 # Аудит смарт контрактов BitWork.

Статус - завершен.

Код контрактов проверен на предмет программных закладок и критических ошибок способных привести к потере денег инвесторами.

Некоторые контракты используют методы библиотеки **SafeMath** для безопасных вычислений.

 # 1. Контракт CourseContract

Заявленный функционал контракта **CourseContract** - продажа обучающих курсов за криптовалюту (ETH).

Контракт наследует функционал контрактов **Adminable** и **Ownable**.

Функционал контракта **Adminable** реализует ограничение доступа к функциям контракта посредством модификатора **onlyAdminAndOwner**.

Модификатор **onlyAdminAndOwner** ограничивает вызов функций ролями **owner** либо **admin**. Установка роли **admin** осуществляется через функцию **setAdmin**, вызов которой ограничен ролью **owner**, посредством модификатора **onlyOwner**, реализованного в контракте **Ownable**.

Функционал контракта **Ownable** реализует ограничение доступа к функциям контракта посредством модификатора **onlyOwner**.

Модификатор **onlyOwner** ограничивает вызов функции ролью **owner**. Роль **owner** устанавливается при деплое контракта в значение адреса, с которого происходит деплой. Установка нового значения **owner** осуществляется посредством вызова функции **transferOwnership** через параметр address **newOwner**, значение **newOwner** не может быть нулевым.

 # Переменные:

1. name:
   название;
   (по умолчанию устанавливается на "BitWork Course Sale Conrtact”);

   Примечание: возможная опечатка в слове “Conrtact”

2. Ethusd:
   текущая стоимость 1 eth к usd;
   (устанавливается при деплое контракта);
3. firstCoursePrice:
   стоимость первого курса в usd;
   (по умолчанию устанавливается значение “250”);
4. secondCoursePrice:
   стоимость второго курса в usd;
   (по умолчанию устанавливается значение “500”);
5. firstCoursePriceWei :
   цена первого курса в wei;
6. secondCoursePriceWei:
   цена второго курса в wei;
7. minFirstCoursePriceWei:
   минимальная цена к оплате за первый курс (90% от цены);
8. minSecondCoursePriceWei:
   минимальная цена к оплате за второй курс (110% от цены);
9. maxFirstCoursePriceWei:
   максимальная цена к оплате за первый курс (110% от цены);
10. maxSecondCoursePriceWei:
    максимальная цена к оплате за второй курс (110% от цены);
11. totalPayedSumWei:
    оплаченная стоимость;
12. courseLevelOneBonus:
    бонус в процентах к стоимости курса для спонсора первого уровня;
    (по умолчанию устанавливается значение “20”);
13. courseLevelTwoBonus:
    бонус в процентах к стоимости курса для спонсора второго уровня;

(по умолчанию устанавливается значение “10”);

14. courseLevelThreeBonus:
    бонус в процентах к стоимости курса для спонсора третьего уровня;
    (по умолчанию устанавливается значение “10”);
15. courseLevelFourBonus:
    бонус в процентах к стоимости курса для спонсора четвертого уровня;
    (по умолчанию устанавливается значение “5”);

16. courseLevelFiveBonus:
    бонус в процентах к стоимости курса для спонсора пятого уровня;
    (по умолчанию устанавливается значение “5”);
17. coursePaymentsAreStopped:
    подтверждение активности принятия оплат; (по умолчанию устанавливается значение “false”);

*Примечание: С учетом того что курс установленный владельцем (**ethusd**) является публичной переменной, стоит быть готовым к тому что покупатели будут отправлять легко рассчитываемые 91% от цены.*

 # Ивенты:

1. FirstCourseBought:
   покупка первого курса;
2. SecondCourseBought:
   покупка второго курса;

 # Основной фукционал:

Конструктор контракта (function **CourseContract**) выполняется единоразово при создании контракта и принимает один параметр – курс ETH к USD (допустим 400, в дальнейшем владелец контракта может изменять курс при помощи функции **setETHPrice**). Конструктор проверяет курс на корректность (не должен быть равен нулю), записывает владельца контракта как спонсора, создает instance контракта **FundContract** и запускает функцию **pricesRecalculation**.

