|  |  |
| --- | --- |
|  |  |

**《数据结构与算法课程设计》课程**

**区块链客户端设计**

**实验报告**

# 设计要求

## 基本算法设计

#### 创建交易单和区块链数据结构

#### 实现SHA256算法

#### 实现工作量证明算法

#### 实现区块对比和区块链对比算法（验证区块链合法，防篡改）

## 系统算法设计

#### 要求设计程序同时在三个节点运行

#### 三个节点输入相同交易单数据，分别将新得到的交易单存入各自节点队列

#### 当队列长度达到4时，生成新区块，并清空队列，同时运行工作量证明算法

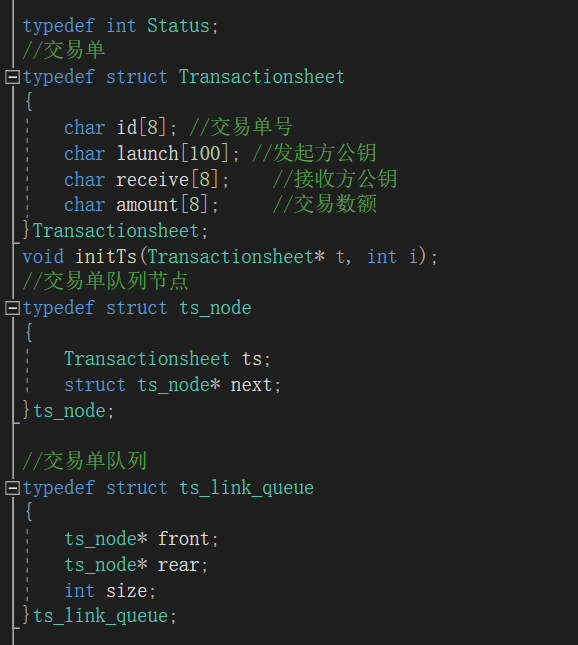
#### 当工作量证明算法满足条件时，直接将新区块加入区块链中

#### 当工作量证明算法不满足条件时，以任意其他节点新区块和区块链为输入，与本地新区块和区块链对比（验证区块链合法），当相同时，将新区块加入本地区块链。

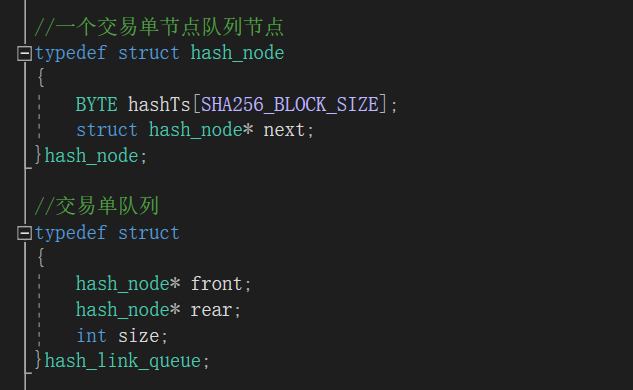
# 设计内容

## 数据结构设计

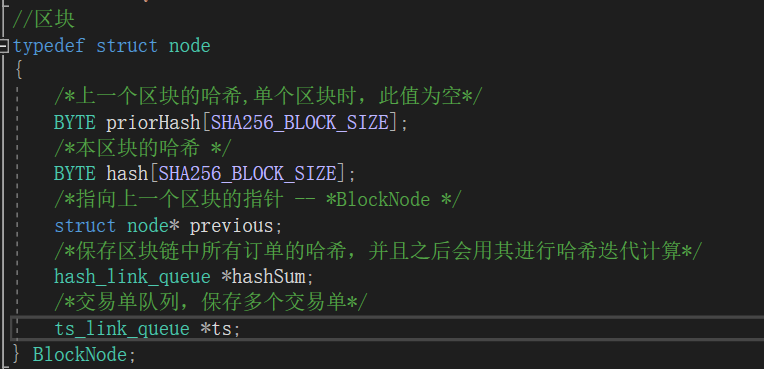
#### 交易单队列



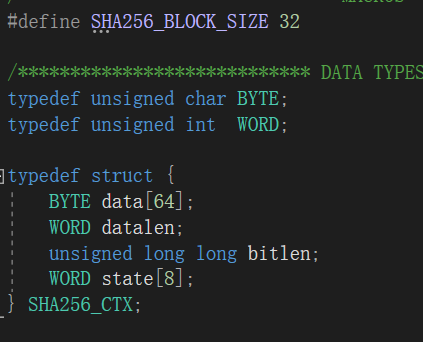
#### 区块交易单hash队列



#### 区块

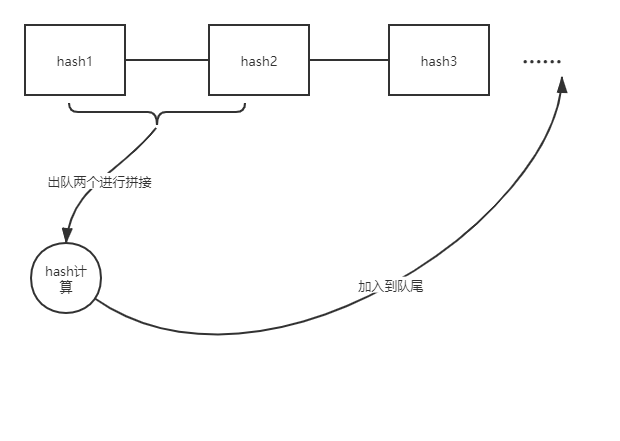


#### SHA256



## 普通算法设计

#### 区块hash计算



* 1. 将区块交易记录（本实验是4个交易单）分别进行hash计算，并把计算好的hash保存到区块hash队列中。
  2. 每次出队两个hash值，将这两个hash进行拼接，将拼接后的hash进行hash计算，然后把计算的hash值加入到区块hash队列，如此循环，直到区块hash队列长度为1，即该值是区块hash，区块hash计算完毕。

