

数据结构课程设计

——银行业务

项目说明文档

作 者 姓 名： 韦世贸

学 号： 2351131

指 导 老 师： 软件学院

专 业： 软件工程

同济大学

Tongji University

1 介绍

1.1 背景介绍

，本题为现实中银行业务的简化版，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。银行业务办理明显是一直需要的一种技能，也希望之后随着学习的深入可以解决更多与实际相关的问题。

1.2 功能介绍

项目功能要求：通过输入为一行正整数，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。

在输出时需要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，但是要求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

该项目明显是对队列的一个应用实现典例，此题采用了链式队列进行操作，构建三个链式队列A，B，All\_order用以解决问题。

2.2 类结构设计

模板类以及具体链式队列的实现类，以头尾指针作为数据成员，类中包含了队列的基本操作如进出队列，获取队首元素，获取队列长度。额外写了一个链表结点结构体。比较重要的部分是重载了链式队列的输出运算符

template<class T>

ostream& operator<< (ostream& os, LinkedQueue<T>& Queue)

{

LinkNode<T>\* current = Queue.front;

for (LinkNode<T>\* current = Queue.front; current != NULL; current = current->link)

os << current->data << " \n"[current->link == NULL];

return os;

}

2.3 成员与操作设计

私有成员包含了头尾指针

private:

LinkNode<T>\* front, \* rear;

以下为队列的基本操作

template<class T>

class LinkedQueue

{

public:

LinkedQueue() :front(NULL), rear(NULL){}

~LinkedQueue(){}

bool isEmpty()

{

return front == NULL;

}

void makeEmpty()

{

LinkNode<T>\* del;

while (front != NULL)

{

del = front;

front = front->link;

delete del;

}

}

//进队列

bool EnQueue(T x)

{

if (front == NULL) {

//当队列为空时，新结点作为队列的第一个结点，同时作为队首和队尾

front = rear = new LinkNode<T>(x);

if (front == NULL) {

cout << "存储分配失败！" << endl;

return false;

}

}

else {

//队列不空，将新结点接到队尾后

rear->link = new LinkNode<T>(x);

if(rear->link==NULL) {

cout << "存储分配失败！" << endl;

return false;

}

rear = rear->link; //更新队尾

}

return true;

}

//出队列

bool DeQueue(T& x)

{

if (isEmpty()) //队列为空

return false;

LinkNode<T>\* del = front;

x = del->data;

front = front->link;

delete del;

return true;

}

//取得队首元素

bool getFront(T& x)

{

if (isEmpty()) //队列为空

return false;

x = front->data;

return true;

}

//取得队列长度

int getLength()

{

LinkNode<T>\* current = front;

int len = 0;

while (current != NULL) {

current = current->link;

len++;

}

return len;

}

//输出运算符重载

template<class T>

friend ostream& operator<< (ostream& os, LinkedQueue<T>& Queue);

3 实现

3.1 银行业务功能的实现

3.1.1 银行业务功能流程简介

银行业务功能是将输入的编号判断奇、偶以后分别进入到A、B中，根据题目所给规则，将队列元素转移到All\_order中，若还有剩余，按B中有无元素进行分类讨论，按规则将剩余元素转移到All\_order中，最后输出All\_order中元素

3.1.2 银行业务功能核心代码

void Sort()

{

int data;

while (A.getLength() >= 2 && !B.isEmpty()) {

//当a窗口处理完两个业务时，b窗口处理完一个业务且先输出a窗口的客户编号

A.DeQueue(data);

All\_order.EnQueue(data);

A.DeQueue(data);

All\_order.EnQueue(data);

B.DeQueue(data);

All\_order.EnQueue(data);

}

//处理剩余的客户

if (A.getLength() < 2 && !B.isEmpty()) {

//a窗口的输出即将结束

int alen = A.getLength();

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

A.DeQueue(data);

All\_order.EnQueue(data);

}

//此时只剩余b窗口的输出

int blen = B.getLength();

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

B.DeQueue(data);

All\_order.EnQueue(data);

}

}

else if (B.isEmpty()) {

//此时只剩余a窗口的输出

int alen = A.getLength();

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

A.DeQueue(data);

All\_order.EnQueue(data);

}

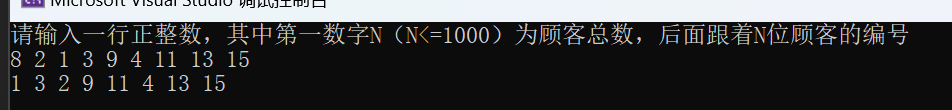
}

}

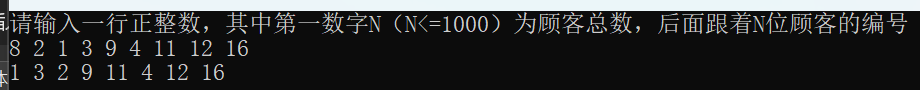
3.1.3银行业务功能截屏示例

正常的银行业务功能展示，未涉及边界情况。

测试一：8 2 1 3 9 4 11 13 15

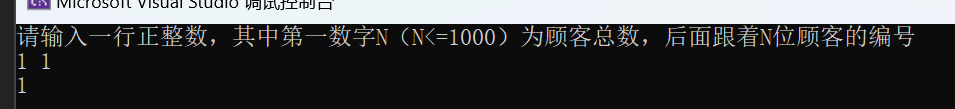


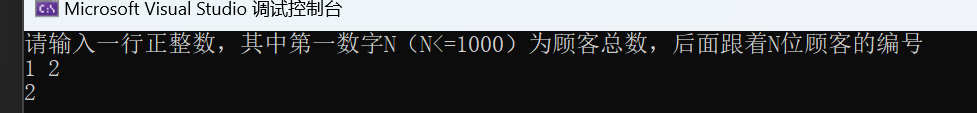
测试二：8 2 1 3 9 4 11 12 16



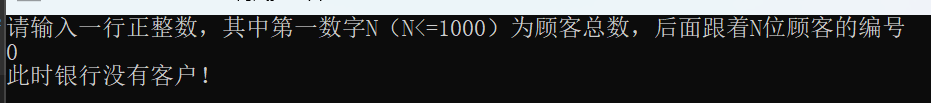
3.1.4 银行业务功能边界测试

测试一（只有一个队列有元素）：





测试二：两个队列中均没有元素

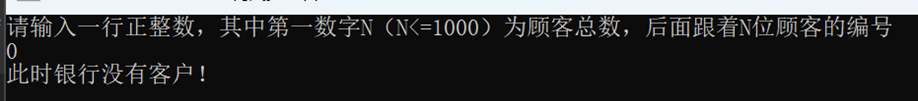


4 .设计小结

在这次银行业务系统的编程中我掌握了队列的基本类实现，用模板类先进行声明再进行具体实现，学会了用代码完成队列的各种基本操作如进出队列，获取队首元素，获取队列长度。在这次的银行业务设计中，我完成了基本的要求，也在一些地方做出了改进。

改进一、我使用的队列为链式队列，理论上N可以为无限大，不受限制，通用性更加强。

改进二、我进行了输入0时的情况的边界考虑



当然除了改进以外我也在这个程序中学到了不少：我加深了对于模板类的理解，还巩固了队列的知识，最重要的是更加理解了封装的概念，对类的封装有了更加深刻的认识。但同样也有地方可以改进：对两个队列的处理并没有引入类，也没有把处理函数封装进类中，希望以后能在编程中应用起来这个方法，提高编程的安全性，同时也能够学到更多东西。