Функция **pricesRecalculation** осуществляет пересчет переменных содержащих стоимость первого (**firstCoursePriceWei**) и второго (**secondCoursePriceWei**) курсов из usd в wei.

Привязка уже опубликованного контракта **FundContract** осуществляется с помощью функции **setFundContract**, доступной только владельцу контракта. Также при вызове данной функции значение заданного курса (**ethusd**) передается в **FundContract**.

*Проблема: две функции противоречат друг другу: создание instance of the **FundContract** в конструкторе **CourseContract** и функция **setFundContract** привязывающая опубликованный **FundContract** – если будет осуществляться привязка контракта через функцию **setFundContract**, то создание instance можно убрать из конструктора для экономии газа.*

Контракт содержит функцию, которая выполняется автоматически при поступлении на него адрес эфиров в количестве, превышающем значение переменной **minFirstCoursePriceWei**, а так же при условии, что значение переменной **coursePaymentsAreStopped** = false. Первоначальное значение переменной **coursePaymentsAreStopped** устанавливается в false при вызове конструктора контракта **CourseContract** (т.е. при его деплое в сети).

Значение переменной **coursePaymentsAreStopped** может быть изменено в true только вызовом функции **stopCoursePayments**, доступной только для роли owner. Значение переменной **coursePaymentsAreStopped** может быть изменено в false только вызовом функции **stopCoursePayments**, доступной только для роли owner. Расшифровка: приостановить прием оплаты за курсы и возобновить его может только владелец контракта в любой момент.

1. Если сумма перечисленных на адрес контракта эфиров меньше либо равна значению переменной **maxFirstCoursePriceWei**, осуществляется вызов функции реализующей покупку курса: **courseBought** с параметром в виде количества переведенных средств, а так же генерируется событие покупки первого курса - **FirstCourseBought**.

2. Если сумма перечисленных на адрес контракта эфиров меньше либо равна значению переменной **minSecondCoursePriceWei**, разница между количеством присланных средств и

значением переменной **firstCoursePriceWei** выплачивается отправителю, осуществляется вызов функции реализующей покупку курса: **courseBought** с параметром в виде цены первого курса: **firstCoursePriceWei**, а так же генерируется событие покупки первого курса - **FirstCourseBought**.

3. Если сумма перечисленных на адрес контракта эфиров меньше либо равна значению переменной **maxSecondCoursePriceWei**, осуществляется вызов функции реализующей покупку курса: **courseBought** с параметром в виде количества переведенных средств, а так же генерируется событие покупки второго курса - **SecondCourseBought**.
4. Если сумма перечисленных на адрес контракта эфиров больше значения переменной **maxSecondCoursePriceWei**, разница между количеством присланных средств и значением переменной **secondCoursePriceWei** выплачивается отправителю, осуществляется вызов функции реализующей покупку курса: **courseBought**, а так же генерируется событие **SecondCourseBought**.

– **Функция courseBought:**

Вызывается только из других функций связанных контрактов.
Принимает два параметра: адрес покупателя (**buyer**), сумму платежа (именно в веях, а не эфирах: **realPayedPrice**).
Выполняет следующее:

Примечание: спонсор = реферер.

1. Записывает спонсора покупателя (**sponsors[buyer]**) как спонсора первого уровня (sponsor1)
2. Устанавливает переменную выплаченных бонусов (**bonusesPayed**) равной 0.
3. Инициализирует временную переменную **bonusAmount**
4. Если спонсора первого уровня нет, то записывает сумму платежа на рассмотрение к администратору (**pending**), для спонсоров остальных уровней просто записывает владельца как спонсора

Примечание: в 200 строке кода можно изменить sponsors[buyer] на sponsor1, для этого и создавалась переменная.

