项目说明文档

数据结构课程设计

——银行业务

作 者 姓 名： 崔鑫宇

学 号： 1853444

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

某银行中有A、B两个业务窗口，且处理业务的速度也不一样，其中A窗口处理速度是B窗口的两倍——即当A窗口每处理完2个顾客是，B窗口处理完1个顾客，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。现要按照业务处理完成的顺序输出顾客的编号。

## 1.2 功能分析

从文字中的意思中可以读出本程序要求可以从列数字中提取出每个元素，并将顾客以编号的奇偶分别分往A、B窗口，然后按一定顺序与规则依次对每个队列执行输出操作。

综上所述，一个处理银行业务的程序至少应该具有输入、提取、输出的功能。

# 2 设计

## 2.1 数据结构设计

如上功能分析所述，由于本程序要求从由用户输入的序列中提取出一组数据，并以奇偶性为依据将其中的奇数与偶数分别置入A、B两组中。并按照先后顺序依次从A中提取两个数据再从B中提取一个数据。

综上所述，本程序比较适合用队列这一数据结构完成

## 

## 2.2 类结构设计

本程序主要的实现工具是队列(Queue)类，实现方法是先将数据分别放置于A、B两个队列中，在以A12、B1、A12、B1……的顺序从A、B两队列中弹出当时最靠前的一个或两个元素。

## 

## 2.3 成员与操作设计

**队列类（Queue）**

public:

Queue();

Queue(int n);

~Queue();

//弹出与提取元素

T pop();

T get();

void push(T data);//向队列尾后添加元素

void print();//打印队列中元素

int size = 0;//队列当前大小

int head=0;//头元素位置

int max\_size=0;//队列最大长度

//判断队列是否为满或是否为空

bool isEmpty();

bool isFull();

private:

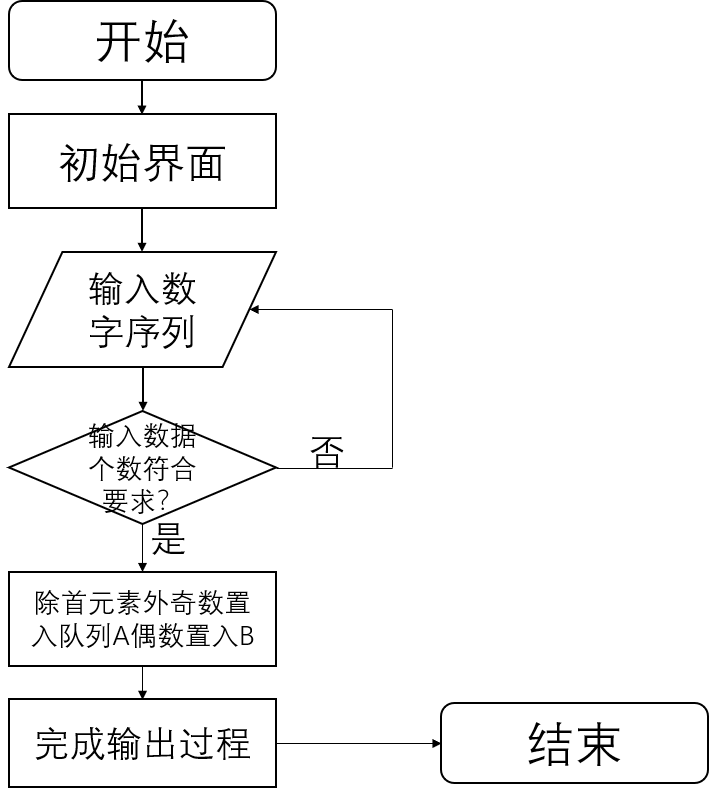
T\* queue;

## 2.4 系统设计

系统直接调用BuildQueue()函数提示用户进行输入工作，并规范用户的输入，然后根据用户的输入进行进一步操作。

# 3 实现

### 3.1 总体系统流程图



### 

### 3.2 总体系统核心代码

cout << "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。" << endl;

string str;

getline(cin, str);

int sum = 0;

int n = 0;

int num\_of\_odd = 0, num\_of\_even = 0;

//提取出输入序列中的第一个数字（顾客个数）

while (isNum(str[n])) {

sum = (int)(str[n] - '0') + 10 \* sum;

n++;

}

//奇数队列、偶数队列以及最终用于输出的队列

Queue<int> A, B, Out;

A.max\_size = sum;

B.max\_size = sum;

Out.max\_size = sum;

sum = 0;

for (unsigned int i = n + 1; i < str.size(); i++) {

if (isNum(str[i])) {

sum = (int)(str[i] - '0') + 10 \* sum;

if (str[i + 1] && isNum(str[i + 1])) {

continue;

