Lab1 Report

1 一.实验要求

◆2.6⑤ 银行业务模拟

【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。

【基本要求】

利用动态存储结构实现模拟，即利用C语言的动态分配函数malloc和free。

【测试数据】

一天营业开始时银行拥有的款额为10000（元），营业时间为600（分钟）。其他模拟参量自定，注意测定两种极端的情况：一是两个到达事件之间的间隔时间很短，而客户的交易时间很长，另一个恰好相反，设置两个到达事件的间隔时间很长，而客户的交易时间很短。

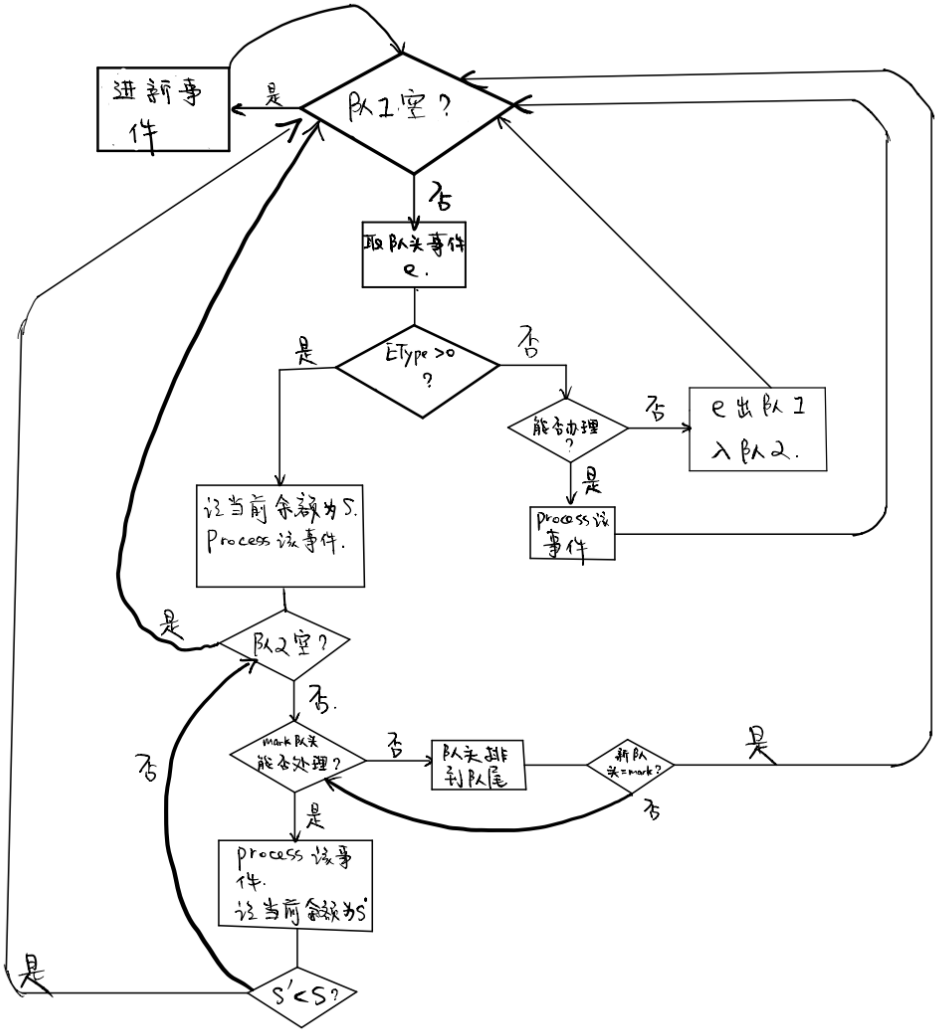
2 二.设计思路

事件驱动。

定义数据结构：一个事件结构体Event，两个顾客排队的队列Q。

初始化第一个事件，后面的事件的OccurTime由前一个事件触发，然后根据题目条件入队列Q1或Q2。同时全局变量CurTime记录当前时间，保证银行准时歇业下班。

