实验 2-4: 银行业务模拟

1. 问题描述

客户业务分为两种：

第一种是申请从银行得到一笔资金，即取款或借款；

第二种是向银行投入一笔资金，即存款或还款。

银行有两个服务窗口，相应地有两个队列。客户到达银行后先排第一个队，处理每个客户业务时，如果属于第一种，且申请额超出银行现存资金总额而得不到满足，则立刻排入第二个队等候直至满足时才离开银行;否则业务处理完后立刻离开银行,每接待完一个第二种业务的客户，则顺序检查和处理(如果可能)第二个队列中的客户，对能满足的申请者予以满足,不能满足者重新排到第二个队列的队尾。注意，在此检查过程中，一旦银行资金总额少于或等于刚才第一个队列中最后一个客户(第二种业务)被接待之前的数额，或者本次已将第二个队列检查或处理了一遍，就停止检查(因为此时已不可能还有能满足者)转而继续接待第一个队列的客户。任何时刻都只开一个窗口。假设检查不需要时间，营业时间结束时所有客户立即离开银行。

写一个上述银行业务的事件驱动模拟系统，通过模拟方法求出客户在银行内逗留的平均时间。

输入要求: 第一行输入三个数N、total、close_time、average_time，分别表示来银行的总人数、银行开始营业时拥有的款额、今天预计的营业时长和客户交易时长 之后的N行每行输入两个数a、b，第一个数a为客户办理的款额，用负值和正值分别表示第一类和第二类业务。第二个数b为客户来到银行的时间

输出要求: 前N行输出分别对应每个客户的等候时长 第N+1行输出为客户等候平均时长，结果去尾法保留整数即可

2. 算法的描述

2.1 数据结构的描述

本程序主要的数据结构是 队列(queue) ， 栈的实现和函数操作分别被定义在 `queue.c queue.h` 中，主函数在 `main.c` 中。

对来银行的每个人，都定义了一个结构体 `aPerson`, 定义在 `queue.h` 中，对每个人，定义他们进入银行的时间 `time`, 通过减法得出他们的总等待时间。`main` 函数中，存在两个队列 `firstQueue` 和 `secondQueue` 分别表示队列 1 和队列 2. `process_time` 表示了当前正在被处理的人的处理时间，当达到 `average_time` 时，表示处理完成。`curr_time` 表示了现在的时间，若达到了 `close_time` 银行即关门，计算所有 `Queue` 中剩余的时间。

每当存钱动作完成时，就会调用 `FindInQueue` 在第二个队列中查找是否有人符合条件可以存钱，若有，则开启第二个窗口，关闭第一个窗口。

2.2 程序结构的描述

`main.c` 中主要的函数有:

- `main()` 处理银行的窗口操作

- `FindInQueue()` 在 Q 中查找符合 `deps_money + total >= 0` 的人。

3. 调试分析

3.1: 在开始时，没有计算 当银行关闭时，尚未完成的人的时间，导致程序与答案不相符。

3.2: 在测试样例中，存在 `average_time` 为 0 的情形，之前未能处理好，于是使用如下方式进行调整：

```
if(average_time == 0) {
    average_time = 1;
    process_zero = 1;
}
```

其中 `process_zero` 代表这是 `average_time = 0` 的情形

3.3: 使用 DEBUG 的 flag 对程序的各个地方进行输出，记录各个时段的情况，方便调试错误。

4. 实验体会和收获

第二次实验使我熟悉了队列的栈的详细操作，并通过实际的样例和实验更熟悉了调试的方法和写出健壮程序的技巧。