**Golang và Blockchain**

Mục lục

[**I.** **Cấu trúc của một Blockchain** 2](#_Toc108635540)

[**a)** **Block** 2](#_Toc108635541)

[1. Block cơ bản: 2](#_Toc108635542)

[2. Tính hash (hàm SetHash): 2](#_Toc108635543)

[3. Tạo 1 block (hàm NewBlock) 3](#_Toc108635544)

[**b)** **BlockChain** 3](#_Toc108635545)

[1. BlockChain phiên bản đầu tiên 3](#_Toc108635546)

[2. Thêm một Block vào BlockChain (Hàm AddBlock) 4](#_Toc108635547)

[3. Tạo Block đầu tiên cho 1 BlockChain (hàm GenesisBlock) 4](#_Toc108635548)

[4. Khởi tạo 1 BlockChain (hàm InitBlockChain) 4](#_Toc108635549)

[**c)** **Kiểm tra sự hoạt động của BlockChain chúng ta tạo ra** 5](#_Toc108635550)

[**II.** **Cơ chế đồng thuận dựa trên thuật toán Poof of work** 6](#_Toc108635551)

[**a)** **Cấu trúc lại BlockChain ở phần 1** 6](#_Toc108635552)

[1. Thêm biến Nonce vào cấu trúc Block 6](#_Toc108635553)

[2. Thay đổi hàm NewBlock 6](#_Toc108635554)

[3. Chú ý quy tắc đặt tên thuộc tính trong class 6](#_Toc108635555)

[**b.** **Cấu trúc lại BlockChain ở phần 1** 7](#_Toc108635556)

[1. Thiết lập độ khó cho việc tìm hash hợp lệ 7](#_Toc108635557)

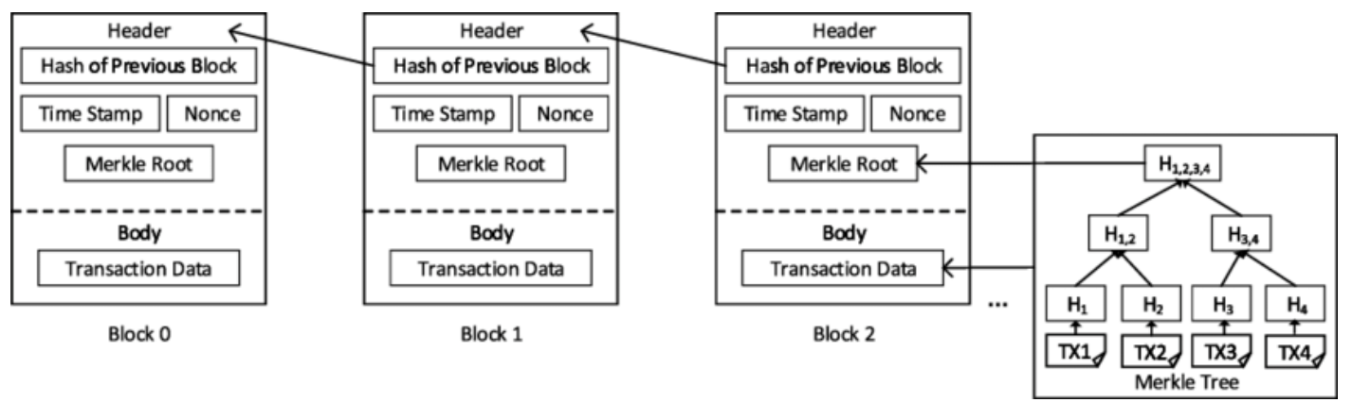
[2. Cấu trúc của ProofOfWor và hàm tạo ra ProofOfWord 7](#_Toc108635558)

[3. Tạo ra biến data như info như ở phần 1 7](#_Toc108635559)

[4. Tìm kiếm hash hợp lệ 8](#_Toc108635560)

[**c.** **Kiểm tra sự hoạt động của BlockChain chúng ta tạo ra** 9](#_Toc108635561)

