项目说明文档

数据结构课程设计

——**银行业务**

作 者 姓 名： 肖杨

学 号： 1950430

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 3

2 设计 3

3 实现 7

4 测试 11

1 分析

1.1背景分析

项目的实际场景是银行处理业务。现在有A窗口和B窗口，A窗口和B窗口处理业务的速度不一样，每当A处理完2个顾客时，B处理完1个顾客。

1.2 功能分析

在项目的背景下，需要实现的功能是给定到达银行的顾客序列，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.按照业务完成的顺序输出顾客序列。并且不考虑顾客先后到达的时间间隔，当A，B同时处理完顾客时，A窗口的顾客优先输出。

2 设计

2.1 数据结构设计

如上功能分析所述，项目需要存储顾客的处理顺序，排在前面的顾客先进行业务服务，满足先进先出原则，所以需要数据结构队列。 这里队列采用连续地址存储，虽然使用了循环队列，但为了保证不出现溢出情况，未使用到循环功能。

2.2 类结构设计

为了保证设计的数据结构的泛用性，本项目选择将queue类设计为模板类， queue类储存队首和队尾，保存内部数据，并提供入队、出队等接口。最后的处理顺序不进行储存，直接进行输出。

同时设计了BankService类，用于进行输入、输出操作并储存相关元素。

本程序支持小于INT\_MAX的正整数序列的顾客序号，并有输入检测。

2.3 成员与操作设计

队列类

类定义

template<class T>

class queue

{

private:

int size;

int number;

T\* buffer;

int head;

int tail;

public:

queue(int num);

~queue();

void push(const T& elem);

T pop();

int getSize();

bool empty();

bool full();

};

私有成员：

int size;

//队列大小

int number;

//当前已储存的数目

T\* buffer;

//指向储存元素地址的指针

int head;

//队列头部

int tail;

//队列尾部

公有操作：

queue(int num);

//用队列大小初始化

void push(const T& elem);

//入队函数

T pop();

//出队函数，并返回出队元素

int getSize();

//返回队列大小

bool empty();

//队列判空

bool full();

//队列判满

银行服务类

类定义

class bankService

{

private:

queue<int> odd;

queue<int> even;

int size;

public:

bankService(int number);

void serve();

void pop(bool isEven);

};

私有成员：

queue<int> odd;

//偶数队列

queue<int> even;

//奇数队列

int size;

//总人数

公有操作：

bankService(int number);

//用人数进行初始化的构造函数

void serve();

//模拟服务并输出

void pop(bool isEven);

//用于输出的函数

2.4 系统设计

系统首先询问有多少顾客，然后让用户输入顾客到达序列，在求出顾客处理顺序之后将其输出。

3 实现

3.1 顾客分离功能的实现

3.1.1 顾客分离流程

在输入顾客序列的过程中，判断每个顾客的序号，序号为偶数的入队even，序号为奇数的入队odd，直到所有顾客都入队。输入过程中需要对输入进行判断，判断是否为数字。

3.1.2 核心代码

cout << "请输入排队的序号" << endl;

int temp = 0;

string temps;

bool isNumber = true;

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

{

cin >> temps;

isNumber = true;

for (auto elem : temps)

{

if (!(elem >= '0'&&elem <= '9'))

{

cout << "请输入正整数" << endl;

i--;

isNumber = false;

break;

}

}

if (!isNumber)

{

continue;

}

if (atoi(temps.c\_str()) < INT\_MAX)

{

temp = stoi(temps);

}

else

{

i--;

cout << "请输入小于INT\_MAX的整数" << endl;

continue;

}

if (temp % 2 == 1)

{

odd.push(temp);

}

else

{

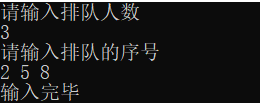
even.push(temp);

}

}

cout << "输入完毕" << endl;

