项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名：陈垲昕

学　　号：[redacted]

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

可通过按住 Ctrl 并单击访问说明文档各个模块：

数据结构课程设计项目说明文档
——银行业务
1. 分析
1.1 背景分析
1.2 功能分析
2.设计
2.1 主要数据结构设计
链表结点类设计
链表队列类设计
2.2 系统类设计
流程图
3.核心功能实现
3.1 对 A 与 B 窗口顾客的分流生成最终的顺序结果
3.2 输入以及检查程序
1.顾客总数输入检查
2.结点空间分配检查
3.队列分流
4.测试
4.1 常规结果测试
Task 1 正常测试，A 窗口人多
Task 2 正常测试，B 窗口人多
Task 3 最小顾客数测试
4.2 错误结果测试
Task 1 顾客数量为负数

1. 分析

1.1 背景分析

排队问题是常见的问题，研究排队多窗口问题可以提高处理的通勤效率与速度。本题作为一个基本排队问题的模拟，是这类问题的一个最简单模型：

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 主要数据结构设计

银行业务问题是典型的队列问题，具有典型的“先进先出”FIFO 特征，故考虑使用链表形式的队列存储

本课程设计的队列为链表形式存储结构，实现包括信息获取，出入队列，清空，打印的操作。对结点类，包含 int 类型的序号与下一个指针域成员，实现了默认，赋值，复制构造函数，对当前节点的序号有判断奇数还是偶数的方法

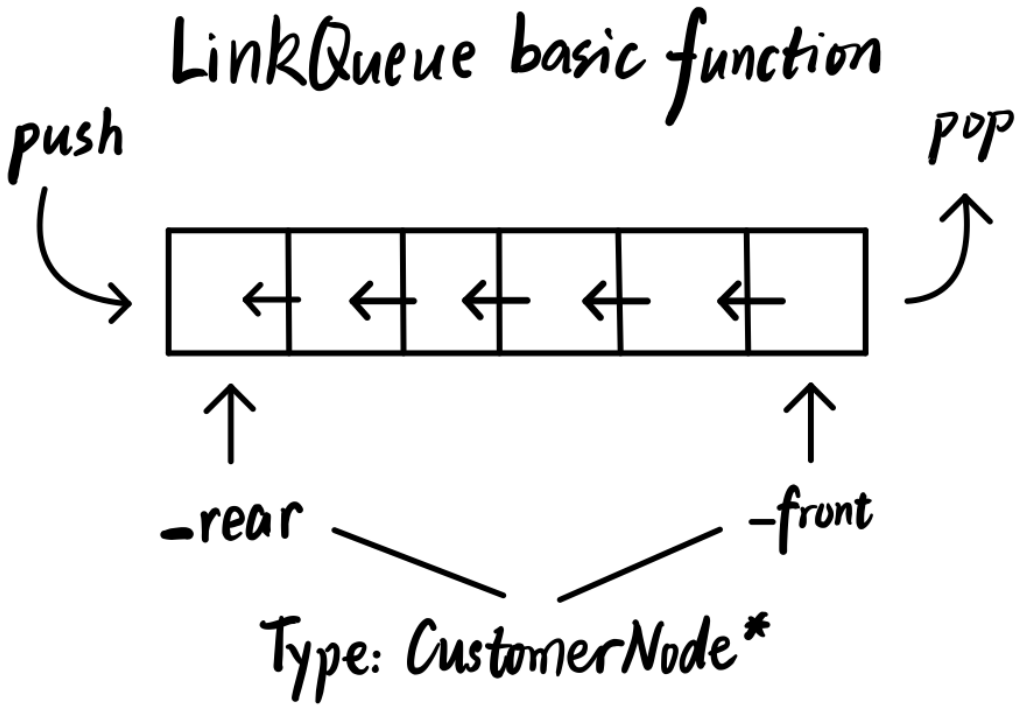
链表结点类设计

```
class CustomerNode {
	friend class LinkQueue;
public:
	//构造函数
	CustomerNode() :_number(-1), _next(nullptr) {}
	CustomerNode(const int& initNumber) ;

	CustomerNode(CustomerNode* node)
```

```
        //输出函数
        void output(std::ostream& os = std::cout) { os << _number; }

        //判断当前节点的数值是奇数还是偶数
        bool isOdd()const { return _number % 2 == 1; }
        bool isEven()const { return _number & 2 == 0; }
private:
        //数值与后继节点成员
        int _number;
        CustomerNode* _next;

};
```

链表队列类设计

```
class LinkQueue {
public:
        //构造与析构函数
        LinkQueue();

        ~LinkQueue() { clear(); }
        //信息获取
        size_t size()const { return _size; }
        bool empty()const { return size() == 0; }
        //队列单元操作
        void push( CustomerNode* node);
        void pop();
        CustomerNode* top()const { return _front->_next; }
        //队列操作
        void clear();
        void printQueue();

private:
        //头尾结点指针，方便出栈入栈
        CustomerNode* _front, * _rear;

        size_t _size;
};
```



