Лабораторная работа 3

Основы Solidity. Модификаторы, события

Деплой смарт-контракта в тестовую сеть Ropsten

Теоретический блок

Типы видимости функций и глобальных переменных

- Функции могут быть указаны, как *external, public, internal или private*.
- Значение по умолчанию — public.
- Для глобальных переменных external невозможен, и по умолчанию указан internal.
- Для глобальных переменных как правило применяется модификатор private, а их изменение как правило осуществляется через функции.
- Отличия external и public: External функция не может быть вызвана изнутри.
- Документация: https://solidity.readthedocs.io/en/v0.5.12/contracts.html#visibility-and-getters

Модификаторы

- Модификаторы — это наследуемые свойства контрактов, и они могут быть переопределены в контрактах-наследниках.
- Модификаторы могут, например, автоматически проверять состояние до выполнения функции.
- Можно применять несколько модификаторов к одной функции. В таком случае они будут обработаны в представленном порядке.
- Документация: https://solidity.readthedocs.io/en/v0.5.12/contracts.html#function-modifiers

События

- События позволяют удобно использовать EVM журнал (лог), который в свою очередь можно использовать для обратных вызовов Javascript в пользовательском интерфейсе децентрализованного приложения.
- Пользовательские интерфейсы могут прослушивать события, которые запускаются в блокчейне.
- Слушатель события получает аргументы **from, to и amount**, что позволяет легко отслеживать транзакции.
- Пример
  Объявлении события:
  event Deposit(address indexed _from, uint _value);
  Использование:
  emit Deposit(msg.sender, msg.value);
- Документация: https://solidity.readthedocs.io/en/v0.5.12/contracts.html#events

Константы

- Доступны только для чтения.
- Ключевое слово constant.
- Должны быть инициализированы значением, которое постоянно и не зависит от времени компиляции.
- Поддерживаются только value types и string.
- Документация: https://solidity.readthedocs.io/en/v0.5.12/contracts.html#constant-state-variables

fallback-функция

- Смарт-контракт может иметь только одну функцию без названия. Эта функция не может иметь аргументы и не может что-либо возвращать.
- Она выполняется по вызову контракта, если ни одна из других функций не соответствует указанному идентификатору функции (или если данные не были предоставлены вообще).
- Она вызывается при передаче смарт-контракту какого-либо количества эфиров. В ней прописываются все возможные проверки.
- Вдобавок, чтобы получить эфир, fallback-функция должна быть отмечена payable. Если такой функции нет, то **контракт не может получить эфир** через регулярные транзакции.
- Несмотря на то, что функция возврата не может иметь аргументы, все равно можно использовать msg.data для извлечения полезной информаци, поставляемой с вызовом.
- Контракты, которые получают эфир напрямую (без вызова функции, а с использованием send или transfer) и не объявляют fallback-функцию, генерируют исключение и отправляют эфир обратно.
- НО! Контракт без fallback-функции может получить эфир в качестве предназначения selfdestruct. Контракт не может реагировать на такие передачи эфира, и, таким образом, не может отменить их.
- *function() external payable { }*

Функция **selfdesctruct**

- Деактивирует смарт-контракт.
- Все эфиры будут перенесены на адрес, переданный параметром в данную функцию.
- Такая функция «стоит» 5000 газа. Для сравнения: обычный перевод эфиров стоит примерно 2100 газа.

