SAMSUNG SW ACADEMY FOR YOUTH

對外秘

# 프로젝트 명세서

## 블록체인 간단 구현

對外秘

목차

# 1. 프로젝트 개요

간단한 블록체인 프로젝트를 구현해 봄으로써 블록체인에 대한 기본개념과 해시에 대한 이해를 넓혀보고자 합니다.

## 블록체인

> ***블록체인****(block chain)*은 관리 대상 데이터를 '블록'이라고 하는 소규모 데이터들이 P2P 방식을 기반으로 생성된 체인 형태의 연결고리 기반 분산 데이터 저장 환경에 저장하여 누구라도 임의로 수정할 수 없고 누구나 변경의 결과를 열람할 수 있는 분산 컴퓨팅 기술 기반의 원장 관리 기술이다. 이는 근본적으로 분산 데이터 저장기술의 한 형태로, 지속적으로 변경되는 데이터를 모든 참여 노드에 기록한 변경 리스트로서 분산 노드의 운영자에 의한 임의 조작이 불가능하도록 고안되었다. 블록체인 기술은 비트코인을 비롯한 대부분의 암호화폐 거래에 사용된다. 암호화폐의 거래과정은 탈중앙화된 전자장부에 쓰이기 때문에 블록체인 소프트웨어를 실행하는 많은 사용자들의 각 컴퓨터에서 서버가 운영되어, 중앙에 존재하는 은행 없이 개인 간의 자유로운 거래가 가능하다.

블록체인의 위키 정의를 보면 결코 쉽지 않은 기술처럼 보이며 모든 것이 한번에 이해되지도 않을 것 같습니다.

하지만 글자 그대로 **블록**과 **체인**은 조금만 노력하면 이해가 가능합니다.

### 블록과 체인 구현

블록체인에서 말하는 블록과 체인은 개발코드로 이해하려면 무엇을 말하는 것일까요?

- 블록 - 관리 대상 데이터
- 체인 - 연결고리

위키 정의를 다시 보며 의미를 추출해보면 데이터와 연결고리로 정의 되어 있습니다. 이걸 코드로 구현하려고 생각해본다면 **구조체(객체)** 값에 **링크드 리스트**로 엮은 모습과 비슷할 것 같습니다. 다만, 실제로는 포인터 참조가 아닌 해시값 참조로 연결되어 있습니다. 그래서 해시를 이해해야 합니다.

### 블록 해시

블록해시에서 해시는 해시함수를 의미합니다. 위키정의를 살펴보면,

해시함수란 임의의 길이의 데이터를 고정된 길이의 데이터로 매핑하는 함수이다.

쉬운 예로 나머지(modulas)를 들 수 있습니다. n 값의 나머지 m 을 구하면 0 ~ m-1 범위의 값이 구해집니다. n 을 1 로 줘도 되지만 엄청 큰 수를 준다고 해도 결과값은 0 ~ m-1 범위의 값일 뿐입니다. m 값만으로는 n 값을 정확히 되돌리지는 못합니다.

해시함수는 아래와 같은 특성이 있습니다.

1. 뭘 넣든 비슷한 길이의 알 수 없는 난수가 결과로 출력이 된다.
2. 글자가 한글자만 바뀌어도 완전히 다른 결과가 출력이 된다.
3. 출력값으로 입력값을 예측할 수 없다.
4. 같은 내용을 입력값으로 주면 결과값은 항상 같다.

블록에는 어떠한 데이터가 담겨 있을 것입니다. 이것을 해시하면 결국 고정된 길이의 해시값이 나오게 될 것입니다.

## 블록 구현

블록은 구조체로 표현할 수 있습니다. 그럼 구조체 안에는 어떠한 내용들을 넣을 수 있을까요? 가장 단순한 모델을 고려해본다면 아래와 같습니다.

1. 이전 블록의 해시값을 기억할 변수
2. 현재 블록에서 간직해야 할 데이터

이해를 돕기 위해 최소한의 모델을 고려해보면 위의 2 가지 변수는 반드시 있어야 될 것 같습니다.

## 블록체인 구현

블록을 생성하였다면 직전에 만든 블록과 관계를 맺어줘야 합니다. 그 관계는 해시값을 통해서 가질 수 있습니다. 이전의 블록 해시값을 현재의 블록이 기억을 해둡니다. 그리고 그렇게 생성된 블록의 해시값은 또 다음 블록이 기억을 할 것입니다. 이렇게 블록생성 마다 이런 과정을 반복적으로 해줌으로써 블록간의 관계가 맺어지게 되고 이를 블록체인이라고 합니다.

단 하나 예외적인 경우가 있다면 블럭을 처음 만들 때 입니다. 이때는 이전의 블록이 없기에 해시값도 없습니다. 이런 경우에는 empty 값(null 과는 다른) 혹은 random 값을 넣어주면 됩니다. 앞으로의 반복되어질 블록체인의 과정 속에 던져질 최초 씨앗이 되는 블록이기에 이 블록의 직전 블록이란 것은 의미가 없습니다.