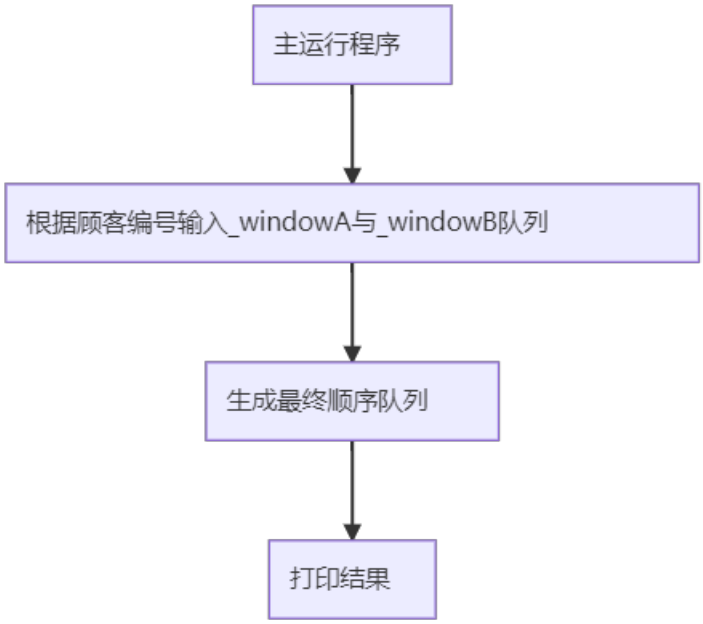
2.2 系统类设计

本课设通过 BankWindow 类实现银行窗口管理系统，此类成员包括：1.代表窗口 A，窗口 B 的队列，以及用于存储最终生成的顺序队列；2.默认构造与析构函数；3.任务运行函数，包括：输入两个队列并根据顾客编号分类,生成顾客顺序队列并打印

```
class BankWindow {
public:
	//默认构造与析构函数
	BankWindow() = default;
	~BankWindow() { _windowA.clear(); _windowB.clear(); }

	//任务运行函数
	void run();

	void inputAndDivide();		//输入，根据顾客编号分类
	void generateOrderQueue();	//生成顾客顺序队列
```

```
private:
	LinkQueue _windowA;                    //窗口A 队列
	LinkQueue _windowB;                    //窗口A 队列

	LinkQueue _orderQueue;          //最终生成的队列
};
```

流程图



3.核心功能实现

3.1 对 A 与 B 窗口顾客的分流生成最终的顺序结果

在一次循环内弹出 2 个 windowA 队列的顾客和 1 个 windowB 队列的顾客。两个队列弹空后不再弹出顾客。注意队列尾部不能有多余的空格，队尾数据输出要单独处理。

```
void BankWindow::generateOrderQueue()
{
    //只要还有没处理的顾客序号在 A 或 B 队列中
      while (!_windowA.empty() || !_windowB.empty())
      {
        //若 A 头部有顾客
            if (_windowA.top())
            {
            //添加该顾客
                  CustomerNode* customer1=new CustomerNode(_windowA.top
());
                  _orderQueue.push(customer1);
            //出队列
                  _windowA.pop();
            }
             //若 A 头部有顾客
            if (_windowA.top())
            {
             //添加该顾客
                  CustomerNode* customer2 = new CustomerNode(_windowA.t
op());
                  _orderQueue.push(customer2);
            //出队列
                  _windowA.pop();
            }
        //若 B 头部有顾客
            if (_windowB.top())
            {
            //添加该顾客
                  CustomerNode* customer3 = new CustomerNode(_windowB.t
op());
                  _orderQueue.push(customer3);
             //出队列
                  _windowB.pop();
            }
      }
}
```

3.2 输入以及检查程序

1.顾客总数输入检查

对初始输入的顾客数量进行检查，要求为输入正整数

```
std::cout << "请输入顾客总数 N(N<1000):";
std::cin >> numberOfCustomer;
while (numberOfCustomer <= 0)
{
	std::cout << "请重新输入客户数量，要求为正整数:";
	std::cin >> numberOfCustomer;
}
```

2.结点空间分配检查

若分配失败，退出程序

```
CustomerNode* customer = new CustomerNode(mark);
if (!customer)
{
	std::cerr << "错误，内存分配失败，将退出程序" << std::endl;
	exit(1);
}
```

3.队列分流

根据顾客节点序号的奇偶性，决定进入的队列

```
if (customer->isOdd())
{
	_windowA.push(customer);
}
else
{
	_windowB.push(customer);
}
```

4.测试

4.1 常规结果测试

Task 1 正常测试，A 窗口人多

测试样例：

```
8
2 1 3 9 4 11 13 15
```

预期结果：

1 3 2 9 11 4 13 15

实际结果：

```
请输入顾客总数N(N<1000):8
请依次输入8位顾客的编号。编号为奇数的顾客需要去A窗口办理业务，编号为偶数的顾客需要去B窗口办理业务。数字间以空格分开）：
2 1 3 9 4 11 13 15
顾客业务完成的顺序为:
1 3 2 9 11 4 13 15
```

Task 2 正常测试，B 窗口人多

测试样例：

```
8
2 1 3 9 4 11 12 16
```

预期结果：

1 3 2 9 11 4 12 16

实际结果：

```
请输入顾客总数N(N<1000):8
请依次输入8位顾客的编号。编号为奇数的顾客需要去A窗口办理业务，编号为偶数的顾客需要去B窗口办理业务。数字间以空格分开）：
2 1 3 9 4 11 12 16
顾客业务完成的顺序为:
1 3 2 9 11 4 12 16
```

Task 3 最小顾客数测试

测试样例：

```
1
6
```

预期结果：

6

实际结果：

```
请输入顾客总数N(N<1000):1
请依次输入1位顾客的编号。编号为奇数的顾客需要去A窗口办理业务，编号为偶数的顾客需要去B窗口办理业务。数字间以空格分开）：
6
顾客业务完成的顺序为：
6
```

4.2 错误结果测试

Task 1 顾客数量为负数

测试样例：

```
-1
```

预期结果：提示不合法并要求用户输入合法的正整数

实际结果：

```
请输入顾客总数N(N<1000):-2
请重新输入客户数量，要求为正整数:2
请依次输入2位顾客的编号。编号为奇数的顾客需要去A窗口办理业务，编号为偶数的顾客需要去B窗口办理业务。数字间以空格分开）：
```