#### 区块上链

* 1. 定义一个临时区块指针，将其指向本地区块链链尾节点。
  2. 将指向本地区块链链尾的区块指针hash赋值给新区块的“上一区块hash”属性。
  3. 将本地区块链尾指针的下一指针指向新区块。

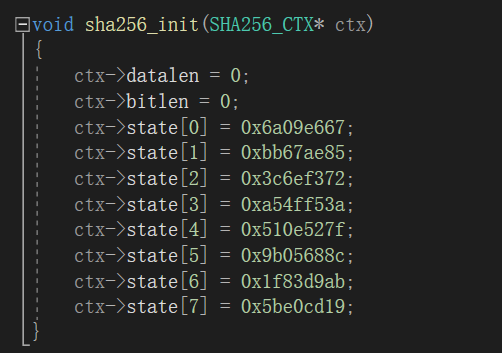
#### 数据读取与写入模拟区块间通信和数据保存。

* 1. 数据格式：每一行四段数据，每一段长度为7，一个区块4行数据，每一行以\n结束。
  2. 为了模拟几个客户端之间相互获取新区块信息，默认设定new1.txt,new2.txt,new3.txt为三个客户端保存新区块的信息的“共享空间”，通过读取其他客户端对应的文件，得到对应的数据，以作为输入。
  3. 每当新的区块上链后，程序将把该区块信息写入到本地文件中，模拟实际中的数据库存储。

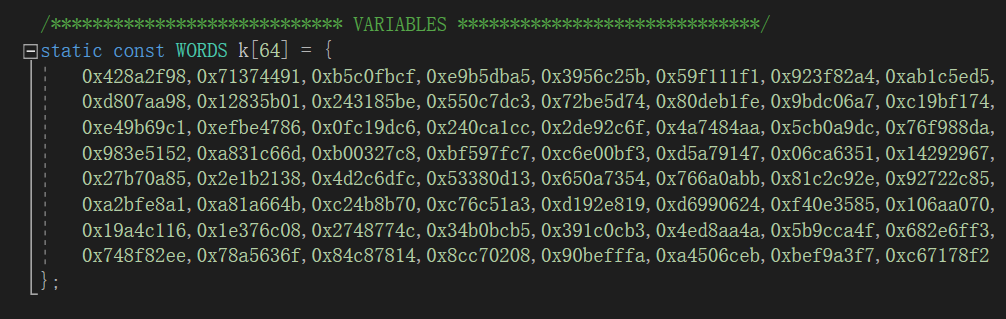
## SHA256算法设计

#### 初始化

8个hash初值。



64个hash常量。



#### 信息预处理

1. 附加填充比特

在报文末尾进行填充，使报文长度在对512取模以后的余数是448

填充是这样进行的：先补第一个比特为1，然后都补0，直到长度满足对512取模后余数是448。

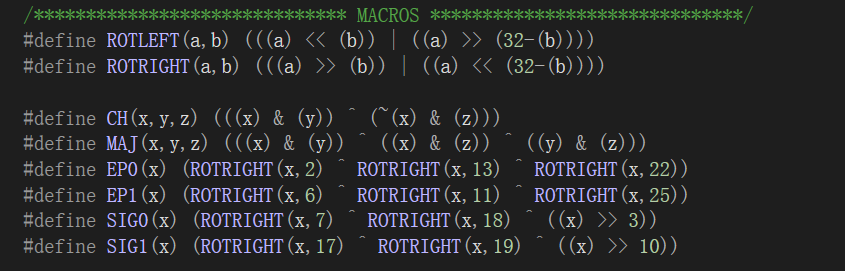
需要注意的是，信息必须进行填充，也就是说，即使长度已经满足对512取模后余数是448，补位也必须要进行，这时要填充512个比特。

因此，填充是至少补一位，最多补512位。

1. 附加长度值

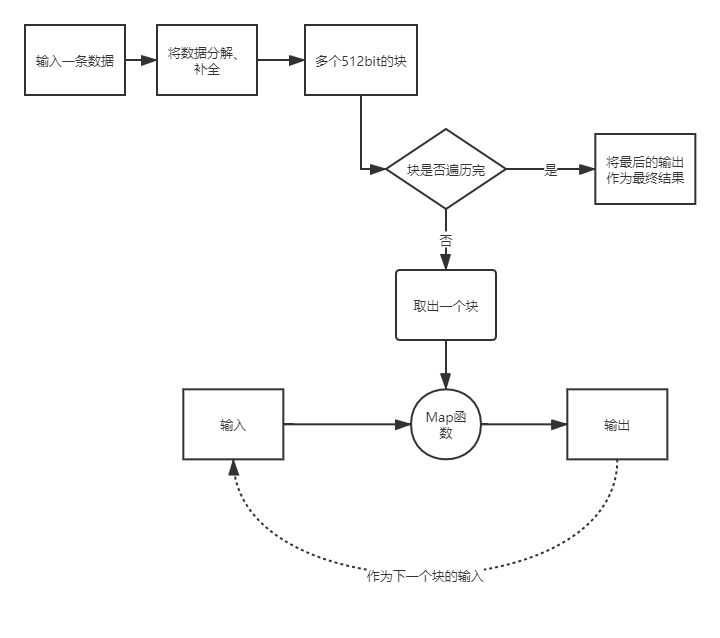
附加长度值就是将原始数据的长度信息补到已经进行了填充操作的消息后面。

#### 逻辑运算

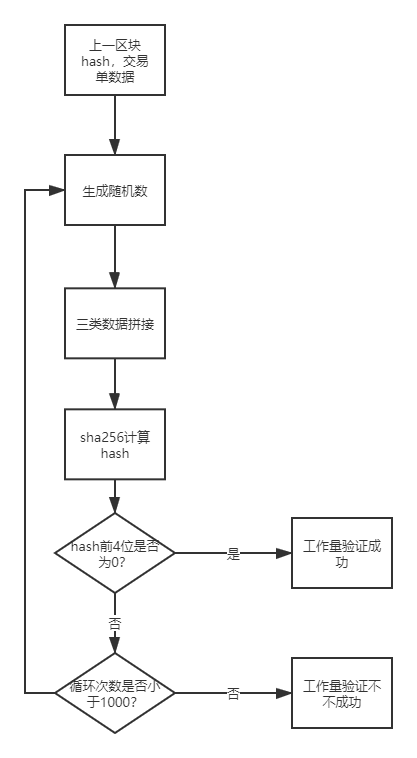


#### 计算消息摘要

1. 将消息分解成512bit大小的块，剩下不完整的块需要被补全，对应sha256\_final函数。
2. 然后对每个块进行hash映射，第i块映射的结果作为第i+1块的输入，此过程在sha256\_update函数中完成。
3. 进行循环加密，不断将原始信息进行打乱。