Изменения, которые были добавлены в новой версии Solidity 0.5.0

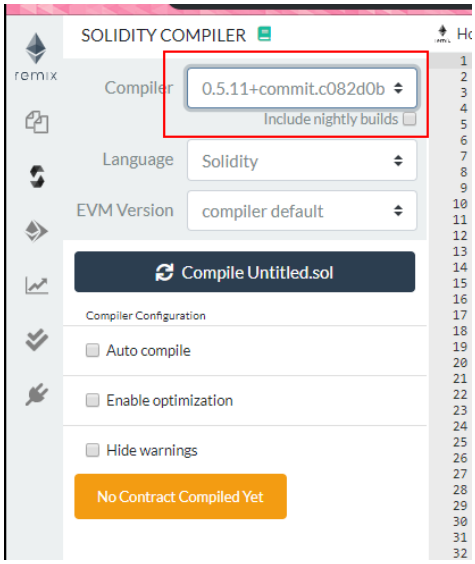
https://solidity.readthedocs.io/en/v0.5.12/050-breaking-changes.html

Практический блок

1. Заменить версию Solidity на 0.5.11 и добавить в уже существующий смарт-контракт для аукциона, разработанный в предыдущей лабораторной работе, следующие элементы:

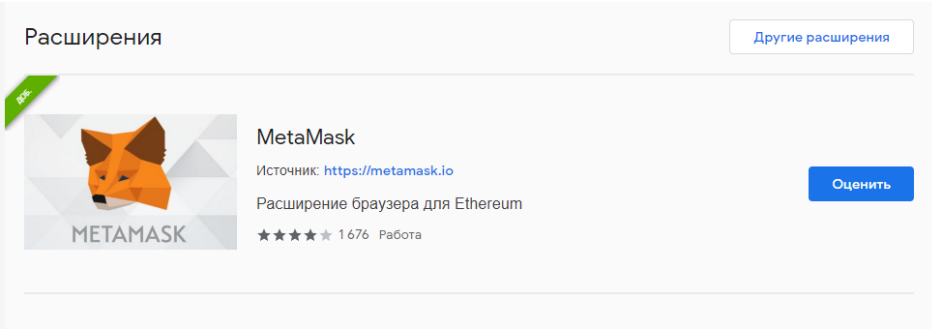
a) видимость функций и переменных состояния (public, private, external, internal). Внимательно продумайте видимость переменных состояния!
b) возвращаемое значение для функции создания нового аукциона (она должна возвращать id аукциона),
c) где необходимо – модификатор payable,
d) изменить присваивание адреса,
e) где необходимо – указатели на расположение данных в памяти (memory, storage, calldata)
f) модификатор, предоставляющий доступ только владельцу смарт-контракта
g) события при добавлении новой ставки и при переводе денег
h) fallback-функцию, которая позволит смарт-контракту получать эфиры
i) функцию kill, которая деактивирует смарт-контракт и перечислит все оставшиеся эфиры на адрес владельца

2. В IDE Remix скомпилировать смарт-контракт с версией компилятора 0.5.11+.



3. Необходимо установить в браузер расширение Metamask. Именно через Metamask контракт будет деплоиться в сеть Эфириума.

Найдите Metamask среди расширений Вашего браузера и установите.

