主函数：

Main

Bank类的构造函数

设置while（输入业务类型：0为退出来结束循环）

用switch来选择业务

（有必要可将上面封装成一个类，就是菜单类）

## 次类：

Window类：

属性:

开始营业时间

结束营业时间

开始业务时间（手动输入初始化）

结束业务时间（手动输入初始化）

窗口状态：true or false

四个业务所需要的时间(构造函数函数初始化，)

。。。拓展

方法：

构造方法

{

输入4个业务分别需要的时间；

输入开始营业时间；

输入结束营业时间；

开始业务时间 = 开始营业时间；

结束业务时间 = 0；

窗口状态 = true；

}

开始营业方法（队的对象）

{

1 while(判断队链是否为空)

1.1判断队链当前节点要处理的业务是哪一个(代号就是0123)

1.2进入相应的if块中

2.1

}

Struct Node

{

Cilent类对象（数据域）

Node \*next;(指针域)

}

BusinessWidow顺序队类：

属性：

Node \*front;

Node \*rear;

方法：

构造方法

入队

出队

判空

。。。拓展

Client类：

属性：

时间（需记录控制在营业范围内）

编号（不需记录）

需要办理的业务（需记录）

方法：

依据最初Bank中的营业时间随机生成

用户的到来时间，需要办理的业务，人员编号（每次从0开始，++）。

银行提供时间 -> 通过时间打票 -> 将票类对象作为客户类的属性，每个客户都有唯一的票号 -> 将客户类对象作为属性传入 ->

对接部分的逻辑：

设置每个窗口结束时间。

第一趟：结束时间 = 开始时间+工作时间。

循环下部分操作。

找到此时最小结束时间的窗口，他的开始时间变为当前结束的时间

将该窗口empty设置为false

队列出队，将这个出队元素的业务时间与当前窗口的开始时间相加为新的结束时间

测试队列：8 2 1 4 8 2 4 8

队列剩余： 8 2 4 8

开始时间：

100

100

100

100

结束时间：

108

102

101

104

### void startWorking(LinkedQueue client\_queue, Window \*window, int window\_length)

### {

**Int i**；

**If(判断队链是否为空)**

**{**

**退出函数。**

**}else**

**{**

**While(队首不等于等于队尾)**

**{**

**执行出队操作**

**出队操作完成之后安排到先遍历四个窗口是否为空**

**将该客户对接到该窗口**

**While(window[i].** **getWindowsEmpty() != true)//遍历查找空窗口**

**{**

**i ++;**

**}**

**i = 0;//给i归位以便下次遍历空窗口**

**for (i=0; i<window\_length; i++)**

**{**

**If (window[i].** **getWindowsEmpty() == true)**

**{**

**Switch (当前客户的业务类型)**

**{**

**Case 业务类型1:…**

**分别执行用该业务的时间（通过Windowd数组中**

**当前对接客户的窗口对象调用）+上该窗口开始办理业务时间。**

**Window[i]**

**Case 业务类型2:**

**分别执行用该业务的时间（通过Windowd数组中**

**当前对接客户的窗口对象调用）+上该窗口开始办理业务时间。**

**Case 业务类型3:**

**分别执行用该业务的时间（通过Windowd数组中**

**当前对接客户的窗口对象调用）+上该窗口开始办理业务时间。**

**Case 业务类型4:**

**分别执行用该业务的时间（通过Window数组中**

**当前对接客户的窗口对象调用）+上该窗口开始办理业务时间。**

**Default：…**

**}**

**}**

**}**

**}**

**}**

### }