项目说明文档

**数据结构课程设计**

**——银行业务**

作 者 姓 名： 吴英豪

学 号： 1953608

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

1.分析

1.1背景分析

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个顾客时，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。

(3)测试用例：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 输入 | 输出 | 说明 |
| 1 | 8 2 1 3 9 4 11 13 15 | 1 3 2 9 11 4 13 15 | 正常测试，A窗口人多 |
| 2 | 8 2 1 3 9 4 11 12 16 | 1 3 2 9 11 4 12 16 | 正常测试，B窗口人多 |
| 3 | 1 6 | 6 | 最小N |

2.设计

2.1数据结构设计

银行排队处理业务有着这样的特点：队伍从队头到队伍，总是队头的人先办理业务，办理完业务之后就可以离开；新来的人需要先排在队尾等待。这样现今先出的特点正好符合队列这一数据结构的特点。

综上，使用队列为此系统最主要的数据结构。

题目中主要涉及队列的有A窗口**queue\_A,** B窗口**queue\_B**,和用于存放最终队列结果的队列**queue\_result**

2.2类结构设计

（1）队列结点QueueNode

用于存放与队列有关的结点信息T \_data; 以及队列下一个元素的指针QueueNode<T>\* \_next;

（2）队列Queue

队列里包含队首指针QueueNode<T>\* \_first;以及队尾指针QueueNode<T>\* \_last;，用于队首的出队和队尾的入队。同时含有队列的基本操作（入队，出队）

注：因为队列Queue需要访问队列结点QueueNode的成员（尤其是私有成员），所以将Queue设置为QueueNode的友元.

2.3成员与操作设计

**(1)队列结点类（QueueNode）**

template<class T>

class QueueNode

{

private:

T \_data;

QueueNode<T>\* \_next;

public:

QueueNode();

QueueNode(T data);

~QueueNode();

friend class Queue <T>;

};

**(2)队列类（Queue）**

template<class T>

class Queue

{

private:

QueueNode<T>\* \_first; //队头指针

QueueNode<T>\* \_last; //队尾指针

public:

Queue();

~Queue();

bool isEmpty()const; //判断队列是否为空

int getLength()const; //获得队列的长度

void inQueue(T data); //队尾入队

void outQueue(T& data); //队头出队

};

2.4文件设计

本系统支持windows系统和Linux系统。

(1)Windows系统

windows系统的相关文件在文件夹05\_1953608\_WuYinghao\_Windows中，

其中类的定义以及其成员函数的定义在05\_1953608\_WuYinghao.h中;main函数以及相关的函数定义在05\_1953608\_WuYinghao.cpp中.05\_1953608\_WuYinghao.exe为Windows平台下的可执行文件。

**05\_1953608\_WuYinghao.cpp和05\_1953608\_WuYinghao.h的编码格式均为**

**简体中文(GB2312)-代码页 936**

(2)Linux 系统

Linux系统的相关文件在文件夹05\_1953608\_WuYinghao\_Linux中,

其中类的定义以及其成员函数的定义在05\_1953608\_WuYinghao\_Linux.h中;main函数以及相关的函数定义在05\_1953608\_WuYinghao\_Linux.cpp中.05\_1953608\_WuYinghao.out为Linux平台下的可执行文件。

**05\_1953608\_WuYinghao\_Linux.cpp和05\_1953608\_WuYinghao\_Linux.h的编码格式均为Unicode(UTF-8 无签名)-代码页65001**

2.5算法设计

本题目采用一遍输入，一遍入队的操作，用户输入完成之后入队同时结束。考虑到题目中要求编号为奇数的顾客到A窗口,编号为偶数的顾客到B窗口，所以在输入时判断编号，依据编号入队。

因为优先输出A窗口,当两个队列queue\_A和queue\_B均不为空的时候，按照从队首queue\_A两个元素出队queue\_B一个元素出队的顺序出队。当其中某一个队列为空，若另一个队列不为空，则输出这个队列中剩余的所有元素，否则输出结束。

3.实现

3.1按序号入队功能的实现

3.1.1功能说明

每当用户输入一个合法的数据之后，判断这个数的奇偶性，若是奇数，从A窗口queue\_A队尾入队；若是偶数，则从B窗口queue\_B队尾入队。执行队列的入队操作inQueue(T data)即可。

在用户输入结束之后，按照两个A窗口元素出队一个B窗口元素出队，一个队为空另一个队全部元素出队的顺序，将节点全部暂存的结果队列queue\_result中，再让结果队列输出所有的值。