}

else {

if (isOdd(sum)) {//将奇数置入奇数队列中并记录个数

A.push(sum);

num\_of\_odd++;//记录奇数个数

sum = 0;

}

else if (isEven(sum)) {//将偶数置入偶数队列中并记录个数

B.push(sum);

num\_of\_even++;//记录偶数个数

sum = 0;

}

}

}

}

//重置A、B队列的元素个数

A.max\_size = num\_of\_odd;

B.max\_size = num\_of\_even;

int Remain\_odd = num\_of\_odd, Remain\_even = num\_of\_even;

while (Remain\_odd > 1 && Remain\_even > 0) {//每次奇数队列先输出两个元素偶数队列再输出一个元素

Out.push(A.get());

Out.push(A.get());

Out.push(B.get());

Remain\_odd -= 2;

Remain\_even--;

}

//考虑奇数剩余不足2个或偶数剩余不足1个的情况

if (Remain\_even == 1 && Remain\_odd == 1) {//奇偶各剩一个元素

Out.push(A.get());//奇数偶数剩余的一个元素先后输出

Out.push(B.get());

}

else if (Remain\_even == 0 && Remain\_odd != 0) {//奇数剩余个数为0但偶数剩余个数不等于0

while (!A.isEmpty()) {//输出偶数队列内剩余所有元素

Out.push(A.get());

}

}

else if (Remain\_even != 0 && Remain\_odd == 0) {//偶数剩余个数为0但奇数剩余个数不等于0

while (!B.isEmpty()) {//输出奇数队列内剩余所有元素

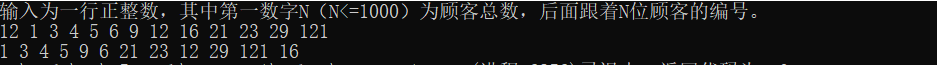
Out.push(B.get());

}

}

Out.print();

### 3.3 总体系统截屏示例



# 4 测试

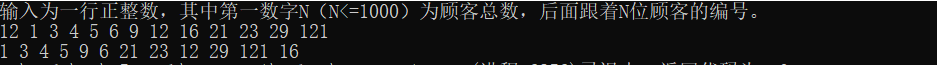
## 4.1 正常功能测试

4.1.1 奇数偶数个数比等于2比1

测试用例：12 1 3 4 5 6 9 12 16 21 23 29 121

预测结果：输出1 3 4 5 9 6 21 23 12 29 121 16

实验结果：

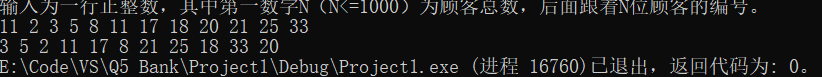


4.1.2 奇数偶数个数比等于2n+1比n+1(n为非负整数)

测试用例：11 2 3 5 8 11 17 18 20 21 25 33

预测结果：3 5 2 11 17 8 21 25 18 33 20

实验结果：



4.1.3 奇数偶数个数比等于2n比n+m(n、m为非负整数且m>0)

测试用例：12 2 4 5 9 14 18 21 23 24 26 38 42

预测结果：5 9 2 21 23 4 14 18 24 26 38 42

实验结果：

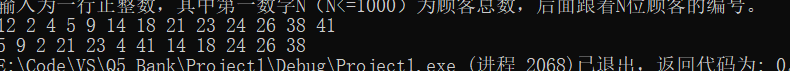


4.1.3 奇数偶数个数比等于2n+1比n+m(n、m为非负整数且m>1)

测试用例：12 2 4 5 9 14 18 21 23 24 26 38 41

预测结果：5 9 2 21 23 4 41 14 18 24 26 38

实验结果：

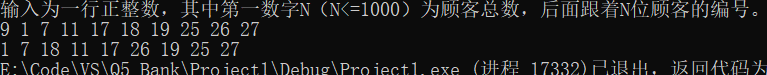


4.1.4 奇数偶数个数比等于2n+m比n(n、m为非负整数且m>0)

测试用例：9 1 7 11 17 18 19 25 26 27

预测结果：1 7 18 11 17 26 19 25 27

实验结果：



## 

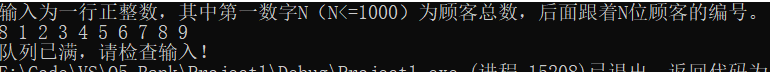
## 4.2 错误输入测试

# 4.2.1 输入数据多于用户自己指定的个数

测试用例：8 1 2 3 4 5 6 7 8 9

预测结果：警告用户队列已满，并要求用户重新检查输入数据

实验结果：



4.2.2 输入数据少于用户自己指定的个数

测试用例：8 1 2 3 4 5 6 7

预测结果：警告用户队列没有填满，并要求用户重新检查输入数据

实验结果：