1. **Cấu trúc của một Blockchain**

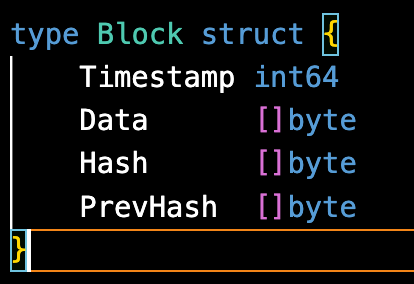
****

*Cấu trúc của một blockchain*

1. **Block**
2. Block cơ bản:

Một block cơ bản gồm 4 thành phần:

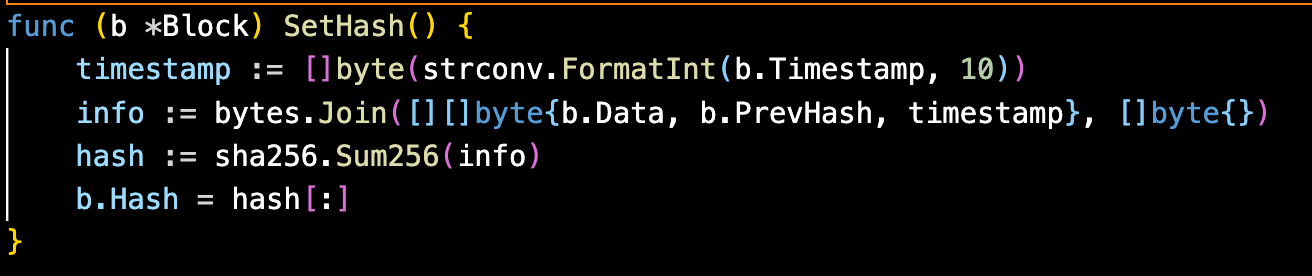
* Timestamp: Đánh dấu thời điểm của block được tạo ra.
* Data: Lưu trữ những thông tin quan trọng của một block.
* Hash: Gọi là hash của block hiện tại. (Hash này được tạo ra bởi việc ghép các thông tin như prevhash+data+timestamp của block hiện tại sau đó mã hóa nó gọi chung là cách tính hash)
* PrevHash: Là hash của block trước đó.



* *Trong đặc tả kỹ thuật của Bitcoin, Timestamp, PrevBlockHash, Hash nằm trong phần block headers, còn các transactions (như Data ở trên, trong bitcoin là thông tin giao dịch) được lưu riêng ở phần dữ liệu khác. Hiện tại chúng ta đang lưu các trường chung trong một struct để đơn giản hoá.*

1. Tính hash (hàm SetHash):

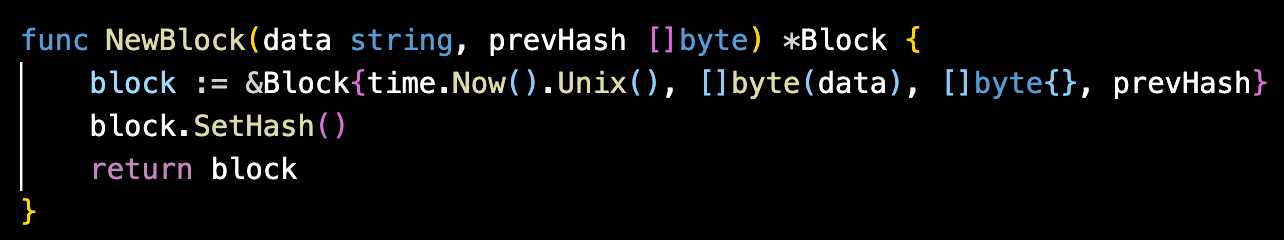
* b: Con trỏ kiểu Block
* timestamp: Lưu trữ thông tin thời gian block được tạo ra sau khi nó đã được chuyển sang kiểu []byte (mảng byte)
* info: Đưa các thông tin như Data, PrevHash, timestamp từ các mảng kiểu byte 1 chiều riêng lẻ vào trong một mảng kiểu byte 2 chiều và Join nó với 1 mảng byte rỗng
* hash: Dùng thuật toán băm sha256 để mã hóa info