3.1.2按序号入队功能核心代码

第一部分：

template<class T>

void Queue<T>::inQueue(T data)

{

try

{

QueueNode<T>\* newNode = new QueueNode<T>(data);

if (NULL == \_first->\_next) //队列为空

{

\_first->\_next = newNode;

\_last->\_next = newNode;

newNode->\_next = NULL;

}

else //队列中至少有一个元素

{

\_last->\_next->\_next = newNode;

newNode->\_next = NULL;

\_last->\_next = newNode;

}

}

catch (bad\_alloc& ba)

{

cerr << ba.what();

}

}

**第二部分：**

cout << "\t\t请输入" << customer\_num << "个顾客的序号:" << endl << "\t\t";

for (int i = 0; i < customer\_num; i++)

{

while (true)

{

cin >> order;

if (cin.fail())

{

cerr << "\t\tERROR.输入失败,请重新输入" << endl << "\t\t";

cin.clear();

cin.ignore();

cin.get();

}

else if (order <= 0)

{

cerr << "\t\tERROR.序号必须为正整数,请重新输入" << endl << "\t\t";

}

else

break;

}

if (order % 2) //order是奇数，去A窗口

{

queue\_A.inQueue(order); //奇数去A窗口

}

else

{

queue\_B.inQueue(order); //偶数去B窗口

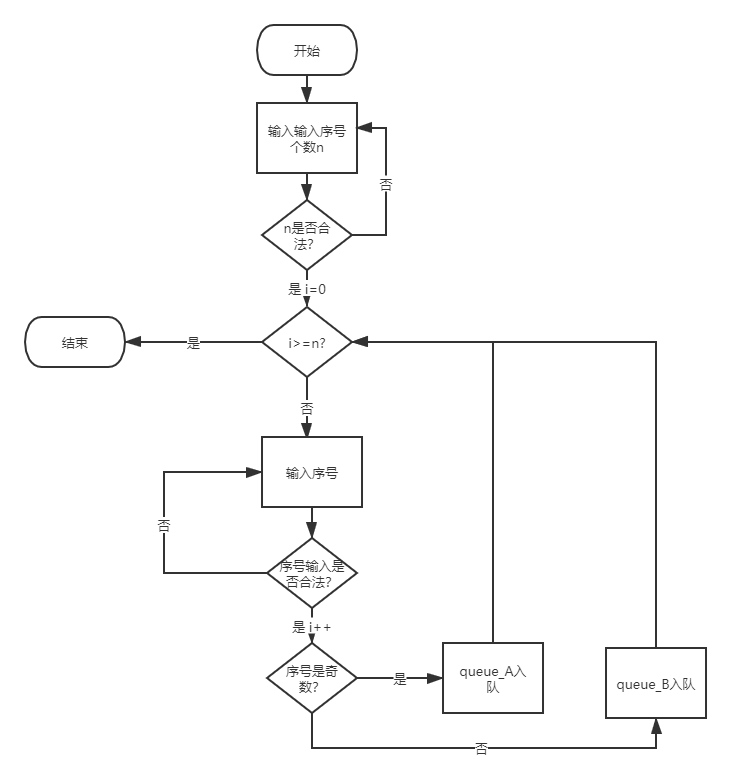
}

}

3.1.3按序号入队功能截屏展示



3.1.3按序号入队功能流程图



3.2 排序功能的实现

3.2.1功能说明

执行按序号入队的操作之后，将按照A窗口优先，A窗口出两个元素，B窗口出一个元素的顺序出队，结果将存在结果队列queue\_result中。

在出队的时候需要注意以下几点

(1)queue\_A一次出两个元素，且优先出队，若queue\_A为空，则将queue\_B剩余的所有元素存储在结果队列中。同理，若queue\_B为空，则将queue\_A中剩余的所有元素都存储在结果队列中。

(2)当queue\_A出队时，若A队仅剩一个元素，则仅(最后)一个元素出队，后续操作按照一队为空处理

3.2.2 核心代码展示

cout << "\t\t业务处理完成的顺序是：";

while ((!queue\_A.isEmpty()) && (!queue\_B.isEmpty()))//当两个队列均不为空时

{

queue\_A.outQueue(temp); //队A先出队一个元素

queue\_result.inQueue(temp);

if (!queue\_A.isEmpty())

**//判断队A是否为空，即A中原先是否仅剩一个元素**

{

queue\_A.outQueue(temp); //队A出队第二个元素

queue\_result.inQueue(temp);