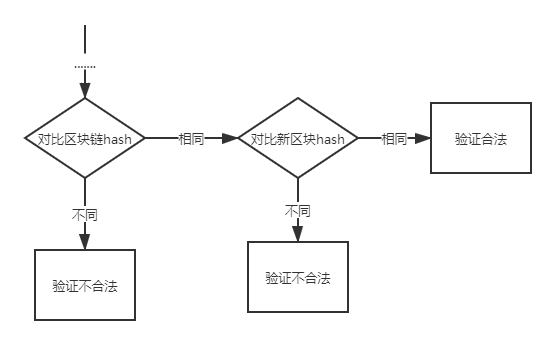


## 工作量算法设计



* 1. 得到上一区块hash，新区块交易单作为输入。
  2. 生成一个随机数，将上一区块hash，交易单数据三类数据进行连接。
  3. 用sha256算法将连接的数据进行加密计算，得到hash值。
  4. 校验得到的hash值，如果hash值的前2位不为0，则重复b，c，d三步。计算最多循环计算1000次，如果1000次都没有找出随机数，则工作量证明失败，否则跳出循环，工作量证明成功。
  5. 工作量证明是否成功，与“难度系数有关”。本设计中，暂定为前8位为0。

## 区块安全验证算法设计



1. 比对本节点与其他节点的区块链hash。如果区块链长度不一样，区块验证不合法；如果区块链长度一样，则依次对比链上的每一块区块hash，全部相同则验证合法，否则不合法。
2. 对比新区块hash，将本节点新区块hash与其他节点新区块hash对比，如果相同，则区块验证合法，否则不合法。
3. 必须满足本节点和其他节点的区块链hash和新区块hash都相同时，区块验证才合法。

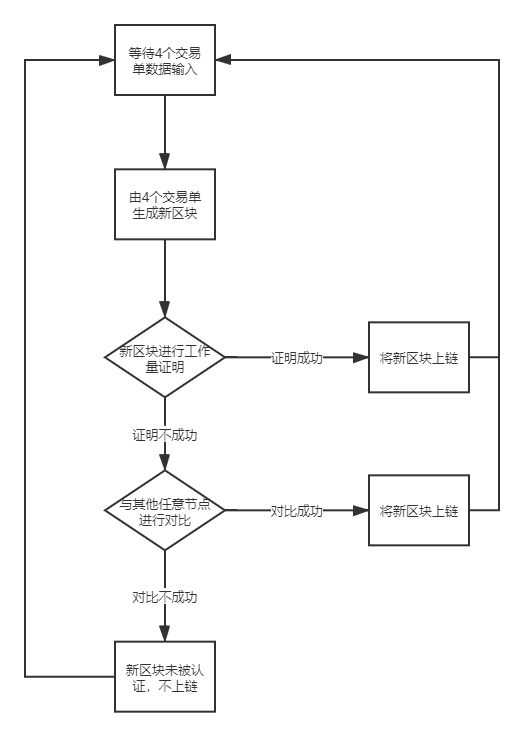
## 系统设计

* 1. 各个文件主要完成的功能。

|  |  |
| --- | --- |
| 文件名 | 功能简述 |
| sha256 | 对字符串进行加密 |
| tsQueue | 订单队列的基本操作 |
| hashQueue | 区块订单hash队列的基本操作 |
| workProof | 工作量证明 |
| block | 区块初始化，上链等区块操作 |
| safe | 区块防篡改，主要是区块hash比对 |
| run | 主程序，控制这个程序 |