****

1. Tạo 1 block (hàm NewBlock)

Hàm NewBlock trả về một con trỏ kiểu Block:

* block: Tạo ra một Block và truyền các giá trị vào cho nó trừ Hash vì Hash sẽ được tạo ở hàm SetHash
* block.SetHash(): Gọi hàm SetHash để tạo ra Hash cho new Block

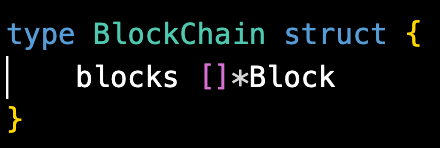


1. **BlockChain**

***blockchain là một bộ dữ liệu có cấu trúc như một linked-list. Các block được lưu theo thứ tự và mỗi block đều có con trỏ link tới block liền kề phía trước.***

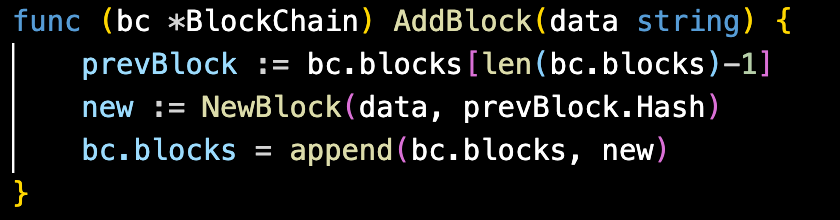
1. BlockChain phiên bản đầu tiên

* Một Blockchain chính là một mảng gồm các phần tử con trỏ kiểu Block

****

1. Thêm một Block vào BlockChain (Hàm AddBlock)

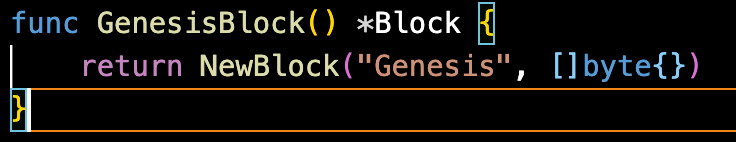
* bc: Con trỏ kiểu BlockChain (Dùng để truy vấn đến blockchain của chúng ta)
* data: Chứa thông tin dữ liệu của Block được thêm vào.
* prevBlock: Truy vấn đến Block cuối cùng của Blockchain hiện tại “len(bc.blocks)-1 là index cuối của mảng blocks (Blockchain) của chúng ta trước khi thêm Block mới vào”
* new: Gọi hàm NewBlock để tạo Block mới chi tiết được giải thích ở I->a->3
* bc.blocks: Gắn thêm Block mới (new) vào cuối mảng blocks thông qua hàm append



1. Tạo Block đầu tiên cho 1 BlockChain (hàm GenesisBlock)

*Để thêm block mới vào blockchain chúng ta cần có block liền trước nó. Vậy lúc blockchain chưa có block nào thì sao? Trong bất kì blockchain nào cũng phải có block đầu tiên này, và nó được gọi là genesis block*

* Hàm GenesisBlock trả về một con trỏ kiểu Block (ở đây là trả về địa chỉ của con trỏ trỏ đến Block Genesis )
* Return NewBlock(...) trả ra một Block có dữ liệu data là “Genesis” và hash của Block trước đó là rỗng



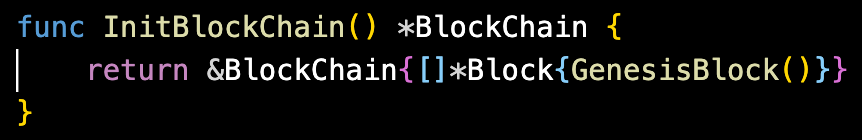
1. Khởi tạo 1 BlockChain (hàm InitBlockChain)