}

queue\_B.outQueue(temp); //队B出队一个元素

queue\_result.inQueue(temp);

}

**//此时已经有一个队列为空**

if (queue\_A.isEmpty()) //队列A空

{

int b\_length = queue\_B.getLength();

int temp;

for (int i = 0; i < b\_length; i++)

{

queue\_B.outQueue(temp);

queue\_result.inQueue(temp);

}

}

if (queue\_B.isEmpty()) //队列B空

{

int a\_length = queue\_A.getLength();

int temp;

for (int i = 0; i < a\_length; i++)

{

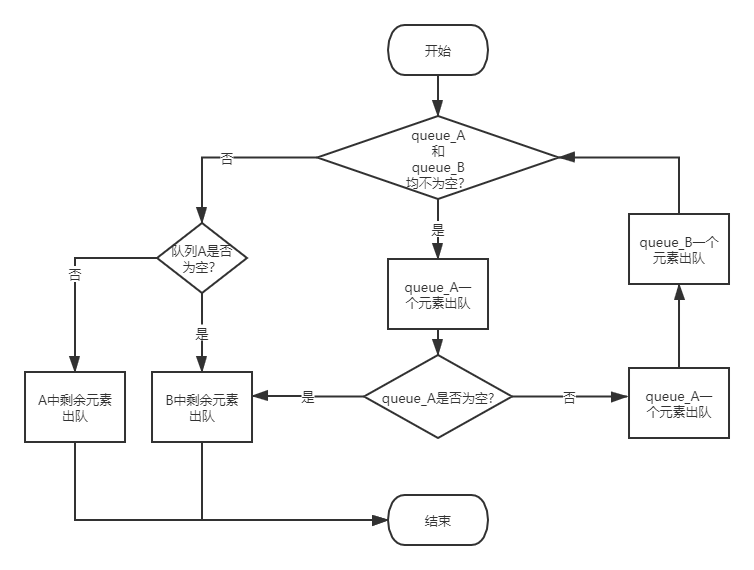
queue\_A.outQueue(temp);

queue\_result.inQueue(temp);