具体流程图如下：



3 三.关键代码讲解

见注释，很详细。

```c
/**----------------------------集合定义-----------------------------**/
typedef struct event
{
    /* 一个交易：一个事件 */
    int OccurTime;      //发生时间
    int DurTime;        //交易时间
    int InterTime;      //与下一位顾客到来的时间间隔
    int Money;          //存取数目
    int EType;          //-1表示取钱，+1表示存钱
    struct event *next; //指向下一个事件
}event, *EventList;


typedef struct QNode
{
    /* 第一队节点定义，队列元素即event */
    event e;
    struct QNode *next;
}QNode;

typedef struct Queue
{
    /* 第一队 */
    QNode *head;
    QNode *rear;
}Queue;
```

```c
/**----------------------------全局变量-----------------------------**/
int TotalTime = 0;          //所有人的总等待时间
int MaxPeople = 0;          //总共接待人数
int CurTime = OpenTime;     //当前时间
int mark_money;             //mark的金额，用于快速判断队列2中的业务能否办理
int TotalPeople = 0;        //当前银行内总人数
int numq1 = 0,numq2 = 0;    //当前队列1和队列2中的人数
double t_ave;               //每个人在银行逗留的总时长
QNode tag_node;             //遍历Q2时的标志符，用于判断队列2中的业务能否办理
Queue Q1,Q2;                //队列1、2
EventList EL;               //事件表
```

```c
int NextEvent(Queue &Q,EventList &EL,event &e_next)
{
    /*时间推移，下一事件变为当前处理事件，入队列1，并由下一事件生成下下事件，
    更新入事件表，成为下一事件。返回值为0（不在营业时间内）1（在营业时间内）*/
    printf("%s","下一位顾客到来，进入队列1\n");
    event *e_nextnext;
    e_nextnext = (event *)malloc(sizeof(event));
    e_next = *EL->next;           //原事件表中元素即为原下一事件
    EnqueueQ(Q,e_next);           //将原下一事件入Q
    *e_nextnext = randomlize(e_next,*e_nextnext); //根据原下一事件，随机生成原下下事件，即新下一事件
    EL->next = e_nextnext;
    CurTime = e_next.OccurTime;   //时间推移，下一事件变为当前事件，更新CurTime
    e_next = *EL->next;           //原下下事件变为新下一事件
    numq1 ++;                     //队列1人数增加
    MaxPeople ++;                 //总接待人数增加
    TotalPeople ++;               //当前银行内人数增加
    if(CurTime > CloseTime) return 0;
    return 1;
}
```

```
int ProcessQ(Queue &Q,EventList &EL, int kind)
{
    /*一队队头出一队，记为当前事件cur_1,process它。同时，在当前事件的交易时间段内看有没有新的事件发生，
    即OccurTime处于当前process事件的OccurTime~OccurTime+DurTime之间的事件，需要让它入队列1，并且随
    机置他的后来事件，入事件表，成为新下一事件*/
    QNode cur_q;                            //当前需要process的事件
    event e_next,e_nextnext;                //原下个事件和原下下个事件
    DeQueueQ(Q,cur_q);                      //当前事件出队，赋值给cur_q
    TotalMoney += cur_q.e.Money;            //处理交易后的totalmoney
    e_next = *EL->next;                     //原下个事件
    int t_end_cur,t_start_cur,t_left;
    int t_wait = 0;
    t_start_cur = CurTime;                  // t_start_cur 是开始 处理当前事件的 时刻
    t_end_cur = CurTime + cur_q.e.DurTime; // t_end_cur 是 处理完当前process事件后 的 时刻
    t_left = t_end_cur - t_start_cur;       // t_left 是 距离当前事件处理完成 还需要的 时间
    while(e_next.OccurTime < t_end_cur)     // 若原下一事件 发生的时间 在 当前process事件处理完时刻 之前，
                                            // 则将原下一事件入队列1排队，同时随机生成下下个事件加
                                            // 入事件表，成为新下一事件
    {
        t_left = t_end_cur - e_next.OccurTime;
        t_wait = e_next.OccurTime - CurTime;        // t_wait 指 当前银行内所有的人都在等待 的时间段
        TotalTime += t_wait * TotalPeople;          // 计算等待时间
        if(NextEvent(Q,EL,e_next) == 0) return 0;   // 推动事件发生，置新下一事件（原下一事件入队列1）
                                                    // 若NextEvent返回值为0，代表银行已经关门，
                                                    // 于是停止处理当前处理的事件
    } //把处理业务期间所有新的事件全部入Q1排队，并update EL中的内容
    TotalTime += t_left * TotalPeople;    // 计算最后大家一起等到当前业务办理结束的时间
    TotalPeople --;                       // 业务办理完，银行内人数减1（事实上队列中人数也需要减1，在主函数中减）
    CurTime = t_end_cur ;                 // 更新CurTime
    return 1;
}
```