* Hàm InitBlockChain trả về một con trỏ kiểu BlockChain (ở đây là trả về địa chỉ của con trỏ trỏ đến BlockChain)
* return &BlockChain{[]\*Block{GenesisBlock()}}: Tạo và lấy ra địa chỉ một con trỏ BlockChain đó với dữ liệu của con trỏ BlockChain này có được đầu tiên bằng việc tạo ra một mảng có phần tử là các con trỏ kiểu Block với phần tử đầu tiên chính là Block Genesis)

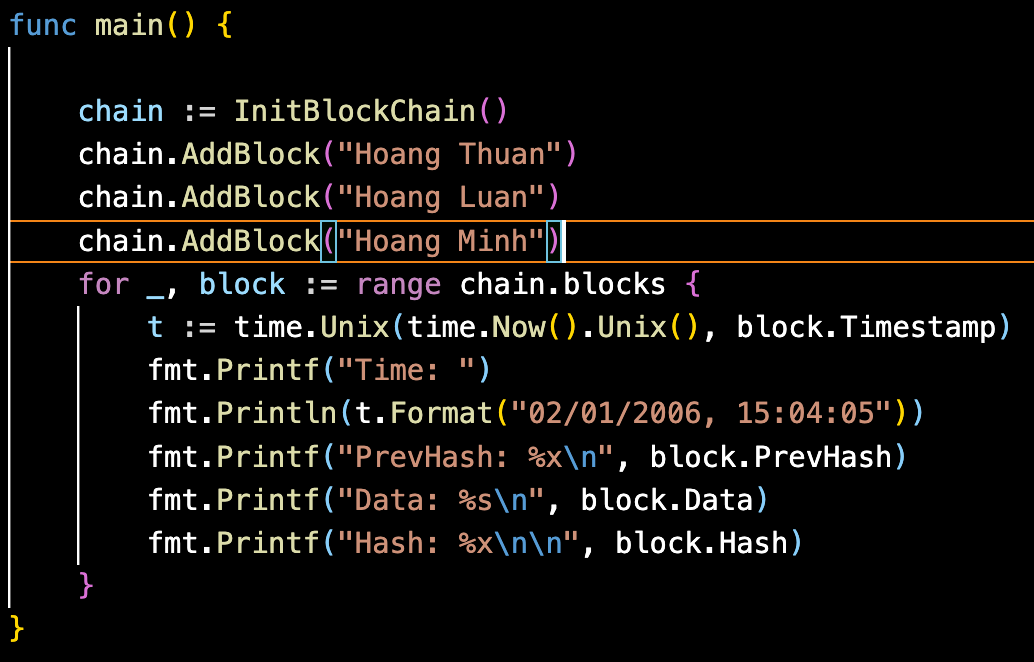
[]\*Block

BlockChain

Genesis Block



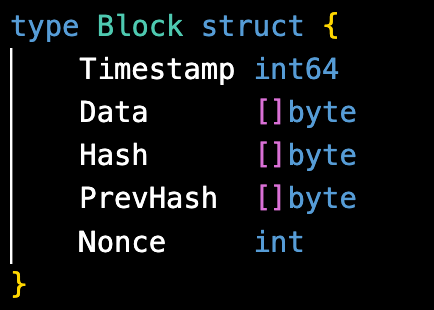
1. **Kiểm tra sự hoạt động của BlockChain chúng ta tạo ra**

