****

项目说明文档

**数据结构课程设计**

**——银行业务**

**培养单位：软件学院**

**本 科 生：蓝 笙 聆**

**学 号：1951096**

**指导老师：张 颖**

二○二○年十二月

目录

[第1章 功能分析 1](#_Toc58802917)

[第2章 设计 2](#_Toc58802918)

[2.1 数据结构设计 2](#_Toc58802919)

[2.2 类结构设计 2](#_Toc58802920)

[2.3 成员与操作设计 2](#_Toc58802921)

[2.4 系统设计 3](#_Toc58802922)

[第3章 实现 4](#_Toc58802923)

[3.1 优先级队列插入的实现 4](#_Toc58802924)

[3.1.1 优先级队列插入功能流程图 4](#_Toc58802925)

[3.1.2 优先级队列插入功能核心代码 5](#_Toc58802926)

[3.2 总体系统的实现 6](#_Toc58802927)

[3.2.1 总体系统流程图 6](#_Toc58802928)

[3.2.2 总体系统代码实现 6](#_Toc58802929)

[3.2.3 总体功能截屏示例 8](#_Toc58802930)

[第4章 测试 9](#_Toc58802931)

[4.1 通常功能测试 9](#_Toc58802932)

[4.2 边界测试 9](#_Toc58802933)

[4.2.1 存在一个队列为空 9](#_Toc58802934)

[4.2.2 输入为空 10](#_Toc58802935)

[4.3 出错测试 10](#_Toc58802936)

[4.3.1 输入非数值字符 10](#_Toc58802937)

[4.3.1 输入负数 10](#_Toc58802938)

# 第1章 功能分析

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。

项目功能要求：

输入说明：输入为一行正整数，其中第一个数字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。

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。

# 第2章 设计

## 2.1 数据结构设计

如上所述，需要完成按顺序出队，则考虑使用优先级队列。

## 2.2 类结构设计

注意到链表类可以直接实现优先级队列类的大部分功能，直接通过继承链表类完成优先级队列类。

## 2.3 成员与操作设计

**模板链表节点结构体（Node）**

template <class T>

struct Node {

    T \_data;      //数据

    Node \*\_next;  //指针域

    Node();

};

**模板链表类（Node）**

template <class T>

class List {

   private:

    Node<T> \*\_head;  //表头结点

    int \_len;        //链表长度

   public:

    void ListIns(int n, T data);  //将数据插入第n个节点中

    void ListDel(int n);          //删除第n个节点

    T ListPop(T data);    //查找链表中的数据返回并删除节点

    int ListFind(T id);   //查找数据在链表中的位置

    int ListLen() const;  //返回链表长度

    bool Empty() const;   //返回是否为空

    void ListChange(int n, T data);  //修改链表第n个节点的值

    Node<T> \*GetHead() const { return this->\_head; }  //返回头结点

    std::string ListPrint(int n) const;               //将链表输出

    List();

    ~List();

};

**模板优先级队列类（Priorityqueue）**

template <class T>

class Priorityqueue : public List<T> {

   private:

   public:

    void Push(T val);  //插入元素并按升序排列

    void Pop();        //移除队首

    T Top();           //返回队首

};

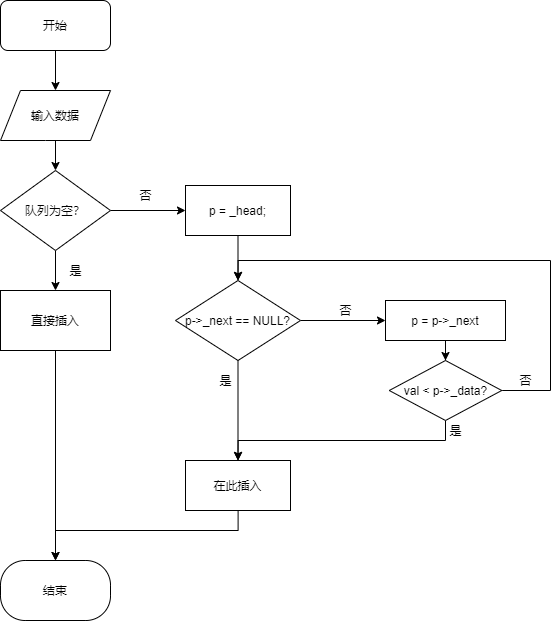
## 2.4 系统设计

系统首先调用opening ()函数实现对屏幕的初始化，然后直接在main函数中输入数据并返回结果。

# 第3章 实现

## 3.1 优先级队列插入的实现

### 3.1.1 优先级队列插入功能流程图



### 3.1.2 优先级队列插入功能核心代码

template <class T>

void Priorityqueue<T>::Push(T val) {

    if (this->Empty()) {

        this->ListIns(0, val);

        return;

    }

    int i = 0;

    auto p = this->GetHead();

    while (p->\_next != nullptr) {

        p = p->\_next;

        if (val < p->\_data) {

            this->ListIns(i, val);

            return;