}

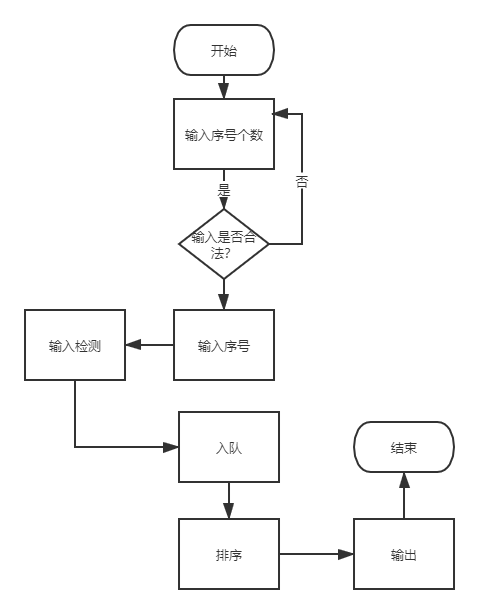
}

3.2.3 流程图表示

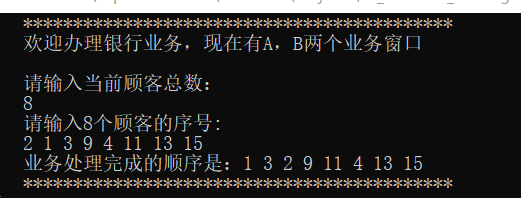


3.3 总操作

3.3.1 总操作流程图



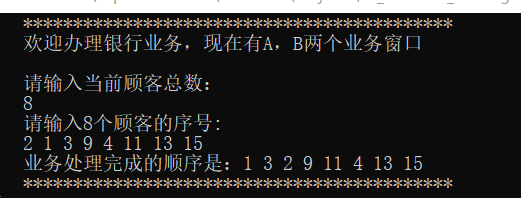
3.3.2 整体操作截图

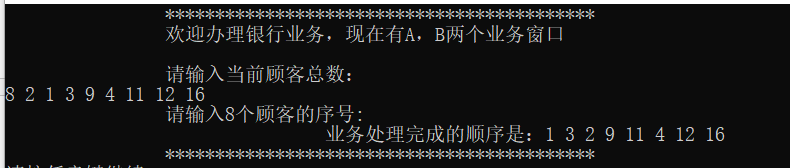


**4.测试**

**4.1功能测试**

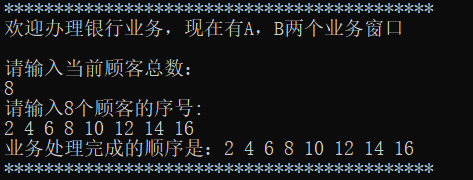
正常功能测试

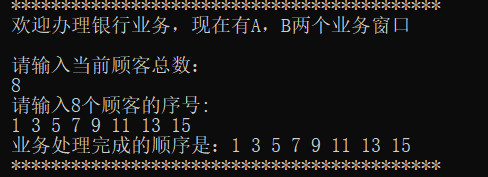




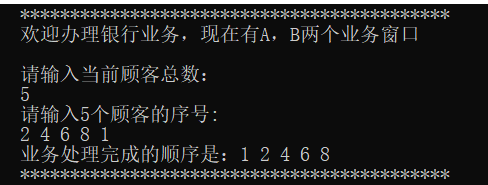
**4.2边界测试**

**4.2.1 其中一个队列为空**



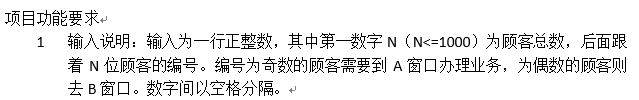


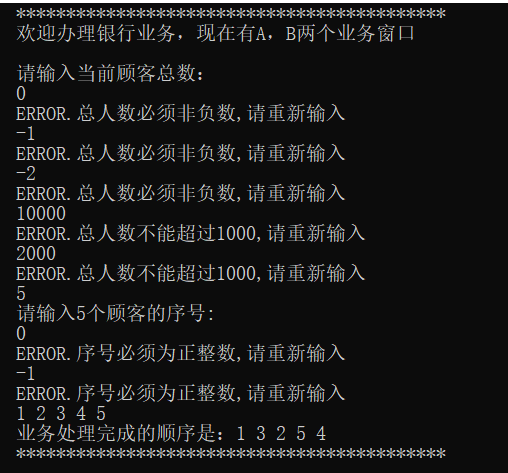
**4.2.2 队列A只有一个元素（无法连续执行两次出队操作）**



**4.2.3 输入的数据在边界之外**

注意：题目中明确强调输入一行正整数，其中第一个数字N（N<=1000），所以在这个范围之外的数据都应当被认为是越界的不合法输入。



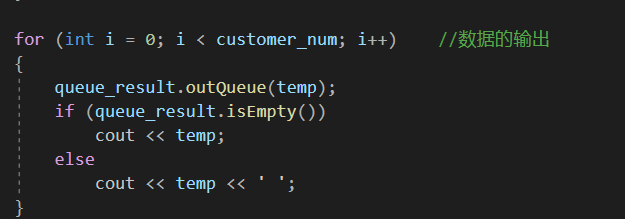


**4.2.4 输出数据**

由于题目中明确要求输出时最后一个元素后不能有空格，所以对于输出需要做额外的检测

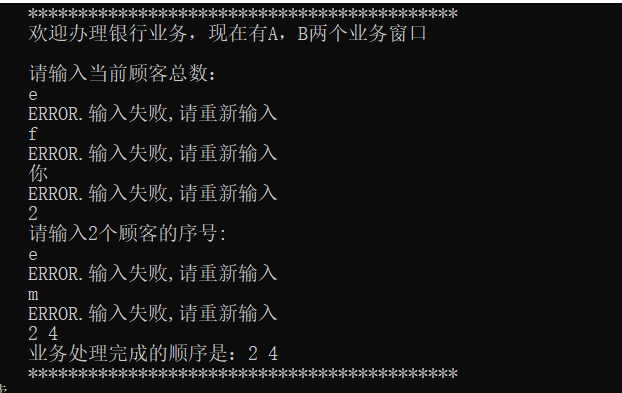


解决方案如下：



**4.3错误测试**

在整个项目中，所有的输入数据都应该是正整数，所以若输入诸如字符之类的数据，系统会给出一定的提示



对应的代码为：

while (true)

{

cin >> customer\_num;

if (cin.fail())

{

cerr << "\t\tERROR.输入失败,请重新输入" << endl<<"\t\t";

cin.clear();

cin.ignore();

cin.get();

}

else if (customer\_num <= 0)

{

cerr << "\t\tERROR.总人数必须非负数,请重新输入" << endl << "\t\t";

}

else if (customer\_num > 1000)

{

cerr << "\t\tERROR.总人数不能超过1000,请重新输入" << endl << "\t\t";

}

else

break;

}