* 1. 数据恢复，程序首先读取本地文件，初始化本地区块链。
  2. 新区块能否上链。

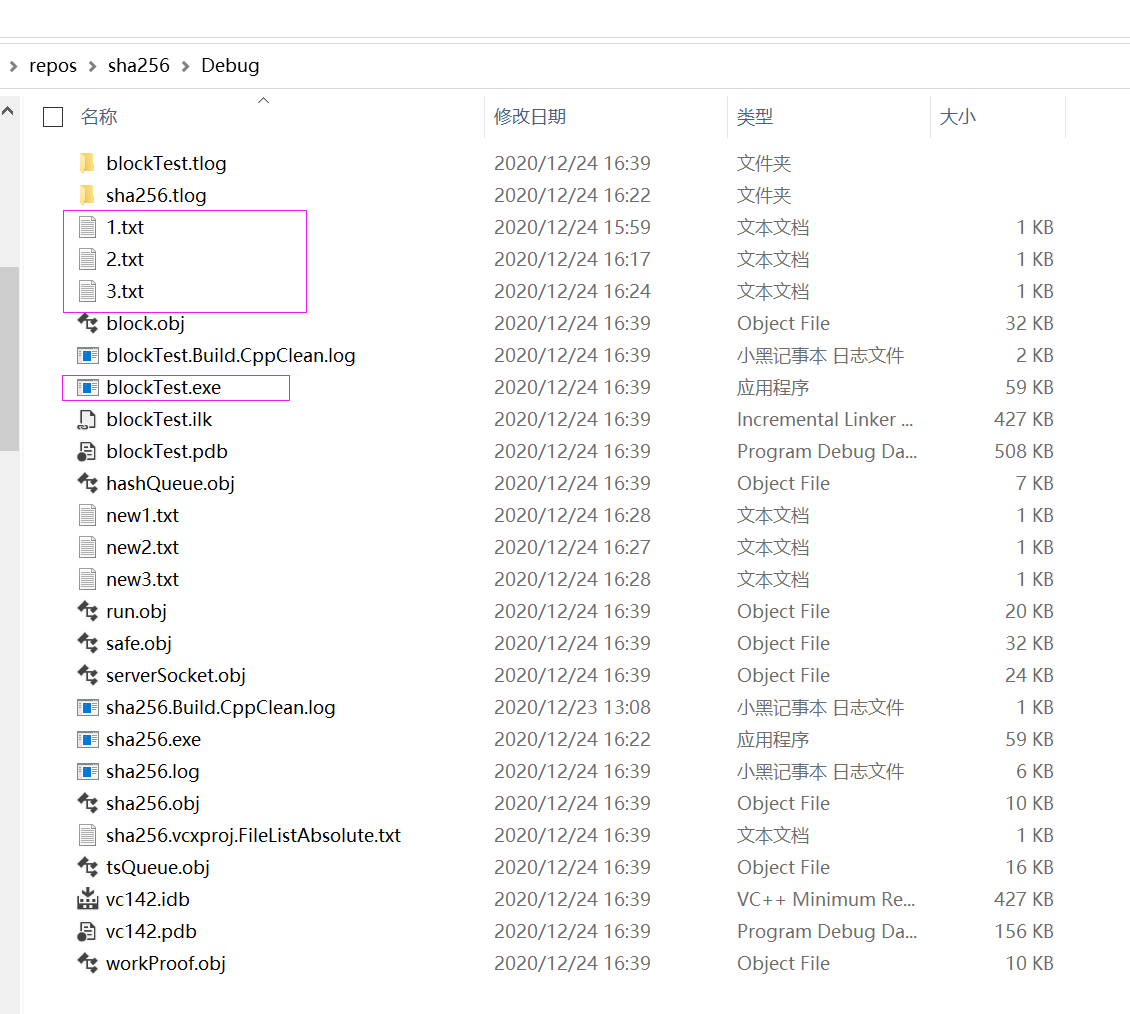
三个节点同时输入相同交易单数据（4个），进行工作量证明。工作量证明如果成功，则创世区块建立成功；如果工作量证明不成功，对比其他节点区块hash和区块链hash，都相同，则创世区块建立成功；否则不成功。