****

****

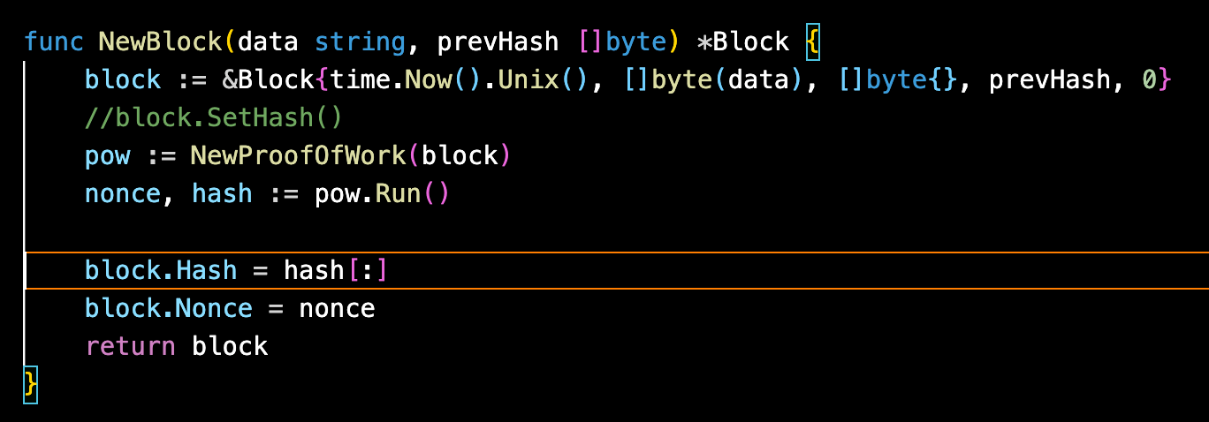
1. **Cơ chế đồng thuận dựa trên thuật toán Poof of work**
2. **Cấu trúc lại BlockChain ở phần 1**
3. Thêm biến Nonce vào cấu trúc Block

Tác dụng của biến Nonce là nó được thêm vào để tăng độ khó cho việc tạo ra một hash hợp lệ (hiểu nôm na là vậy)



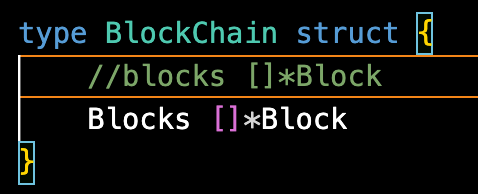
1. Thay đổi hàm NewBlock

Trong phần 1 chúng ta có hàm SetHash của Block, giờ chúng ta thay thế bằng hash sinh ra từ proof-of-work này

****

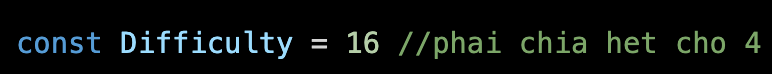
1. Chú ý quy tắc đặt tên thuộc tính trong class

Để Blockchain của chúng ta có thể public thì phải viết hoa chữ cái đầu tiên (tham khảo tại : https://vngeeks.com/go-structure/)



1. **Cấu trúc lại BlockChain ở phần 1**
2. Thiết lập độ khó cho việc tìm hash hợp lệ

* Chú ý là độ khó này phải là một số có giá trị chia hết cho 4 ví dụ set độ khó =16 thì số số 0 nằm trước hash hợp lệ được tạo ra sẽ là 16/4=4

