# 作业1：SHA256算法

## 代码实现与运行：

* 源码见SHA256.cpp
* 运行方式：

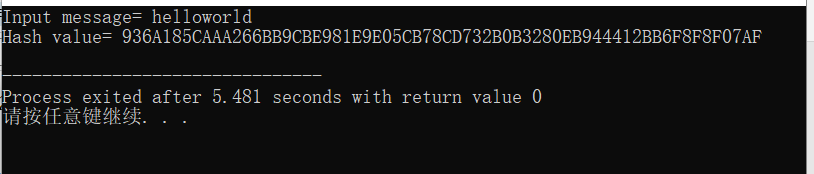
方法1：点击可执行文件SHA256.exe，在终端中输入需要进行哈希的信息，按回车键，即得到经过SHA256算法后的哈希值。

方法2：编译SHA256.cpp，得到可执行文件SHA256.exe，在终端中输入需要进行哈希的信息，按回车键，即得到经过SHA256算法后的哈希值。

## SHA256算法特性分析

**1.无法通过SHA256算法得到的哈希值反推回原输入的值。**

输入“helloworld”字符串信息，经过SHA256算法会得到一个毫无规律并且毫无关联的64位16进制数字串，并且无法从该串反推回原数据。

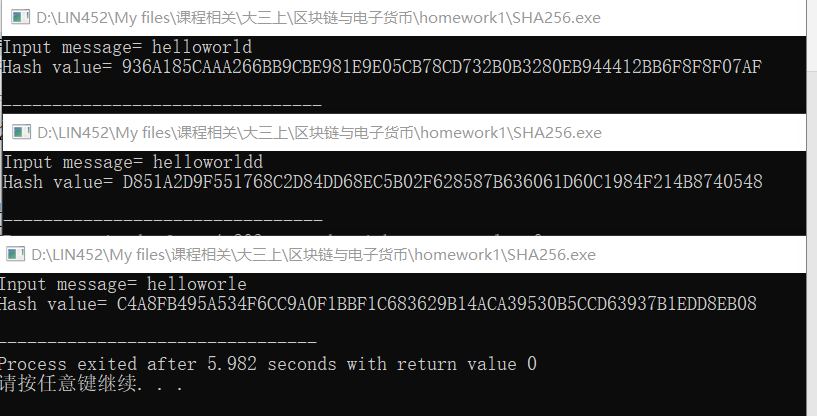


**该特性能够用于进行共识机制，来进行算力竞争解决记账权。**

因为无法通过哈希值推出输入值，而只能够通过不断验证输入的值来确定该输入值。

**2.不同输入信息得到的哈希值不同。**

输入互相有微小差别的字符串“helloworld”“helloworldd”“helloworle”，经过SHA256算法得到的哈希值是完全不同的。