        }

        i++;

    }

    this->ListIns(i, val);

}

## 3.2 总体系统的实现

### 3.2.1 总体系统流程图



### 3.2.2 总体系统代码实现

void opening() {

    cout << "Banking Business" << endl

         << "The input is a line of positive integers. " << endl

         << "The first number is N followed by the number of N customers. "

         << endl

         << "Now enter a line of positive integers. " << endl;

    return;

}

void input(Priorityqueue<int> &a, Priorityqueue<int> &b) {

    string temps;

    int num = 0;

    cin >> temps;

    if (isdigit(temps[0]))

        num = temps[0] - '0';

    else

        throw "Error: Invalid input detected. Please try again.";

    for (int i = 0; i < num; i++) {

        int temp;

        cin >> temps;

        if (isdigit(temps[0]))

            temp = temps[0] - '0';

        else

            throw "Error: Invalid input detected. Please try again.";

        if (temp % 2 == 1)

            a.Push(temp);

        else

            b.Push(temp);

    }

}

bool loop(){

    opening();

    Priorityqueue<int> a, b;

    while (true) {

        try {

            input(a, b);

            break;

        } catch (const char \*e) {

            std::cerr << e << '\n';

        }

    }

    while (!a.Empty() || !b.Empty()) {

        if (!a.Empty()) cout << a.Top() << ' ', a.Pop();

        if (!a.Empty()) cout << a.Top() << ' ', a.Pop();

        if (!b.Empty()) cout << b.Top() << ' ', b.Pop();

    }

    cout << endl;

    std::cout << "Again? (y for yes, n for no)";

    char c;

    std::cin >> c;

    if (c == 'y')

        return true;

    else

        return false;

}

int main() {

    while (loop())

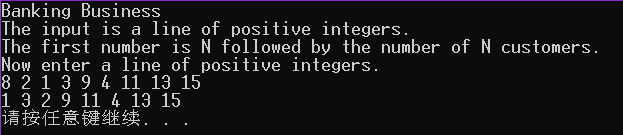
        ;

    std::cout << "Thanks for using it. See you next time! " << std::endl;

    return 0;

}

### 3.2.3 总体功能截屏示例



# 第4章 测试

## 4.1 通常功能测试

**测试用例**：

8 2 1 3 9 4 11 13 15

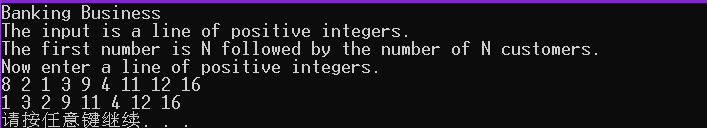
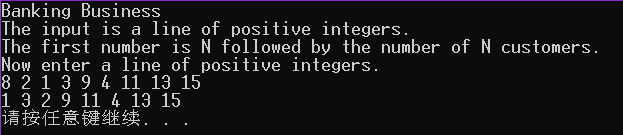
8 2 1 3 9 4 11 12 16

**预期结果**：

1 3 2 9 11 4 13 15

1 3 2 9 11 4 12 16

**实验结果**



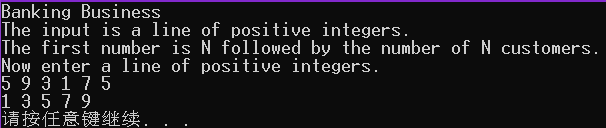
## 4.2 边界测试

### 4.2.1 存在一个队列为空

**测试用例：**5 9 3 1 7 5

**预期结果：**1 3 5 7 9

**实验结果：**

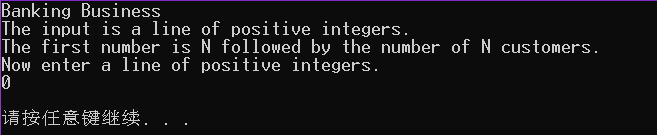


### 4.2.2 输入为空

**测试用例：**0

**预期结果：**（无输出，因为无人在排队）

**实验结果：**



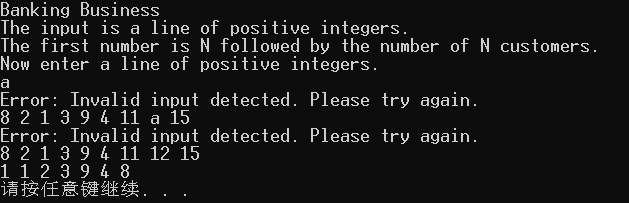
## 4.3 出错测试

### 4.3.1 输入非数值字符

**测试用例：**输入非数值，如a，8 2 1 3 9 4 11 a 15。

**预期结果：**抛出异常，程序要求重新输入，不崩溃。

**实验结果：**



### 4.3.1 输入负数

**测试用例：**输入负数，如-1，8 2 1 3 9 4 -1 12 15。

**预期结果：**抛出异常，程序要求重新输入，不崩溃。

**实验结果：**