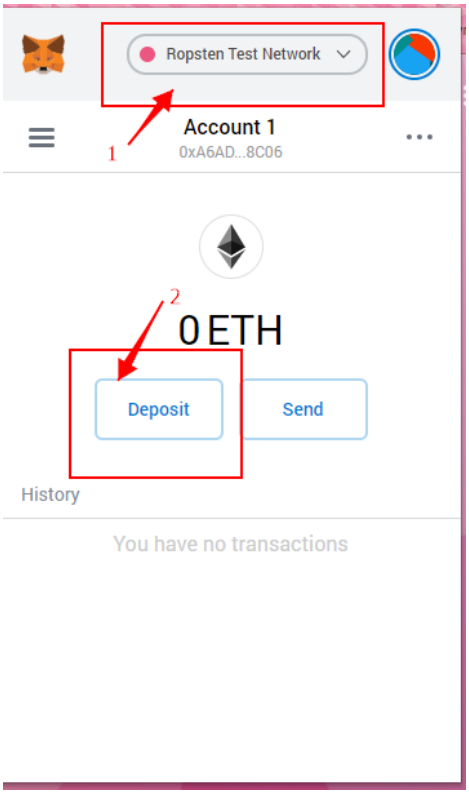


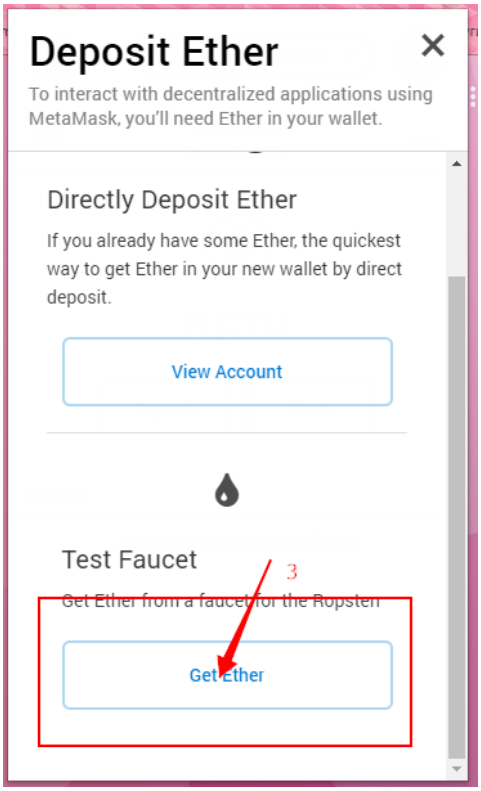
При установке необходимо будет задать пароль. ЗАПОМНИТЕ ЕГО!! Он будет использоваться для следующих лабораторных работ.

4. Получение тестовых эфиров.

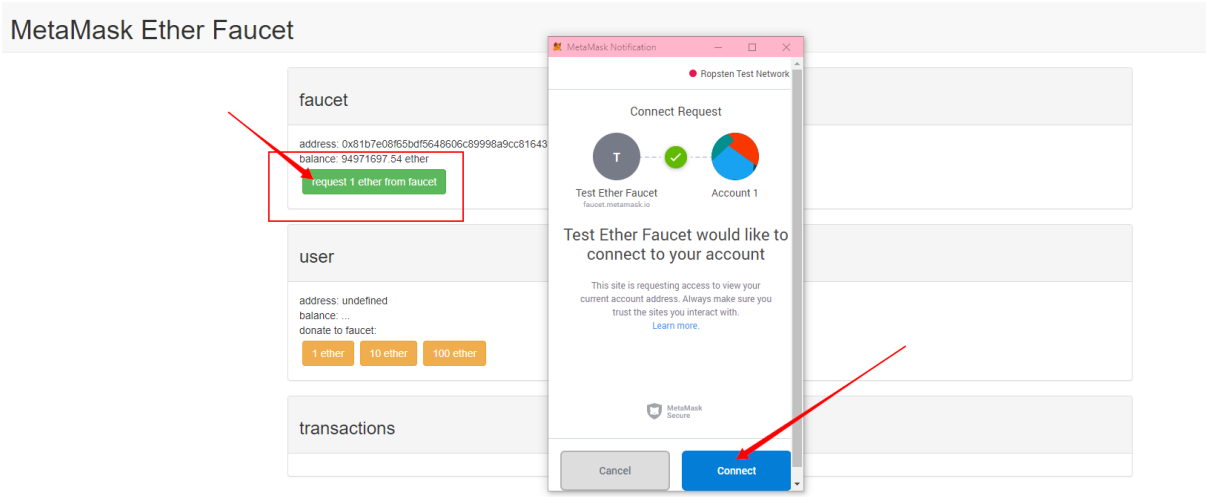
После входа в метамаск в левом верхнем углу выберите Ropsten Test Network. Это тестовая сеть Ropsten. Теперь именно в эту сеть будут деплоиться контракты.