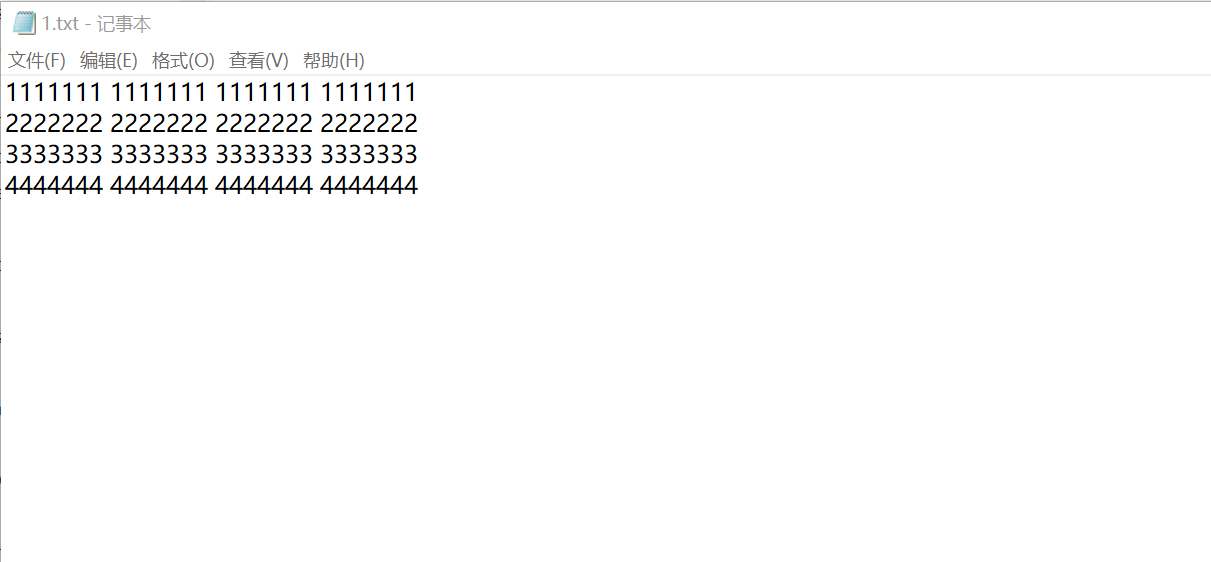
* 1. 如果能够上链，将数据写入到本地文件中。

# 运行结果

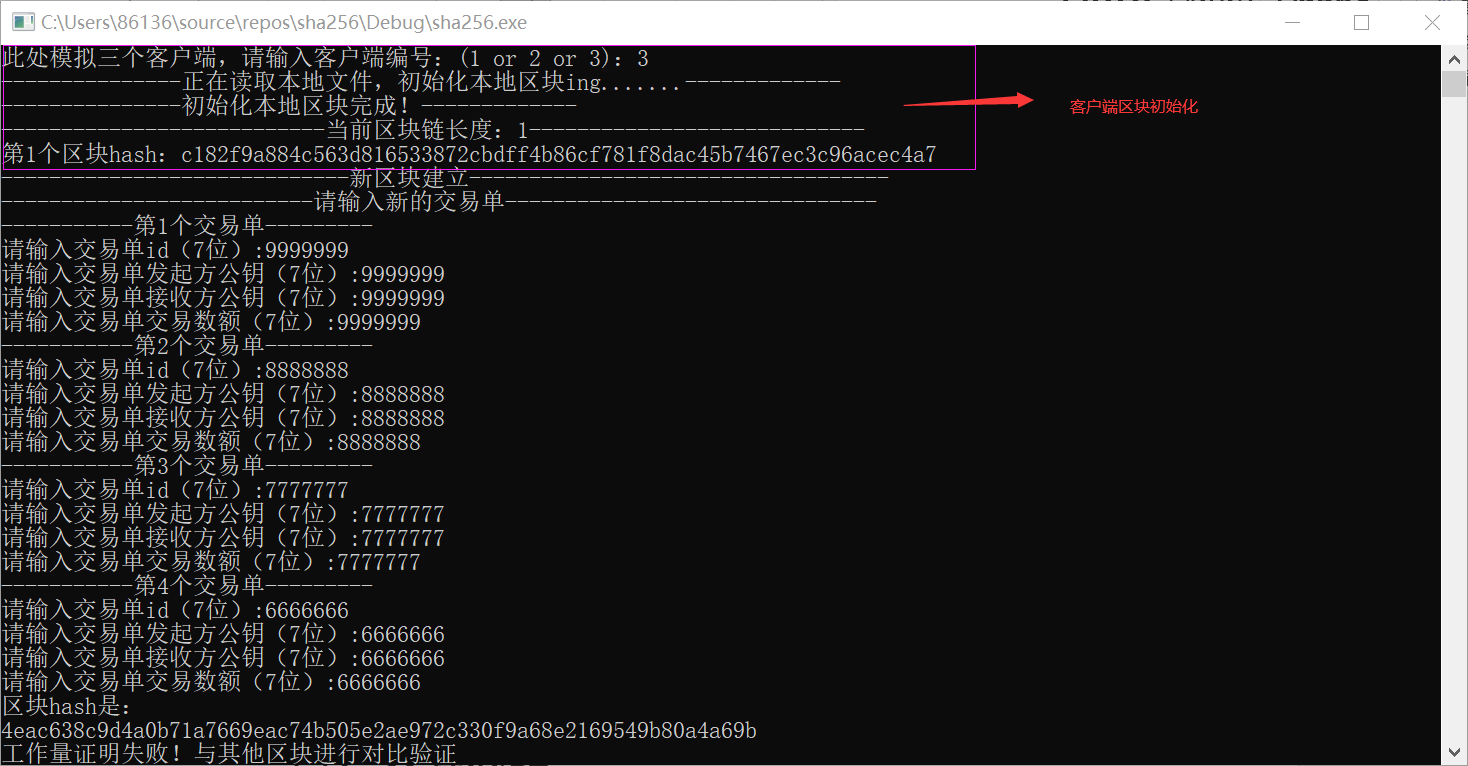
1. 打开三个客户端，分别输入客户端编号，程序会默认读取存放本地区块链数据的文件，进行区块链初始化。
   1. 事先设置了三个本地文件1.txt，2.txt，3.txt，三个文件内容一样，主要是在模拟三台计算机上各自的本地数据文件。注意三个数据文件要与.exe程序在同一目录。



