项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 杨煜

学 号： 1850217

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

[1 分析 1](#_Toc26872108)

[1.1 项目内容 1](#_Toc26872109)

[1.2 项目要求 1](#_Toc26872110)

[2 设计 1](#_Toc26872111)

[2.1 数据结构设计 1](#_Toc26872112)

[2.2 队列结构设计 2](#_Toc26872113)

[2.3 成员与操作设计 2](#_Toc26872114)

[3 实现 2](#_Toc26872115)

[3.1 构造函数的实现 2](#_Toc26872116)

[3.1.1 构造函数 2](#_Toc26872117)

[3.1.2 构造函数核心代码 2](#_Toc26872118)

[3.2 析构函数的实现 3](#_Toc26872119)

[3.2.1 析构函数 3](#_Toc26872120)

[3.2.2 析构函数核心代码 3](#_Toc26872121)

[3.3 入队操作的实现 3](#_Toc26872122)

[3.3.1 入队操作 3](#_Toc26872123)

[3.3.2 入队操作核心代码 3](#_Toc26872124)

[3.4 出队操作的实现 3](#_Toc26872125)

[3.4.1 出队操作 3](#_Toc26872126)

[3.4.2 出队操作核心代码 4](#_Toc26872127)

[3.5 总体流程的实现 4](#_Toc26872128)

[3.5.1 总体流程 4](#_Toc26872129)

[3.5.2 总体系统核心代码 4](#_Toc26872130)

[3.5.3 总体系统流程图 6](#_Toc26872131)

[4 测试 6](#_Toc26872132)

[4.1 功能测试 6](#_Toc26872133)

[4.1.1 正常测试，A窗口人多 6](#_Toc26872134)

[4.1.2 正常测试，B窗口人多 7](#_Toc26872135)

[4.1.3 最小N 8](#_Toc26872136)

[4.2 出错测试 9](#_Toc26872137)

[4.2.1 输入数字非法 9](#_Toc26872138)

# 1 分析

## 1.1 项目内容

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。

## 项目要求

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

如上述项目要求所述，该程序需要按照银行的业务顺序输出顾客的序列。典型的先进先出的输入输出受限制的线性表。因而可以考虑使用使用队列这种数据结构来模拟。本程序需要使用两个队列，A出队两个，B出队一个。

## 2.2 队列结构设计

使用数组作为队列的存储结构，使用一个头指针，一个尾指针做出控制。ben程序使用了循环队列来完成队列的实现。

## 2.3 成员与操作设计

**队列类（SeqQueue）**

class SeqQueue

{

public:

SeqQueue(int sz = 1000);//构造函数

~SeqQueue() { delete[] elements; };//析构函数，释放空间

bool EnQueue(const int& x);//入队操作

int DeQueue();//出队操作

bool isEmpty()const { return(front == rear) ? true : false; };//判断队列是否为空

bool isFull()const { return((rear + 1) % maxSize == front) ? true : false; };//判断队列是否为满

private:

int rear, front;//头指针，尾指针

int\* elements;

int maxSize;//最大长度

};

# 3 实现

## 3.1 构造函数的实现

### 3.1.1 构造函数

构造函数开辟了一个默认长度为100的数组空间，以存放队列数据，其余头尾指针默认为零。

### 3.1.2 构造函数核心代码

SeqQueue::SeqQueue(int sz) :front(0), rear(0), maxSize(sz)

{

elements = new int[maxSize];

}

## 3.2 析构函数的实现

### 3.2.1 析构函数

析构函数直接释放节点空间。

### 3.2.2 析构函数核心代码

~SeqQueue() { delete[] elements; };

## 3.3 入队操作的实现

### 3.3.1 入队操作

入队函数接受的参数为要入队的元素。判断队列是否已满，如果已满，则插入失败。否则队尾元素置为参数，队尾加一模最大元素个数。

### 3.3.2 入队操作核心代码

bool SeqQueue::EnQueue(const int& x)

{

if (isFull() == true)//判满

return false;

elements[rear] = x;

rear = (rear + 1) % maxSize;

return true;

}

## 3.4 出队操作的实现

### 3.4.1 出队操作

出队函数不接受任何参数。判断队列是否已空，如果已空，则删除失败。返回函数为头节点，头节点加1模最大元素个数。

### 3.4.2 出队操作核心代码

int SeqQueue::DeQueue()

{

int x;

if (isEmpty() == true)

return false;

x = elements[front];

front = (front + 1) % maxSize;

return x;

}

## 3.5 总体流程的实现

### 3.5.1 总体流程

首先申明两个队列a，b。输入银行业务的顾客数，按奇偶分别进入两个栈之中。然后，按照a出栈两个，b出栈一个的顺序，出栈。当某个栈为空时，输出另一个栈的内容。

### 3.5.2 总体系统核心代码

int main()

{

int n, k,x;

SeqQueue a, b;

cin >> n;

for (int i = 0; i < n; i++)

{

cin >> k;

if (k % 2 == 1)//输入

{

a.EnQueue(k);

}

else

{

b.EnQueue(k);