Чтобы получить некоторое количество эфира для использования в тестовой сети Ropsten, требуется совершить следующие шаги:

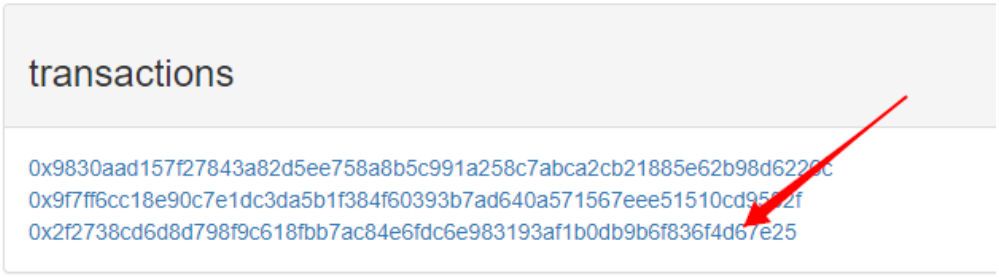




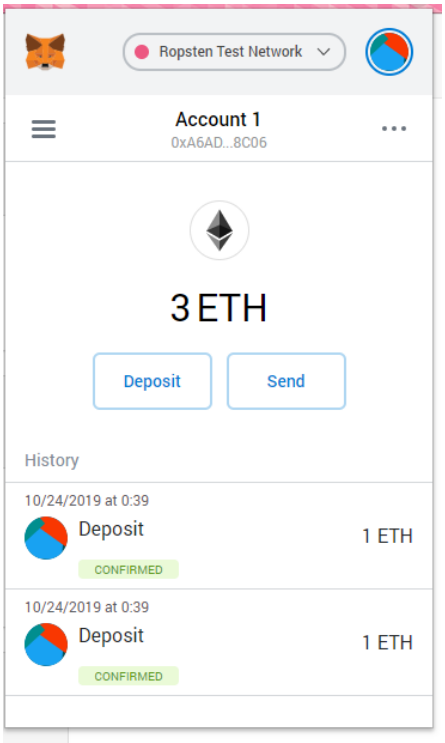
Откроется веб-сайт https://faucet.metamask.io/. Запросите один эфир на свой адрес.



Дождитесь, когда транзакции будут вмайнены в сеть. (Данные о транзакции и e1 состоянии можно попросить кликнув на её хэш).



После чего проверьте баланс на аккаунте.