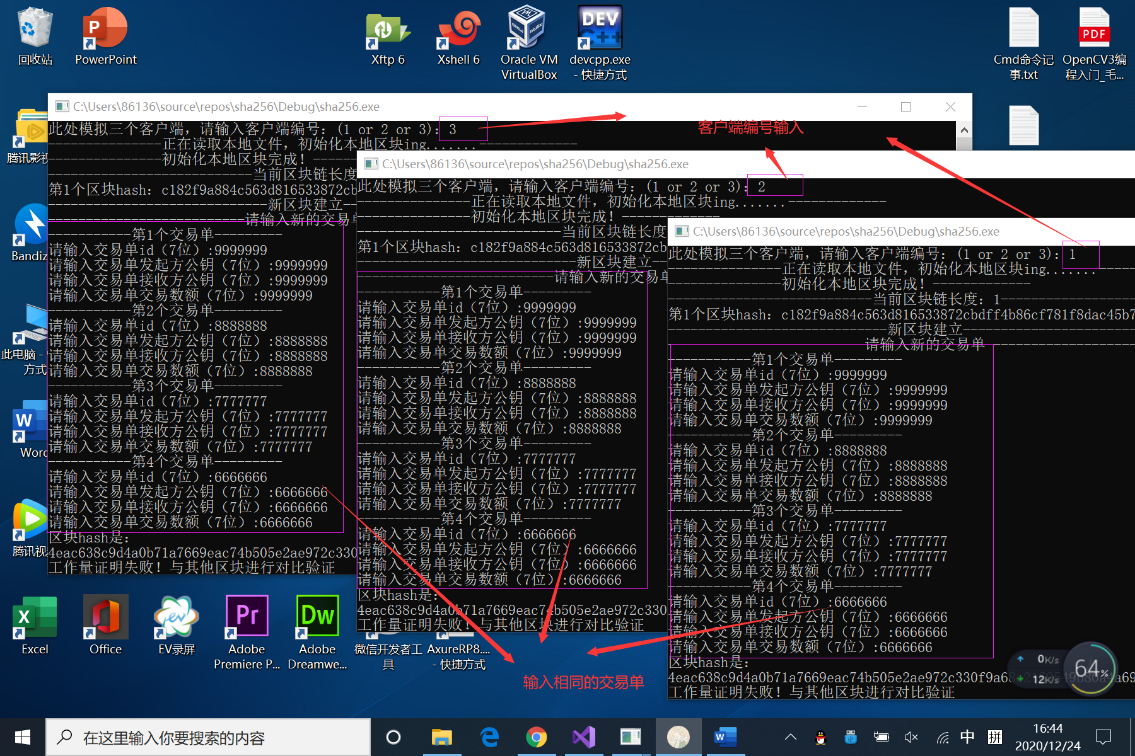
* 1. 具体内容如下，一行代表一个交易单数据：



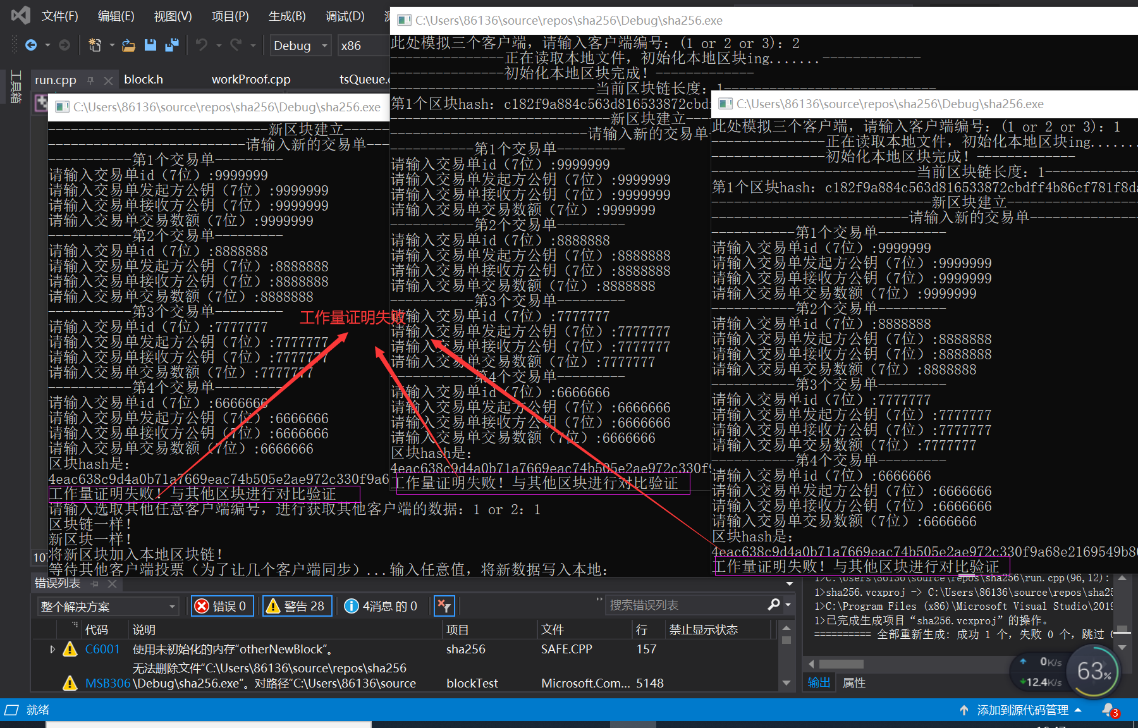
1. 初始化完成。



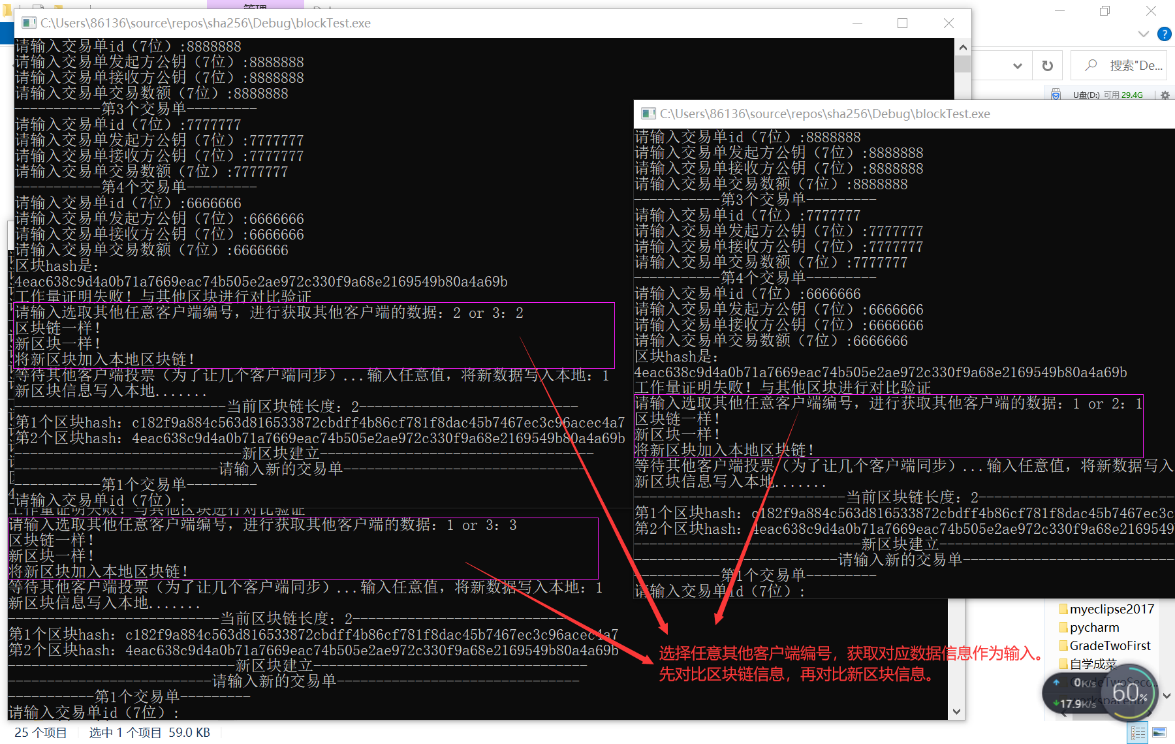
1. 输入4个交易单数据，如图所示。



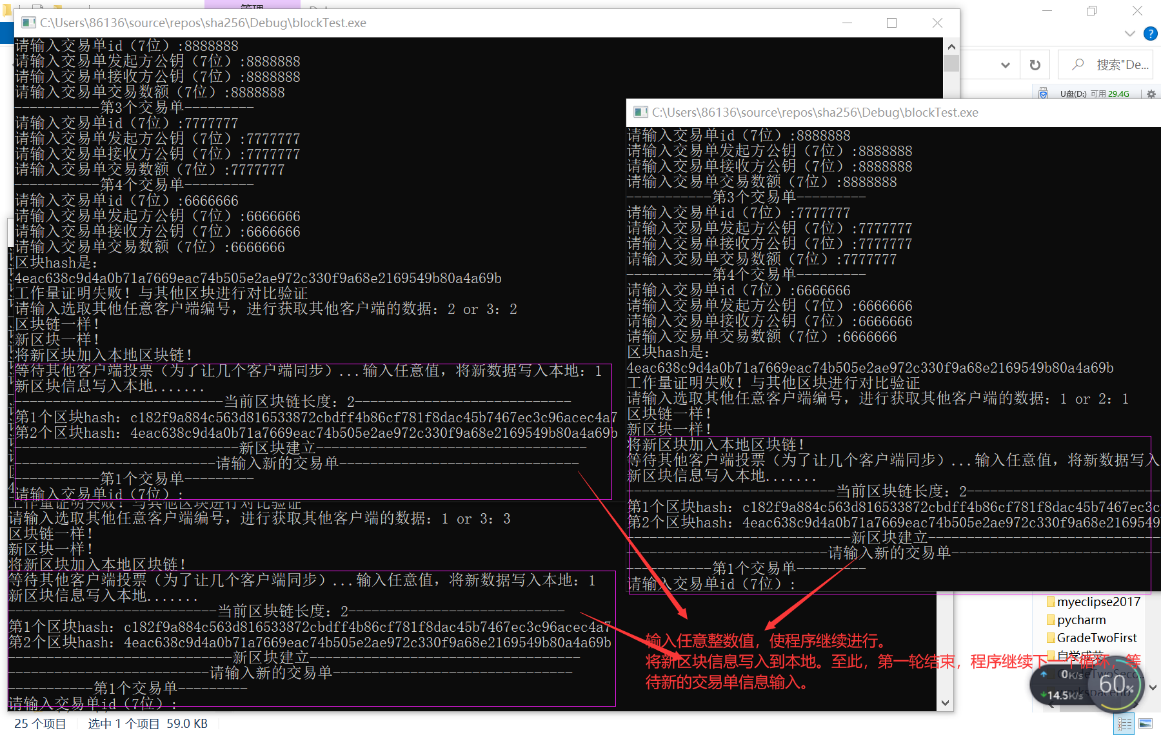
1. 工作量证明计算，结果如下：



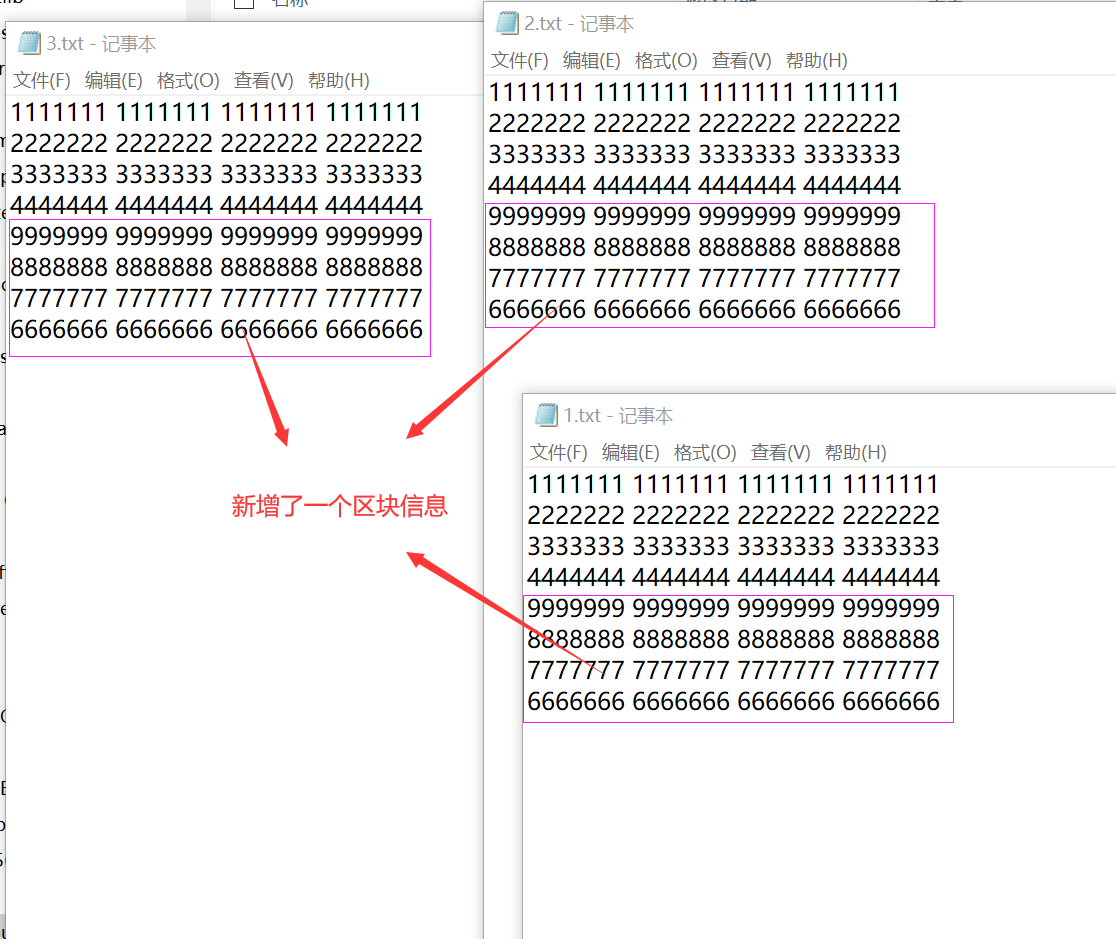