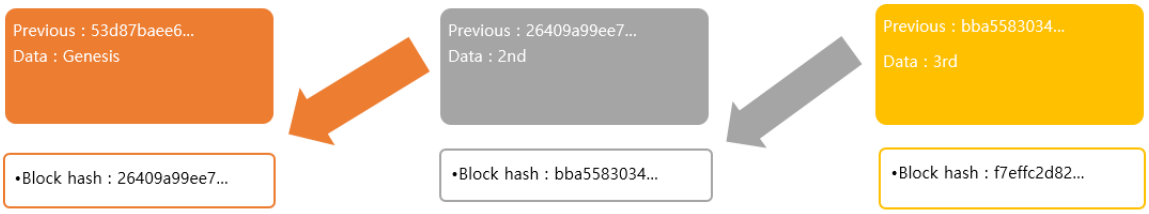
이렇게 최초로 만든 블록을 Genesis 블록이라고 합니다.

# 작업 증명(Proof Of Work)

위의 과정까지가 문자 그대로 블록체인을 구현하였다고 볼 수 있습니다. 너무 단순하여 실용성이 없을 뿐입니다. 이제 조금씩 이해의 범위를 넓혀 보겠습니다.

## 블록을 수정한다면?

이렇게 구현된 블록체인에서 총 3 개 의 블록을 생성하였다고 가정해보겠습니다.




최초의 블록해시 값(26409a99ee7...)은 2 번째 블록에서 기억하고 있습니다. 그런데 최초의 블록값이 1byte 라도 달라진다면 해시값은 완전히 달라질 것입니다. (해시의 특성 참조)

그래서 2 번째 블록에서 이 변경된 해시값을 previous 해시값으로 변경 한다면 3 번째 블록은 어떻게 될까요? 3 번째 블록은 2 번째 블록의 해시값(bba5583034...)을 기억하고 있었는데 2 번째 블록이 변경되면 해시값이 달라 질 것입니다.

이런 식으로 블록간에는 해시로 연결되어 있어서 이전의 블록을 변경한다는 것은 이후의 블록 모두가 변경되어야 한다는 것을 의미합니다. 이후의 블록 개수가 많으면 많아질수록 부담이 될 것입니다.

하지만 현대 컴퓨터는 무척 빠르기 때문에 이 정도는 크게 부담스러운 일이 아닐 것입니다. 그래서 블록을 만들 때 컴퓨터라도 빨리 만들지 못하게 만들어야 하겠습니다. 일종의 숙제(work)를 줘서 빠른 시간내로 만들지 못하게 할 것입니다.

## Nonce

위의 그림을 예시로 하여 컴퓨터에게 숙제를 주도록 하겠습니다.

위의 예에서 첫번째 블록 해시값은 "26409a99ee7..." 입니다. sha256 같은 해시함수를 사용하여 오래 걸리지 않아 구해질 것입니다. 그리고 2 번째 블록에서는 이 값을 보관한 상태로 해시값을 구할 것입니다. 그러면 "bba5583034..."값이 구해질 것입니다.

하지만 이제부터 해시값은 "0"으로 시작하는 해시값만 사용하려고 합니다. "07cb7d1168..." 이런 0 으로 시작하는 해시값을 구하려고 합니다. 해시값을 변경하려면 구조체 값이 달라져야 변경이 가능합니다(해시의 특성 참조).

기존의 구조체 변수들(previous 해시, data) 을 변경해서는 안될 것입니다. 그것은 반드시 보관해야 할 값들입니다. 여기서 다른 변수를 추가합니다. 그러면 전체적인 구조체 인스턴스 값도 달라지게 되니깐요.

그렇다고 해시값이 바로 '0'으로 시작하지는 않을 것입니다. 그래서 값을 조금씩 변경하면서 '0'으로 시작할 때까지 계속해서 변경을 가해보는 것입니다.

좀더 구체적으로는 정수형(integer) 변수를 추가하여 0 부터 값을 줍니다. 그런 후 해시값을 구해보고 조건이 맞지 않으면 1 씩 증가시켜 보는 것입니다. 이렇게 계속 변수값을 변경해서 해시를 구하다 보면 언젠가는 "0"으로 시작하는 해시가 나올 것입니다.

