ДЗ. СмК Камень-ножницы-бумага

Форма: отчёт, код

Область: Ethereum

Цель работы: получить знания о внутреннем устройстве EVM, получить навыки программирования на языке Solidity, проектирования и создания смарт-контрактов.

Результат: написанный и размещенный в Goerli смарт-контракт.

Описание

Виртуальная машина Ethereum и смарт-контракты

В этой части лабораторной работы мы получим навыки работы с виртуальной машиной Ethereum. В блокчейне Ethereum есть возможность создавать программы, которые называются смарт-контракты. После написания смарт-контрактов идет этап компиляции и размещения в блокчейне.

Solidity

Solidity является основным языком написания смарт-контрактов для блокчейна Ethereum. Solidity это контрактно-ориентированный язык программирования императивного типа, который испытал влияние JavaScript, C++ и прочих объектных языков. Язык активно развивается и с некоторой периодичностью релизы ломают компиляторам обратную совместимость. Поэтому нужно внимательно следить за текущей версией языка и компилятора.

В языке есть свои особенности, такие как специфичные для EVM типы данных: address, block, tx, msg, …. Так и способы работы с памятью, их бывает несколько видов: storage, memory, stack, calldata.

EVM это замкнутая среда, которая не имеет доступа во внешний мир и для того, чтобы уведомить внешний мир о том, что в блокчейне был вызван смарт-контракт используется механизм Событий (event).

Этап проектирования

При реализации логики смарт-контракта используется несколько подходов к проектированию: использование абстрактных контрактов, интерфейсов, библиотек, наследования от сторонних контрактов и переиспользование уже существующих смарт-контрактов.
Чаще всего при проектировании смарт-контрактов используют подход по абстрагированию некоторой функциональности в отдельный смарт-контракт, от которого либо наследуется другой смарт-контракт, либо вызывается по адресу смарт-контракта.

При проектировании смарт-контрактов нужно учитывать ограничения на смарт-контракт, т.е. в нём нельзя запустить бесконечный цикл, или то что размер хранимых данных в смарт-контракте ограничен, размещенный код смарт-контракта в блокчейне невозможно изменить. Для решения подобных (но не всех) трудностей сообществом изобретены “обновляемые” смарт-контракты.

Этап компиляции

На этом этапе исходный код смарт-контракта преобразуется из человекочитаемого текста, в машиночитаемые инструкции. После компиляции смарт-контракта создаются следующие артефакты: байткод, ABI (Двоичный интерфейс приложений), метаинформация по компиляции.

Байткод представляет собой последовательный вызов машинных инструкций EVM, список всех инструкций найдете в https://www.ethervm.io.

При необходимости, компилятор Solidity можно настроить на несколько этапов оптимизации кода, который позволит уменьшить количество используемых смарт-контрактом ресурсов EVM.

Этап размещения

При размещении смарт-контракта в блокчейне, если это заложено логикой смарт-контракта, вызывается конструктор смарт-контракта, который может принимать аргументы или вызывать конструктор базового класса.

Конструктор вызывается только при размещении и может установить в смарт-контракт начальные значения.

Задание (8 баллов)

1. Спроектировать и написать смарт-контракт реализующий логику игры “Камень-ножницы-бумага”
2. Реализовать ролевую модель и использовать modifier (https://habr.com/ru/post/572004/)
3. Использовать схему commit-reveal для фиксации выбора игрока и обосновать почему она нужна
4. Добавить события в функции смарт-контракта
5. Разместить в тестовом блокчейне Ethereum
6. Опубликовать исходный код в github + адрес смк в тестнете
7. Отправить ссылку didexBot

Условие

отсутствуют

Контрольные вопросы (10 баллов)

1. В чём особенность смарт-контрактов без конструктора?
2. Чем отличается тип смарт-контракта library от типа contract?
3. Какие типы памяти существуют в EVM?
4. Зачем нужен ABI?
5. Зачем нужны вставки assembly в смарт-контакт?
6. Зачем нужен тип msg, tx, block?
7. Как можно задать случайное значение в смарт-контракт?

Полезные ссылки

1. Contracts — Solidity 0.8.9 documentation
2. Remix
3. OpenZeppelin/awesome-openzeppelin: Blockchain educational resources curated by the OpenZeppelin team

4. Ethereum Virtual Machine Opcodes
5. Software Engineering Techniques - Ethereum Smart Contract Best Practices
6. Creating Upgradeable Contracts From Solidity