4 四.调试分析

调试正常。

5 五.代码测试

case1：业务办理时长和两件业务间隔时长相差不大

```
请输入业务办理时长的下、上界:30 40
请输入两件业务之间间隔时长的下、上界:30 40
请输入办理业务金额的下、上限：（负代表取钱，正代表存钱）-4000 4000
开业
初始化队列1...
初始化队列2...
下一位顾客到来，进入队列1
当前时间为0;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为1人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为38;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为2人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为70;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为3人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为105;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为4人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为143;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为5人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为175;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为6人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为210;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为7人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为248;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为8人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为279;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为9人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束
```

```
当前时间为314;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为10人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为352;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为11人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为383;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为12人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为417;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为13人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为455;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为14人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为486;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为15人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为520;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为16人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为557;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为17人
开始办理队一业务
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为588;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为18人
开始办理队一业务
下一位顾客到来，进入队列1
银行关门
总时间为791
总人数为19
平均时间为41.631579
```

case2：业务办理时长 > >两件业务间隔时长

```
请输入业务办理时长的下、上界:80 100
请输入两件业务之间间隔时长的下、上界:5 10
请输入办理业务金额的下、上限:（负代表取钱，正代表存钱）-4000 4000
开业
初始化队列1...
初始化队列2...
下一位顾客到来，进入队列1
当前时间为0;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为1人
开始办理队一业务
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为96;当前队一有13人;当前队二有0人;当前银行内有13人;截至目前总接待人数为14人
开始办理队一业务
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为176;当前队一有23人;当前队二有0人;当前银行内有23人;截至目前总接待人数为25人
开始办理队一业务
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
当前业务办理结束
```

```
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为345;当前队一有46人;当前队二有0人;当前银行内有46人;截至目前总接待人数为50人
开始办理队一业务
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为434;当前队一有58人;当前队二有0人;当前银行内有58人;截至目前总接待人数为63人
开始办理队一业务
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
当前业务办理结束

当前时间为527;当前队一有70人;当前队二有0人;当前银行内有70人;截至目前总接待人数为76人
开始办理队一业务
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
下一位顾客到来，进入队列1
银行关门
总时间为24335
总人数为87
平均时间为279.712644
```

case3:业务办理时长 < < 两件业务间隔时长

```
请输入业务办理时长的下、上界:5 10
请输入两件业务之间间隔时长的下、上界:80 100
请输入办理业务金额的下、上限:（负代表取钱，正代表存钱）-5000 5000
开业
初始化队列1...
初始化队列2...
下一位顾客到来，进入队列1
当前时间为0;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为1人
开始办理队一业务
当前业务办理结束

下一位顾客到来，进入队列1
当前时间为80;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为2人
开始办理队一业务
当前业务办理结束

下一位顾客到来，进入队列1
当前时间为168;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为3人
开始办理队一业务
当前业务办理结束

下一位顾客到来，进入队列1
当前时间为265;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为4人
开始办理队一业务
当前业务办理结束

下一位顾客到来，进入队列1
当前时间为362;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为5人
开始办理队一业务
当前业务办理结束

下一位顾客到来，进入队列1
当前时间为447;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为6人
开始办理队一业务
当前业务办理结束

下一位顾客到来，进入队列1
当前时间为541;当前队一有1人;当前队二有0人;当前银行内有1人;截至目前总接待人数为7人
开始办理队一业务
当前业务办理结束

下一位顾客到来，进入队列1
银行关门
总时间为44
总人数为8
平均时间为5.500000
```

6 六.实验总结

1. 完成基本内容
2. 完成事件驱动的统计及清晰输出，一目了然

7 七.附录

lab1.cpp：完成lab01所有代码