1. 用户任意选择其他客户端编号进行区块链比对和新区块比对，结果如下：



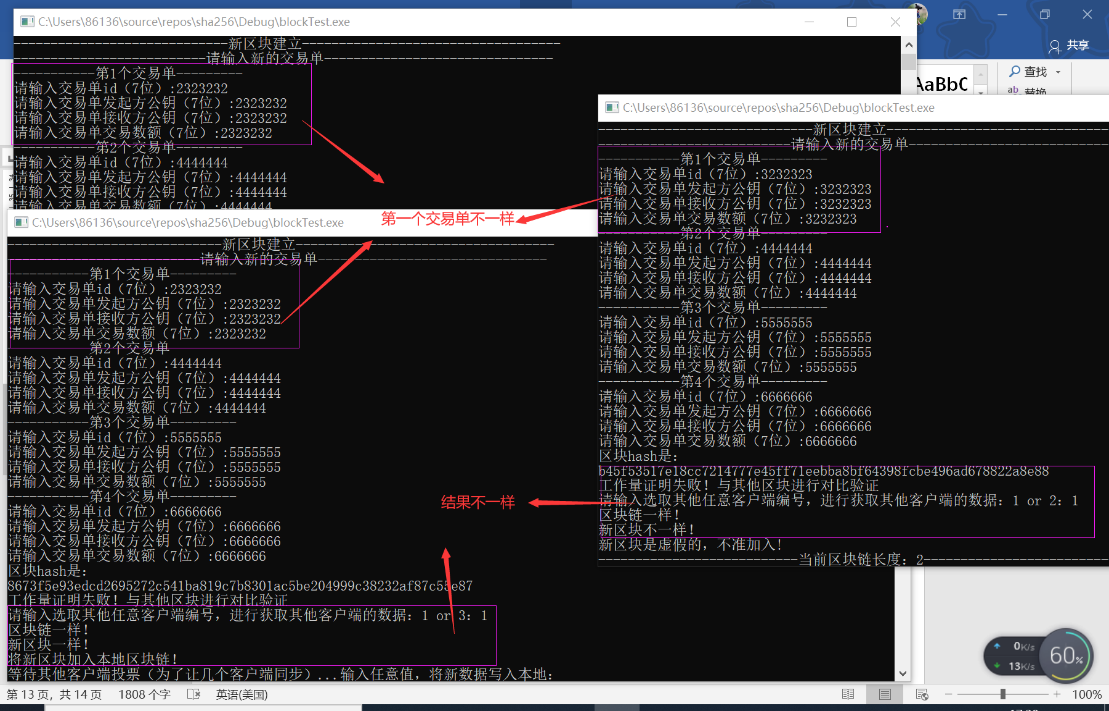
1. 对比成功，用户输入任意键，更新本地数据文件。



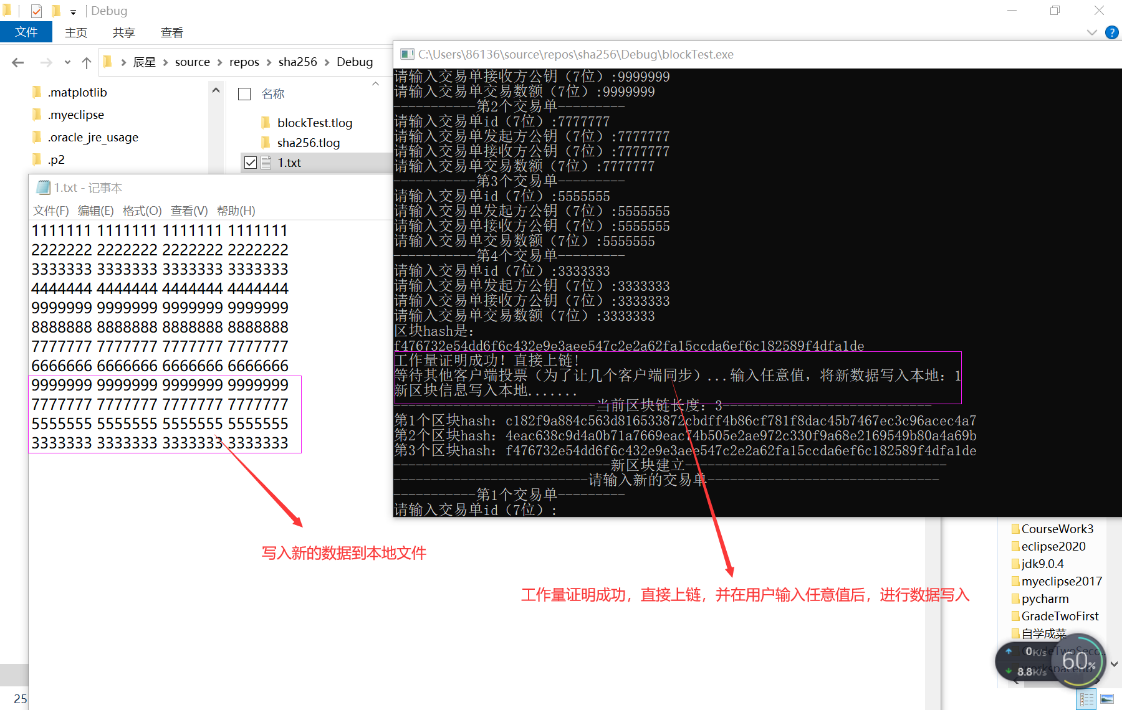
1. 查看本地文件，数据是否写入。



1. 进入下一个循环，测试输入的交易单不同时的结果。



1. 调节一下难度系数，额外测试工作量证明成功的情况，结果如下。



# 四、实验总结

1.本实验设计主要使用到了链表，队列的数据结构，主要有出队、入队、遍历数据元素，获取链中指定位置的数据等常见方法；除此之外，还有字符串拼接，文件读写操作，sha256算法等，较熟练地掌握了基本的数据结构。

2.因为没有实现通信与多线程，故采用文件的读写来尽量模拟。事先给每个客户端配备一个txt，用于保存本地区块链数据；同时给每个客户端再配备一个txt文件，用户存放新区块的信息，其他客户端通过读取该客户端的数据文件，以模拟实现其他客户端获取该客户端的数据。

3.在本实验中，char，int，unsinged char等字符串之间的相互转化有些重要，稍不注意会导致数据的丢失、不完整。

4.本实验主要模拟了对区块链的大致流程，包括由交易记录生成新区块；区块工作量证明；区块防篡改，数据的保存与恢复。让我们基本了解了区块链的工作原理。