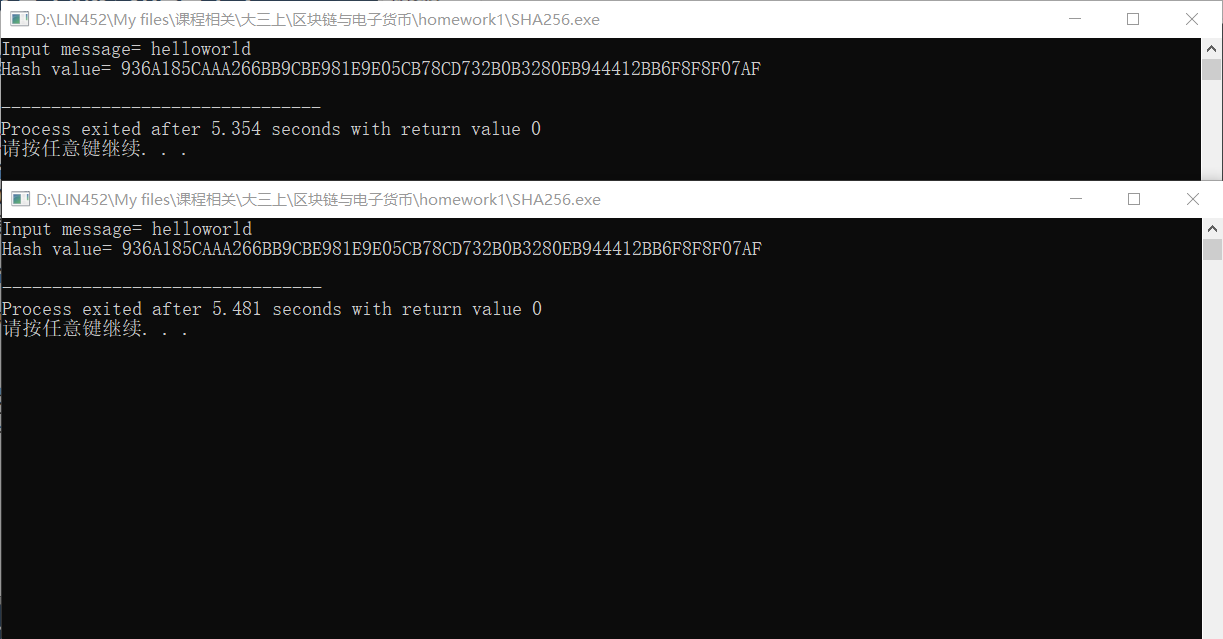


**该特性可以用于生成区块头的数字指纹，标识区块和Merkle树的归纳校验。**

由于不同输入得到的输出哈希值各不相同，我们可以用该哈希值来作为区块的标识符和Merkle树节点的哈希值。

**3.固定输入得到的输出是确定的。**

多次输入同一输入“helloworld”，得到的输出是相同的。



**该特性能够用于进行Merkle树的校验。**

由于相同输入得到输出是固定的，因此可以通过哈希值的改变来判断Merkle树种某个数据（交易）发生了修改，并且快速定位被修改的数据。

## 实验设计与分析

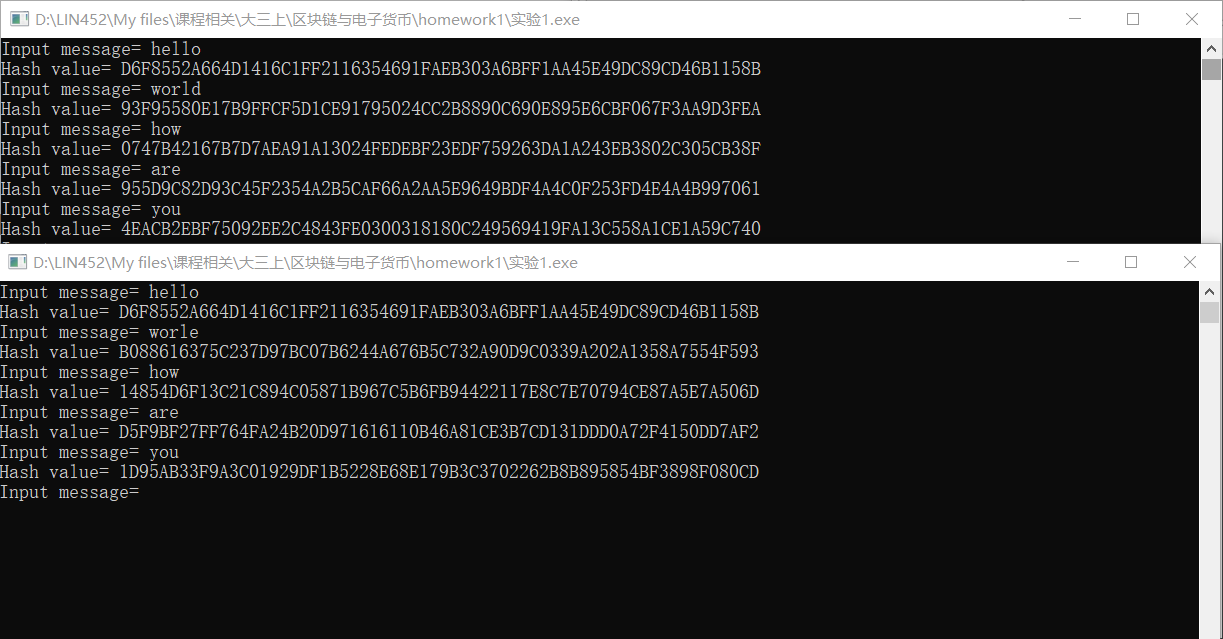
**实验1：区块数据锁定实验**

将SHA256的代码稍作修改,使其能够连续处理几条输入,并且每输入一条消息,将之前的哈希值加到输入后面再进行依次哈希,以此类推（模拟区块链的哈希形成）。

输入1：“hello” “world” “how” “are” “you”