이런 목적으로 추가하는 변수를 nonce 라고 합니다.

| 해시넷의 논스 설명 인용 |
| --- |
| *논스(nonce)*는 비트코인의 창시자인 사토시 나카모토(Satoshi Nakamoto)가 쓴 비트코인 백서에 나오는 용어이다.[1]<br><br>블록체인은 다수의 거래내역을 모아 하나의 블록을 구성하고, 그 블록을 대표하는 해시값을 생성하여 다른 블록과 체인처럼 연결된다. 이 때, 블록을 대표하는 해시값인 블록해시를 생성하려면, 일정한 조건을 만족해야 한다. 그 일정한 조건이란, 블록 난이도에 따라 자동으로 설정된 '목표값'보다 더 작은 블록해시값을 찾아야 한다는 제약조건이다. 해시는 랜덤하게 생성되기 때문에, 수없이 많은 연산을 반복해서 미리 정해진 목표값 이하의 해시값이 나오도록 해야 한다. 이때 랜덤한 해시값을 생성할 수 있도록 매번 임시값을 사용해야 하는데, 그 임시값이 바로 논스이다. |

### Proof of Work(Pow)

nonce 를 올려가며 해시를 만들다 보면 언젠가는 "0"으로 시작하는 해시값이 나오게 될 것입니다. 만약 "0"이 한개가 아니라 "00" 처럼 2 개로 시작하는 해시를 구하라고 한다면 최종 해시값을 구하는 시간이 더 걸릴 것입니다. 이렇게 난이도를 올려가면서 해시를 구하는 시간을 조절할 수 있습니다. (시간적인 이야기는 이번 Daily Project 범위에서 제외하겠습니다) 이렇게 숙제(work)를 열심히 하여 과제물(nonce)을 제출하였다면 숙제검사는 어떻게 하면 될까요? 구해진 nonce 값으로 해시값을 그냥 구해보고 일치하면 nonce 랑 해시값이 옳다고 판단하면 됩니다.

이렇게 nonce 를 변경해가면서 최종 해시를 구하는 것을 작업증명이라고 합니다. 비트코인에서 하고 있는 방식입니다.

이제 앞의 블록 변경에 대한 이야기를 다시 해보겠습니다. 블록을 하나 수정한다면 이후의 블록은 해시를 다시 구해야 하며 nonce 를 다시 풀어야 된다는 말이 됩니다. 그렇다면 시간이 필요할 것입니다. 블록생성에 비용이 들어가게 된 셈입니다.

만약 블록체인이 실행되고 있다고 가정해보면, 중간의 블록을 악의적인 목적으로 수정을 하면 이후의 모든 블록을 변경해야 합니다. 그리고 매번 숙제를 다시 풀면서 구해야 할 것입니다.

## 정리

지금까지 블록과 체인을 나누어서 코드적인 측면에서 고려해보았고, 작업증명(PoW) 개념까지 살펴보았습니다. 작업증명은 암호화폐 중 가장 대표적인 비트코인이나 이더리움(차후 변경할 예정)에서 사용하는 방식입니다. 작업증명을 통하여 최종 블록을 생성하는 것을 블록을 마이닝(채굴)한다고 표현합니다.

물론 보상, 트랜잭션, 분산 등 블록체인에서 다루지 않은 개념들이 많이 있지만 지금부터의 내용을 토대로 하나씩 범위를 넓히면서 파악하신다면 블록체인이 어떻게 구현되어 있는지 이해에 도움이 될 것입니다.

# 2. 과제

지금까지 설명한 블록체인의 개념을 직접 구현합니다. 개발언어는 Java, Python 혹은 다른 언어라도 상관 없습니다. 혼선을 줄이기 위해 아래와 같은 제약을 둬서 범위를 좁히겠습니다.

1. Console 실행 환경 구현(print 출력만으로 충분함)
2. 블록에 대한 구조체(객체) 구현
3. 해시함수는 sha256 사용. sha256 사용법은 언어별로 검색해볼 것
4. Genesis 블록 생성
5. Genesis 블록 해시값을 다음 블록에서 기억
6. 이후 해시값은 앞에 "0"이 5 개인 경우만 통과. ex: "000009e3b554345....."
7. 블록이 생성될 때마다 블록 내용 console 출력.
8. 블록은 3 개 이상 생성해보세요.

**결과 예제**

구현 결과 예제

```
nonce: 0
data: Genesis Block
prevhash:
hash: 8500b59bb5271135cd9bcbf0afd693028d76df3b9c7da58d412b13fc8a8f9394

nonce: 423714
data: 2nd
prevhash: 8500b59bb5271135cd9bcbf0afd693028d76df3b9c7da58d412b13fc8a8f9394
hash: 00000f7813f5a2419b7d6113b8555ece466672469ba90bc3c9c0f123bc4d947c

nonce: 3281353
data: 3rd
prevhash: 00000f7813f5a2419b7d6113b8555ece466672469ba90bc3c9c0f123bc4d947c
hash: 000004f2e64b7239a062b1cca06fcb337804dede50069f5453d24c137bdd3f26
```

# 3. 심화 과제

블록체인 기술을 이해하는 데 있어 아주 중요한 것이 해시(hash)입니다. 기본 코드구현에서는 sha256 해시함수를 사용하였습니다. 이렇게 블록체인 외에도 해시함수는 많은 곳에서 응용이 가능합니다.

**해시함수를 활용한 사례를 조사하여 정리해보십시오.**

# 4. 산출물 제출

1. 코드제출 - 자신이 편한 언어를 사용하여 소스파일 제출

2. 심화과제 - Markdown Text 파일 제출