3.1.3 输入功能演示



3.2 排序功能实现

3.2.1 排序功能流程

排序时直接判断队列是否为空，若非空，则按照奇数列输出两个，偶数列输出一个的顺序进行输出，输出至两个队列均为空时停止。

输出时，最后一个序号后无空格，符合题目要求。

3.2.2 核心代码

void bankService::serve()

{

cout << "服务完毕的顺序为:" << endl;

while (!odd.empty() || !even.empty())

{

pop(false);

pop(false);

pop(true);

}

}

void bankService::pop(bool isEven)

{

if (isEven)

{

if (!even.empty())

{

cout << even.pop();

size--;

}

else

{

return;

}

}

else

{

if (!odd.empty())

{

cout << odd.pop();

size--;

}

else

{

return;

}

}

if (size > 0)

{

cout << ' ';

}

}

3.2.3 排序功能演示



3.3 总体系统实现

3.3.1 工作流程

首先输入顾客总数，之后进行顾客序号按奇偶分离，随后进行模拟业务处理并输出。对于顾客总数进行了输入检测。

3.3.2 核心代码

int main()

{

cout << "请输入排队人数" << endl;

int number = -1;

cin >> number;

if (number < 0)

{

cout << "请输入合法的人数" << endl;

system("pause");

return 0;

}

else if (number == 0)

{

cout << "无人排队" << endl;

system("pause");

return 0;

}

bankService bank(number);

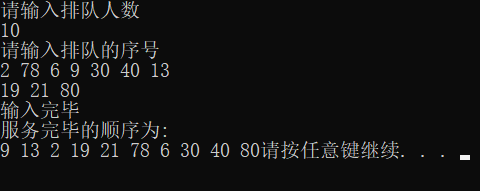
bank.serve();

system("pause");

return 0;

}

3.3.3 总体演示



4 测试

4.1 功能测试

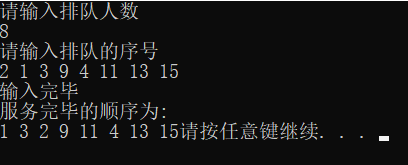
4.1.1 A窗口顾客较多

**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：**



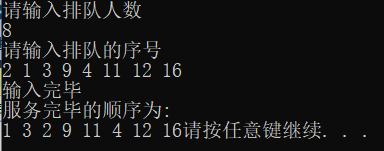
4.1.2 B窗口顾客较多

**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：**



4.2 边界测试

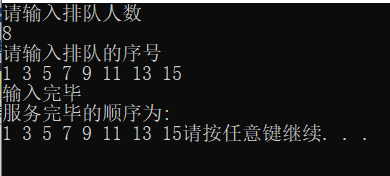
4.2.1 只有A窗口有人

**测试用例：**8 1 3 5 7 9 11 13 15

**预期结果：**

1 3 5 7 9 11 13 15

**实验结果：**



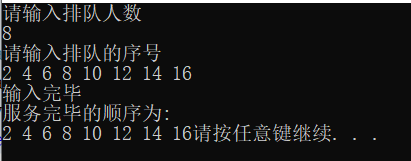
4.2.2 只有B窗口有人

**测试用例：**8 2 4 6 8 10 12 14 16

**预期结果：**

2 4 6 8 10 12 14 16

**实验结果：**



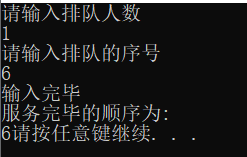
4.2.3 最小N

**测试用例：**1 6

**预期结果：**

6

**实验结果：**



4.3 错误测试

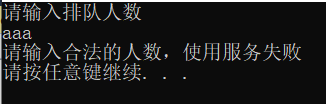
4.3.1 输入人数非正整数

**测试用例：**aaa

**预期结果：**

输出错误信息并结束程序。

**实验结果：**



4.3.2 输入序列中含有非正整数

**测试用例：**8 a 1 2 3 4 5 6 -1 9 10

**预期结果：**

跳过非法输入，程序继续进行。

**实验结果：**