输入2：“hello” “worle” “how” “are” “you”

得到的输出如图：



可以发现输入hello时输出的哈希值是相同的，而修改了world一词的一个字符之后，后面的所有输出都完全不同，可见当区块链中任意区块的任意数据修改之后，此后的所有哈希值都会变化。

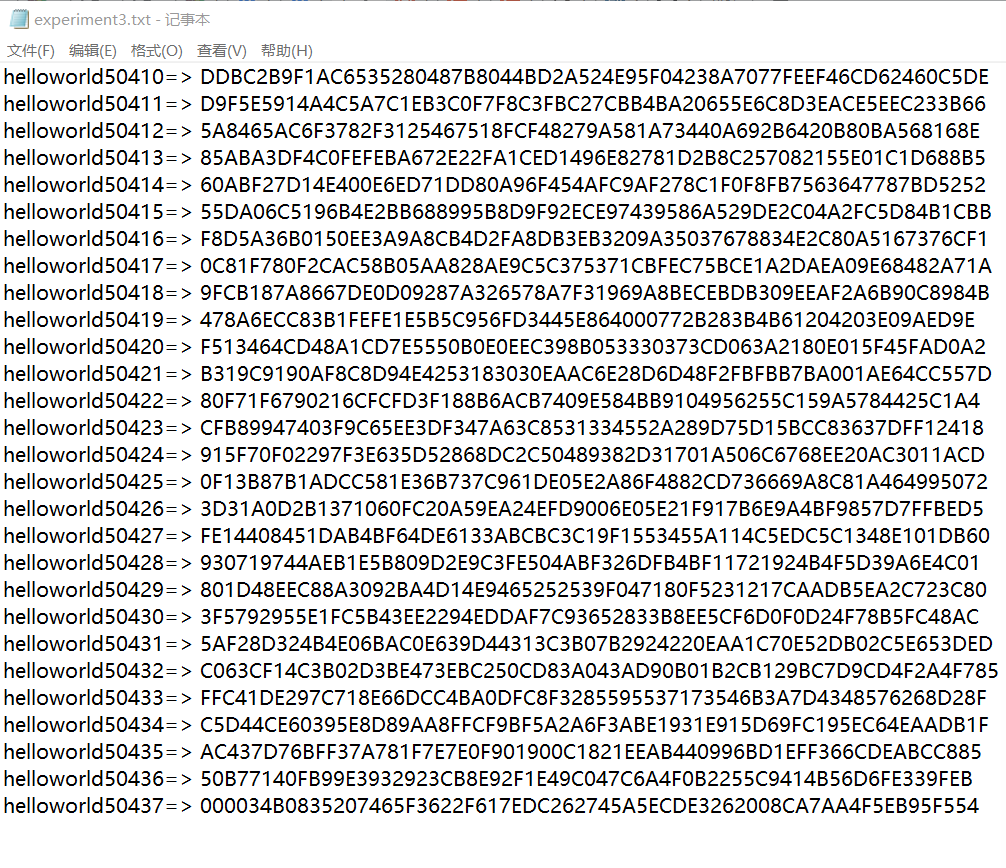
因此运用SHA256算法后无法单独修改某一数据，由此起到了锁定数据的作用。

**实验2：PoW实验**

输入”helloworld”字符，作为假定的区块头，我们想要找到一个随机值nonce，使得helloworld字符串后面加上nonce之后经过哈希得到的值是小于2^240的,即十六进制下,最高4位的值为”0000”。

我们将输出每一个nonce值对应的哈希结果，直到找到满足条件的nonce值。

由于输出过于庞大，只截取了部分，全部输出可见“experiment3.txt”。



可以看到当nonce值为50437时，得到了满足条件的哈希值。