A 队列与 B 队列均为空时，退出 While 循环，同时退出 Output 函数；

3.3.2 流程图



3.3.3 核心代码

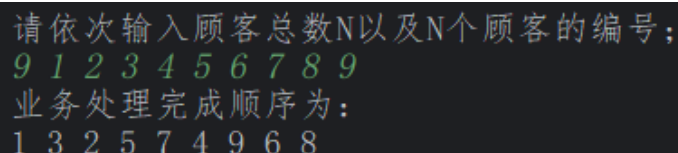
```
void Output(Queue<int>& A, Queue<int>& B)
{
    std::cout << "业务处理完成顺序为: \n";
    bool first = true;
    while (!A.IsEmpty() || !B.IsEmpty()) {
        for (int i = 0; i < 2 && !A.IsEmpty(); ++i) {
            if (!first)
                std::cout << " ";
            std::cout << A.DeQueue();
            first = false;
        }
        if (!B.IsEmpty()) {
            if (!first)
                std::cout << " ";
            std::cout << B.DeQueue();
            first = false;
        }
    }
}
```

```

    }
}
}

```

3.3.4 示例



```

请依次输入顾客总数N以及N个顾客的编号：
9 1 2 3 4 5 6 7 8 9
业务处理完成顺序为：
1 3 2 5 7 4 9 6 8

```

3.4 异常处理功能

3.4.1 输入非法的异常处理

输入顾客人数以及顾客编号后，都会对其进行合法性检查，当全部输入内容没有错误时，Input 函数返回 true，进行下一步操作，当输入内容有错误时，会清除缓冲区，Input 函数返回 false，BankBusiness 函数会重新调用该函数，进行客户人数和编号输入。

客户人数及编号均应该为正整数，其中用户编号不能与已有编号重复，这一点调用 IsSame 函数检查。

3.4.2 动态内存申请失败的异常处理

在进行 Queue 类动态内存申请时，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，对于内存分配失败，则使用 exit 函数退出程序，并限定返回值。例如：

```

void Queue<T>::EnQueue(const T& x)
{
    if (front == nullptr) {
        front = rear = new(std::nothrow) QueueNode<T>(x, nullptr);
        if (front == nullptr) {
            std::cout << "内存分配错误！\n";
            exit(MEMORY_ALLOCATION_ERROR);
        }
    }
}

```



```
    }  
}  
else {  
    rear = rear->link = new(std::nothrow) QueueNode<T>(x, nullptr);  
    if (rear == nullptr) {  
        std::cout << "内存分配错误! \n";  
        exit(MEMORY_ALLOCATION_ERROR);  
    }  
}  
}
```

3.4.3 队列的异常处理

在 Queue 类中，DeQueue 函数需要对已有队头元素进行删除工作。实际使用时，可能存在队列为空却调用 DeQueue 函数的情况。因此在删除前需要检查队列是否为空；对于队列为空的，则使用 exit 函数退出程序，并限定返回值。

第4章 项目测试

4.1 输入功能测试

4.1.1 输入顾客人数功能测试

分别输入超过上下限的整数、负数、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。

```
请依次输入顾客总数N以及N个顾客的编号；
-11111111111111111111
输入顾客总数非法！

请依次输入顾客总数N以及N个顾客的编号；
999999999999999999999999
输入顾客总数非法！

请依次输入顾客总数N以及N个顾客的编号；
1.1
输入顾客总数非法！

请依次输入顾客总数N以及N个顾客的编号；
c
输入顾客总数非法！

请依次输入顾客总数N以及N个顾客的编号；
skcanl
输入顾客总数非法！
```

当输入合法时，程序继续运行，等待输入顾客编号。

```
请依次输入顾客总数N以及N个顾客的编号；
9
|
```

4.1.2 顾客编号输入功能测试

分别输入超过上下限的整数、负数、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。

```
输入顾客编号非法！
0 2870 1 0
请依次输入顾客总数N以及N个顾客的编号；

输入顾客编号非法！
0 c 1 0
请依次输入顾客总数N以及N个顾客的编号；

输入顾客编号非法！
0 0 0 1 0
请依次输入顾客总数N以及N个顾客的编号；

输入顾客编号非法！
0 0000000000000000 1 0
请依次输入顾客总数N以及N个顾客的编号；

输入顾客编号非法！
0 -0000000000000000
请依次输入顾客总数N以及N个顾客的编号；
```

当输入重复时，需要重新输入。

```
请依次输入顾客总数N以及N个顾客的编号；
9 1 2 3 4 5 6 7 2 6
重复输入相同编号的顾客！

请依次输入顾客总数N以及N个顾客的编号；
```

当输入合法时，程序继续运行，等待输入顾客编号或结束输入。

```
请依次输入顾客总数N以及N个顾客的编号；
9 1 2 3 4 5 6 7 8 9
业务处理完成顺序为：
```

4.2 分流及输出功能测试

根据输出结果，查验分流、输出功能是否符合要求。输入合法数据后，观察输出结果。

```
+-----+
|               银行业务               |
|               Bank Business           |
+-----+

**顾客总数以及顾客编号输入输出规则：
1. 顾客总数以及顾客编号应该为正整数；
2. 顾客总数应在范围[1-1000]内，顾客编号不大于2147483647；
3. A窗口的顾客优先输出。

请依次输入顾客总数N以及N个顾客的编号；
20 2 4 6 8 12 15 16 17 13 99 51 66 78 72 49 43 48 101 113 114
业务处理完成顺序为：
15 17 2 13 99 4 51 49 6 43 101 8 113 12 16 66 78 72 48 114
进程已结束，退出代码为 0
```

第 5 章 相关说明

5.1 编程语言

本项目全部 .cpp 文件以及 .h 文件均使用 C++ 编译完成，使用 UTF-8 编码。

5.2 Windows 环境

Windows 系统: Windows 11 x64

Windows 集成开发环境: CLion 2024

工具集: MinGW 11.0 w64

5.3 Linux 环境

基于 Linux 内核的操作系统发行版: Ubuntu 24.04.1

Linux 命令编译过程为:

1. 定位包含项目所在文件夹，包括 .pp 与 .h 文件；具体命令为: `cd /home/bruce/programe/bank_business`

2. 编译项目，生成可执行文件；具体命令为: `g++ -static -o bank_business_linux bank_business.cpp my_queue.h;`

其中指令含义分别为:

`g++`: 调用 GNU 的 C++ 编译器

`-static`: 使用静态链接而非动态链接，将所有依赖库直接嵌入到可执行文件，文件存储空间变大，但可以单独运行

`-o bank_business_linux`: `-o` 表示输出文件选项，`bank_business_linux` 为可执行文件名

`bank_business.cpp my_queue.h`: 编译所需要的文件。

3. 运行可执行文件；具体命令为 `./ bank_business_linux`

```
bruce@Jarvis: ~/programe/bank_business
bruce@Jarvis:~/programe/bank_business$ cd /home/bruce/programe/bank_business
bruce@Jarvis:~/programe/bank_business$ g++ -static -o bank_business_linux bank_b
usiness.cpp my_queue.h
bruce@Jarvis:~/programe/bank_business$ ./bank_business_linux
+-----+
|           银行业务           |
|           Bank Business      |
+-----+

**顾客总数以及顾客编号输入输出规则：
1.顾客总数以及顾客编号应该为正整数；
2.顾客总数应在范围[1-1000]内，顾客编号不大于2147483647；
3.A窗口的顾客优先输出。

请依次输入顾客总数N以及N个顾客的编号；

```