}

}

while (!a.isEmpty() && !b.isEmpty())//出队

{

x = a.DeQueue();

if (x)

{

cout << x << " ";

}

x = a.DeQueue();

if (x)

{

cout << x << " ";

}

x = b.DeQueue();

if (x)

{

cout << x << " ";

}

}

while (!a.isEmpty())

{

cout << a.DeQueue() << " ";

}

while (!b.isEmpty())

{

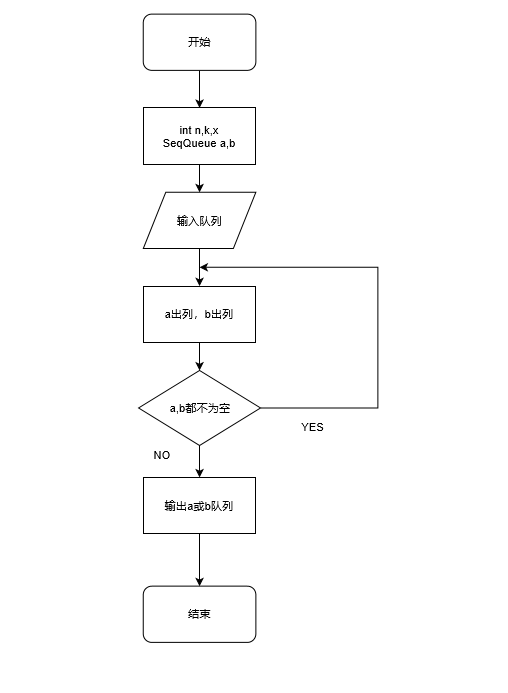
cout << b.DeQueue() << " ";

}

return 0;