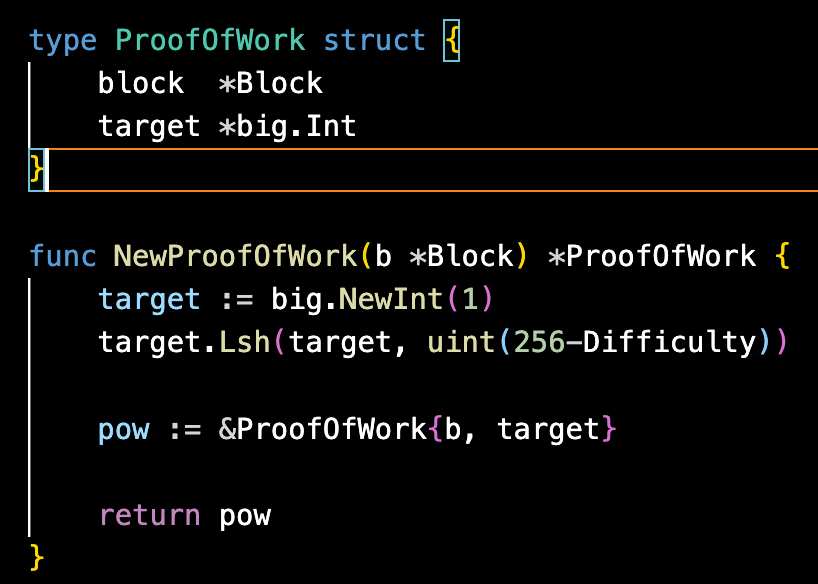
****

1. Cấu trúc của ProofOfWor và hàm tạo ra ProofOfWord

Struct ProofOfWor chứa con trỏ tới 1 block và một target. Biến target này chính là độ khó.

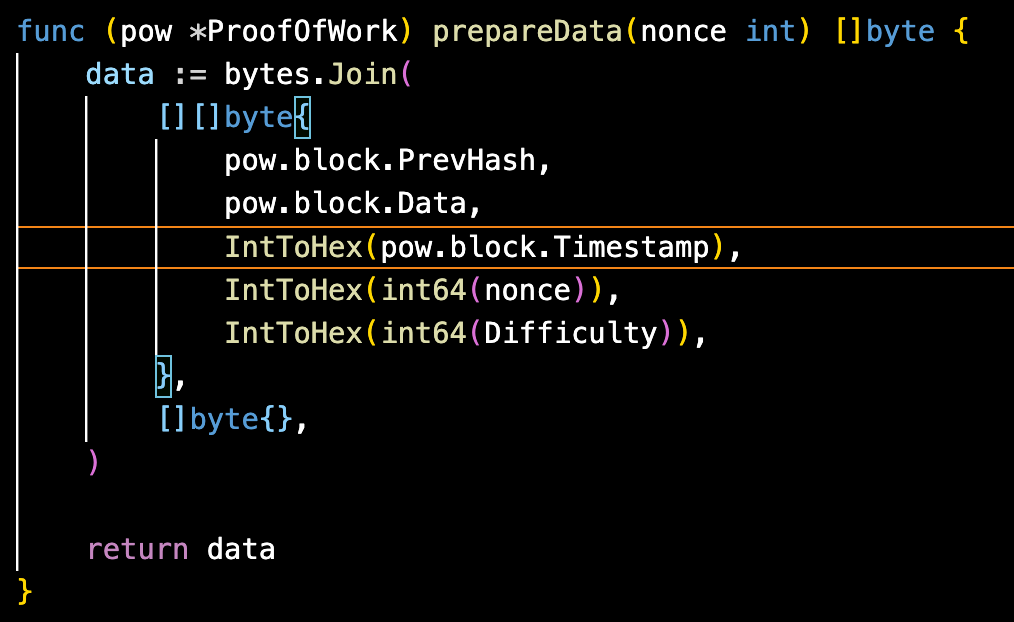
Chúng ta sử dụng big.Int để so sánh hash với target:

* Chuyển hash thành big.Int và kiểm tra xem hash có bé hơn target hay không.
* Trong hàm NewProofOfWork chúng ta khởi tạo target bằng 1 và shift left (dịch trái) 256 - targetBits bits. 256 là độ dài của một hash sinh ra bởi thuật toán băm SHA-256.



1. Tạo ra biến data như info như ở phần 1

* Nhập phần Data, PrevHash, timestamp, targetBits,nonce từ các mảng kiểu byte 1 chiều riêng lẻ vào trong một mảng kiểu byte 2 chiều và Join nó với 1 mảng byte rỗng