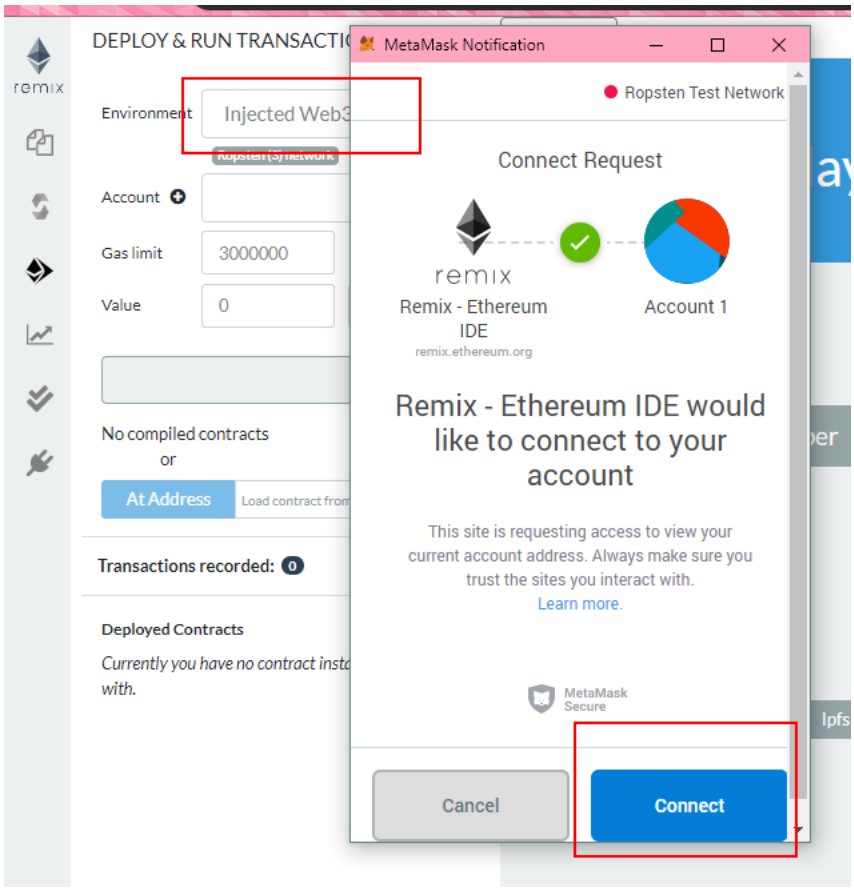


5. Деплой контракта в Ethereum через Remix.

Перейдите на вкладку “Deploy & Run transaction”.

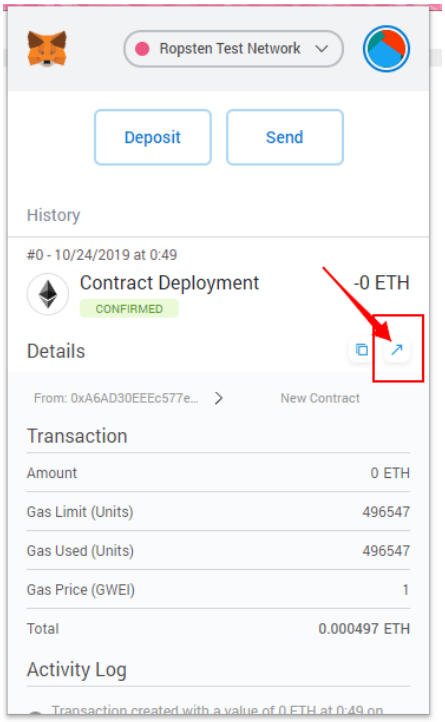
Выставить следующие параметры:

- Environment —“Injected Web3”;
- Account —автоматически подключится аккаунт из metamask;
- Gas limit и Value изменять не нужно.