}

### 3.5.3 总体系统流程图



# 4 测试

## 4.1 功能测试

### 4.1.1 正常测试，A窗口人多

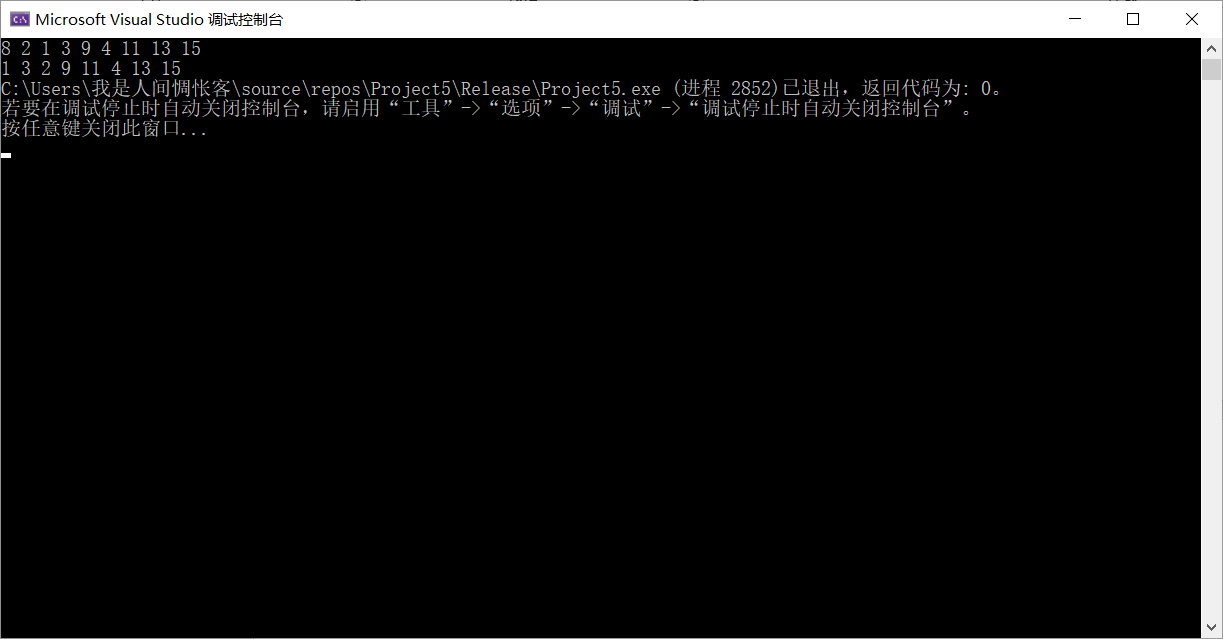
**测试用例**：

8 2 1 3 9 4 11 13 15

**预期结果**：

1 3 2 9 11 4 13 15

**实验结果**



### 4.1.2 正常测试，B窗口人多

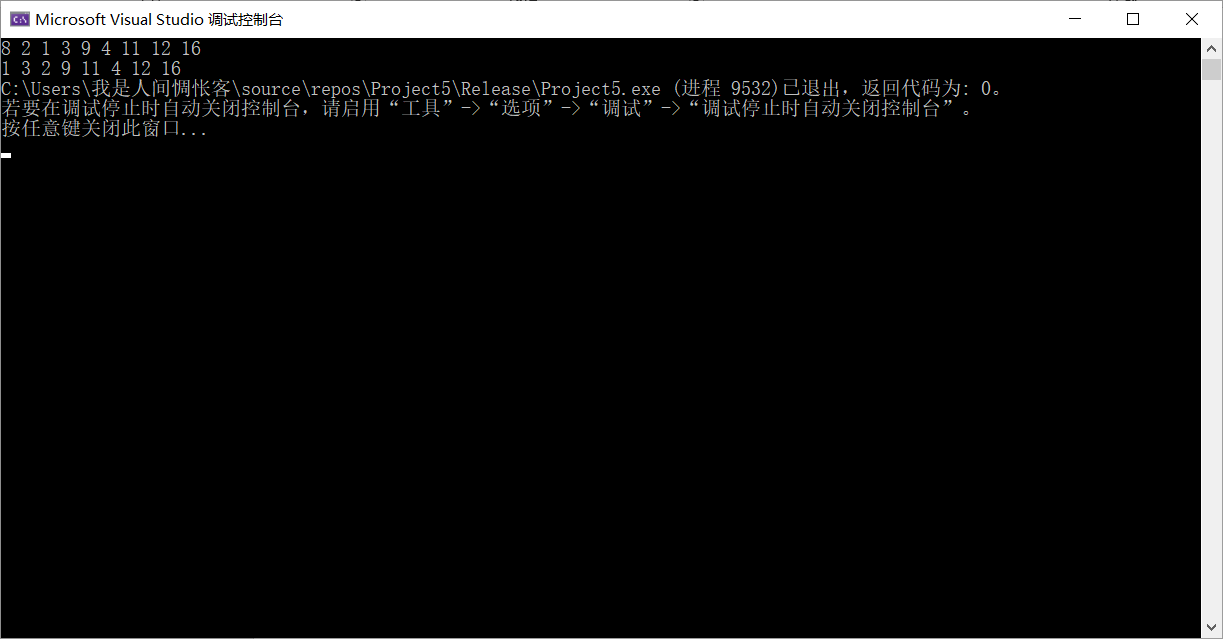
**测试用例：**

8 2 1 3 9 4 11 12 16

**预期结果：**

**1 3 2 9 11 4 12 16**

**实验结果：**



### 4.1.3 最小N

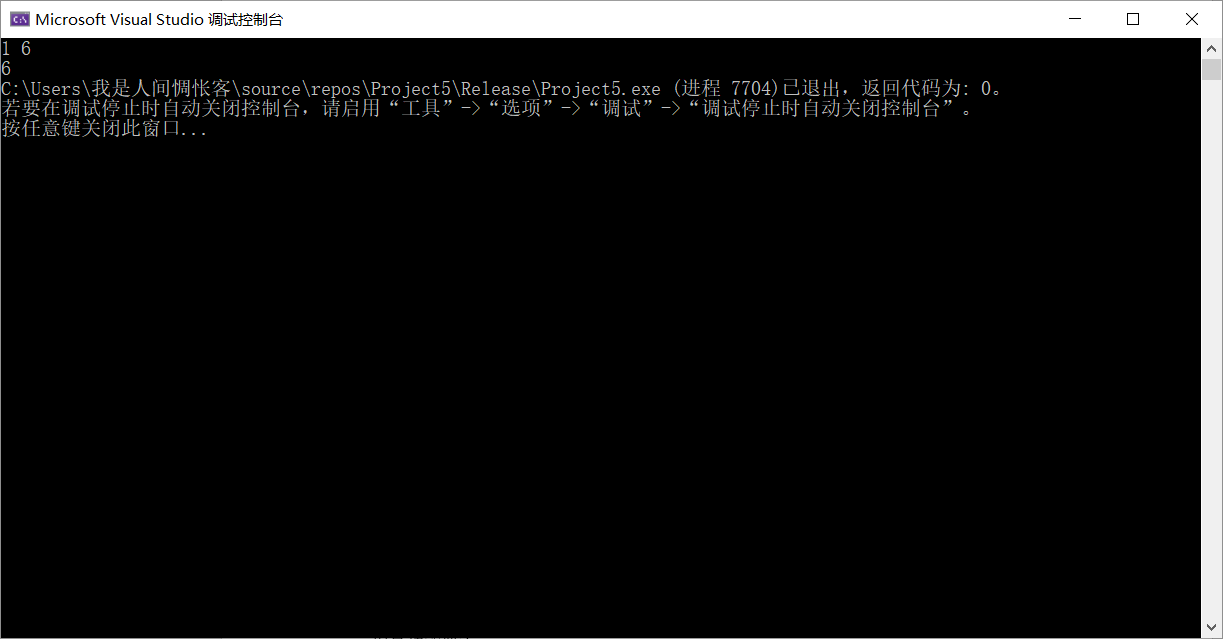
**测试用例：**

1 6

**预期结果：**

6

**实验结果：**



## 4.2 出错测试

### 4.2.1 输入数字非法

**测试用例：**

c 6

**预期结果：**

程序不崩溃

**实验结果：**