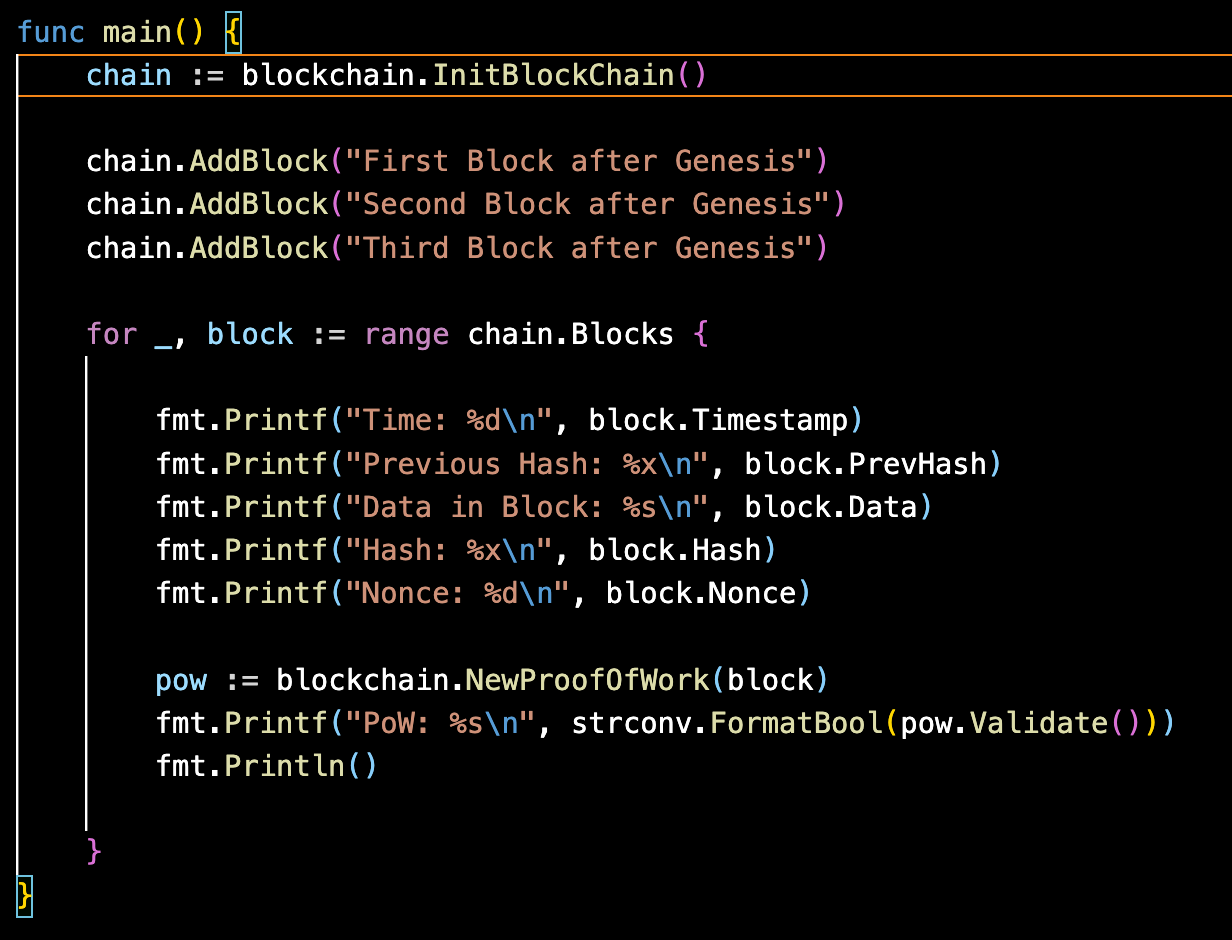
1. Tìm kiếm hash hợp lệ

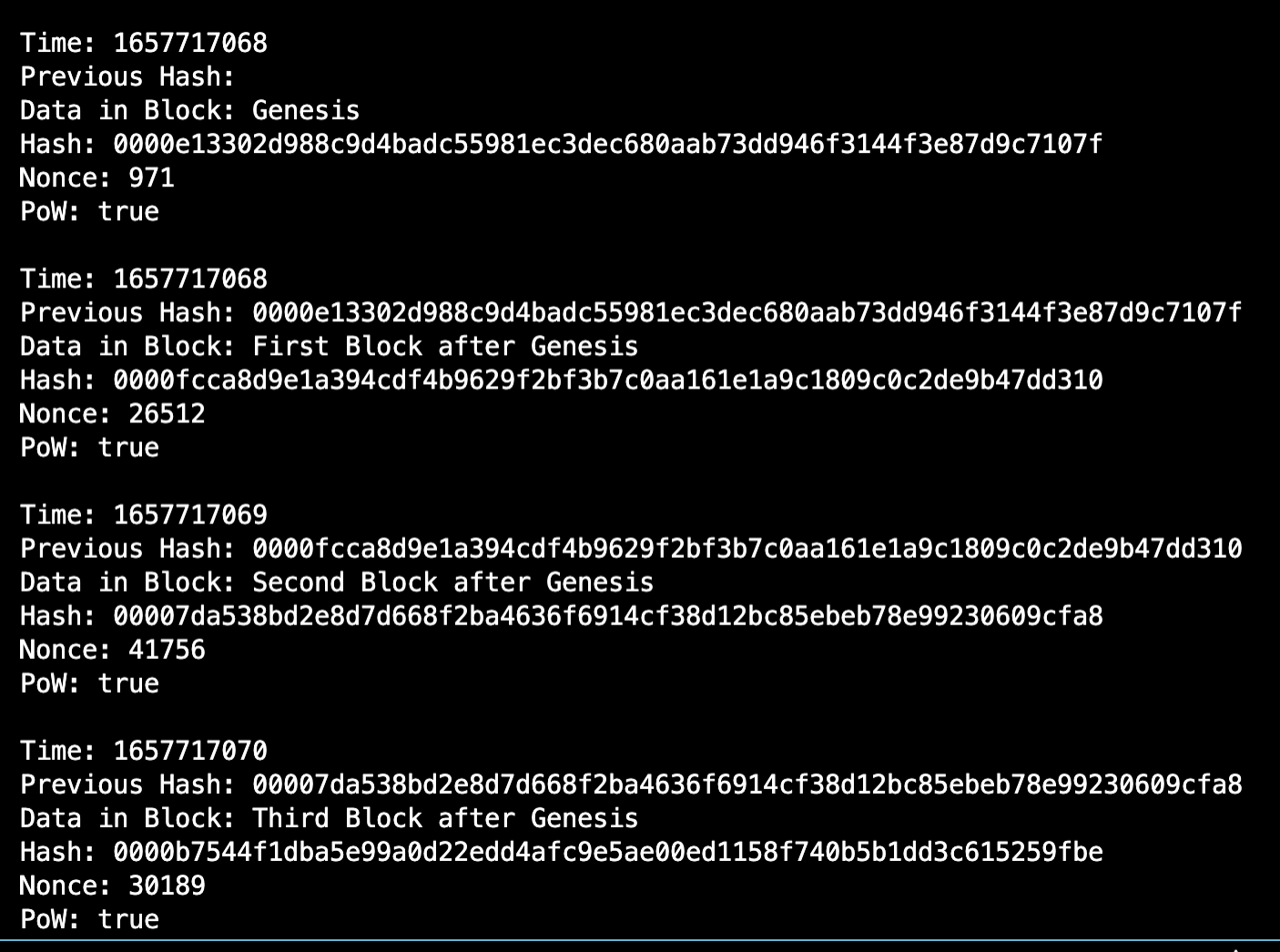
Chúng ta sẽ chạy một vòng lặp liên tục cho tới maxNonce (bằng math.MaxInt64). Thực tế thì chúng ta sẽ tìm ra hash thoả mãn sớm, không đến mức nonce phải tới giá trị math.MaxInt64 cực lớn này. Trong vòng lặp này chúng ta làm những việc sau:

* Chuẩn bị data
* Hash data với thuật toán SHA-256
* Chuyển hash sang big.Int
* So sánh con số nhận được với target



1. **Kiểm tra sự hoạt động của BlockChain chúng ta tạo ra**

****

****