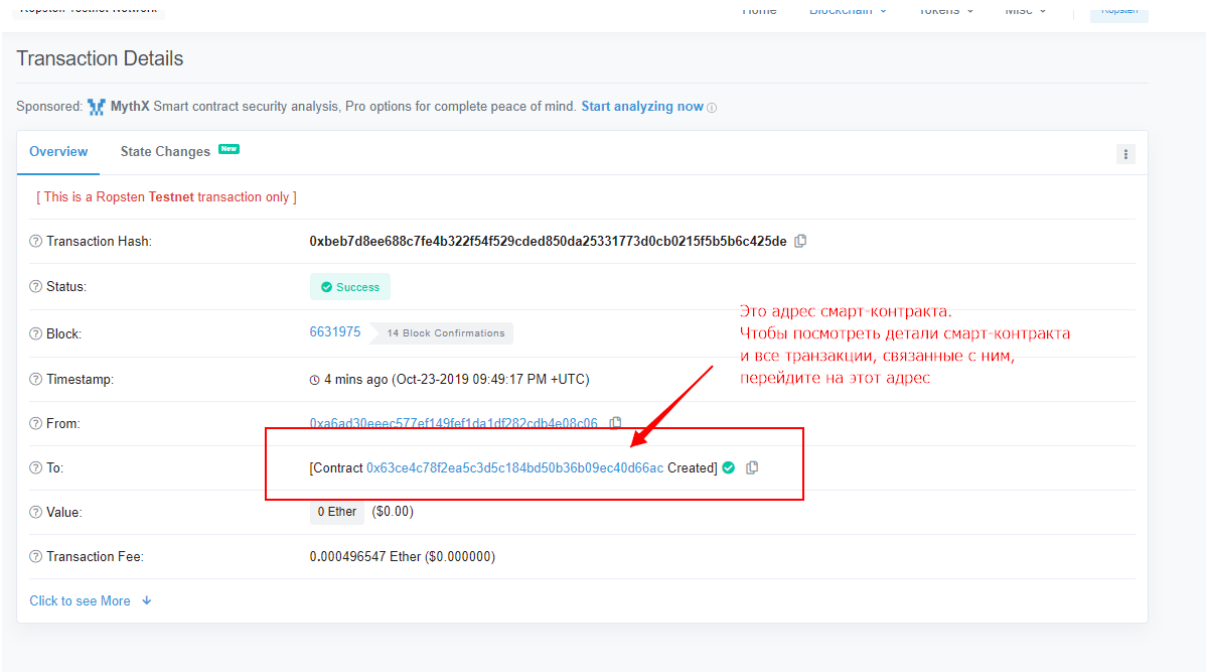


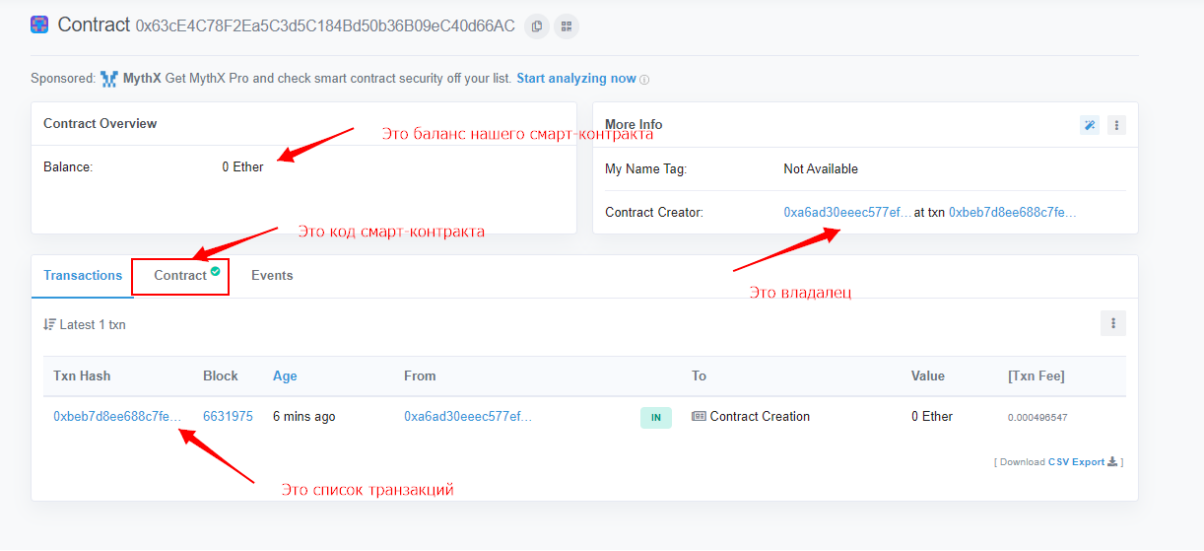
После выставленных параметров нажать “Deploy”. Подтвердить транзакцию в Metamask и дождаться, когда транзакция будет вмайнена.

Детали транзакции можно будет посмотреть перейдя на Etherescan.io, как показано на скриншоте ниже:



В отчёт добавить адрес смарт-контракта!!!!





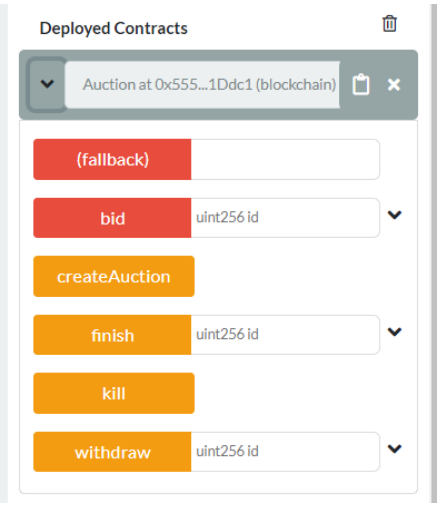
6. Протестируем смарт-контракт!

Наконец мы сможем испробовать его на практике!

Перейдем для этого обратно в IDE Remix. На вкладке Deploy & Run найдите раздел

Deployed Contracts и найдите последнюю версию задеплоенного смарт-контракта. При нажатии вам будут доступны все функции данного смарт-контракта.

В нашем случае это будут функции создания аукциона, ставка, завершения аукциона, деактивации смарт-контракта:

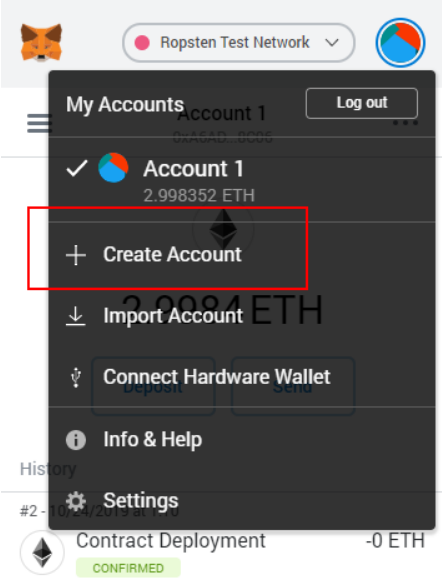


Самостоятельно протестируйте по следующим сценариям свой смарт-контракт:

1) **Создать 2 аукциона**
2) **В аукционе 1 сделать 2 ставки**
3) **Завершить аукцион 1**
4) **В аукционе 2 сделать 3 ставки**
5) **Завершить аукцион 2**

6) **Проверить, что все эфиры дошли до не выигравших участников**
7) **Проверить fallback-функцию**

Для реализации данного сценария вам понадобится создать ещё один аккаунт в сети Ropsten. Это можно сделать следующим образом:



Переключаясь между аккаунтами, можно делать ставки.

!!!!!!!!! В отчёте обязательно должен быть представлен скрин всех транзакций из etherscan.io, а также **адрес** задеплоенного смарт-контракта, чтобы все транзакции можно было увидеть через Etherscan.io

Контрольные вопросы

1. Чем отличаются модификаторы external и public?
2. Какие типы данных применимы к константам?
3. Можно ли применить несколько модификаторов к одной функции?
4. Как передать данные в fallback-функцию?
5. Что будет, если отправить эфир на смарт-контракт, в котором не реализована fallback-функция?
6. Какие параметры мы передаем в функцию selfdesctruct?
7. В каком случаем мы используем тип памяти memory?
8. Какие данные доступны слушателю события?

Задание

1) Изучить теоретический материал

2) Выполнить практическую часть лабораторной работы

3) Ответить на контрольные вопросы

4) Оформить **отчёт**

5) Прикрепить **файл смарт-контракта** с расширением .sol **!!!!**

Защита лабораторной работы 3: отчёт и файл с кодом смарт-контракта должны быть отправлены на портал.

Общие требования к отчёту

Отчёт должен содержать следующие элементы:

1) номер и название лабораторной работы,

2) ФИО студента, группу и курс

3) выполненные задания практического блока: скриншоты с etherscan.io, metamask и Remix и **!!!!!адрес смарт-контракта**

4) краткие ответы на контрольные вопросы.

В случае обнаружения плагиата (в том числе и в ответах на контрольные вопросы) отчёт не будет принят преподавателем!