5. Если спонсор первого уровня – владелец контракта, переводит сумму платежа с контракта на адрес владельца.
6. Рассчитывает и отправляет бонусы для спонсора первого уровня **bonusAmount**, а также сохраняет в mapping **bonusesPayed** количество выплаченных бонусов

7. Проделывает данный алгоритм для оставшихся 4 уровней спонсоров.

8. Переводит сумму платежа за вычетом выплаченных бонусов на счет владельца контракта.

Проблема: если спонсором какого либо уровня окажется владелец контракта то функция будет пытаться перевести ему полную сумму платежа дважды: в исполнении conditional statement и в конце функции.

– **Функция setSponsorInfo:**

Вызывается только администратором или владельцем.
Принимает два параметра: адрес нового участника (**newMember**) и адрес устанавливаемого для него спонсора (**sponsor**).
Выполняет следующее:

1. Проверяет устанавливаемого спонсора на наличие спонсора у него.
2. Если устанавливаемого спонсора нет то спонсором устанавливается владелец контракта: соответствующие бонусы будут перенаправлены владельцу.
3. Устанавливает заданного спонсора новому участнику
4. Заново запускает функцию **courseBought** с параметрами адреса нового участника и суммой задержанной в **pending**[**newMember**]
5. Запускает функцию **newSponsorInfo** в **fundContract**, чтобы синхронизировать спонсорскую систему в двух контрактах.

– **Функция testSetSponsorInfo:**

Вызывается только администратором или владельцем.
Принимает два параметра: адрес нового участника (**newMember**) и адрес устанавливаемого для него спонсора (**sponsor**).
Выполняет следующее:
Запускает функцию **newSponsorInfo** только в **fundContract**

- **Функция getSponsor:**

  Принимает один параметр: адрес участника
  Не требует отправки транзакции и соответственно газа
  Выполняет следующее:
  Возвращает адрес спонсора первого уровня.

- **Фунция setETHPrice:**

  Доступна только владельцу
  Принимает один аргумент - **_price** - новый курс **ethusd**
  Выполняет следующее:

  1. проверяет что новый курс больше 0
  2. Устанавливает новый курс
  3. Устанавливает новый курс в **fundContract**
  4. Запускает перерасчет цен (**pricesRecalculation**)

- **Функция setFirstCoursePrice:**

  Доступна только владельцу
  Принимает один аргумент (**_newPrice**)
  Выполняет следующее:
  1. Проверяет что новая цена больше 0 но меньше 5000
  2. Меняет цену первого курса на новую
  3. Запускает перерасчет цен (**pricesRecalculation**)

- **Функция setSecondCoursePrice:**

  Доступна только владельцу
  Принимает один аргумент (**_newPrice**)

Выполняет следующее:

1. Проверяет что новая цена больше 0 но меньше 5000
2. Меняет цену второго курса на новую
3. Запускает перерасчет цен (**pricesRecalculation**)

- **Функция stopCoursePayments:**

Доступна только владельцу;

Устанавливает **coursePaymentsAreStopped** равным true. Это означает, что пользователи больше не могут посылать эфир на контракт.

- **Функция resumeCoursePayments:**

Доступна только владельцу;

Устанавливает **coursePaymentsAreStopped** равным **false**. Это означает, что пользователи вновь могут посылать эфир на контракт.

2. FundContract

Заявленный функционал контракта **FundContract** – прием депозитов в ETH с выплатой простых процентов в течение срока депозита и выплатой основного тела по истечению срока.

Контракт наследует функционал контрактов **Adminable** и **Ownable**.

Функционал контракта **Adminable** реализует ограничение доступа к функциям контракта посредством модификатора **onlyAdminAndOwner**.

Модификатор **onlyAdminAndOwner** ограничивает вызов функций ролями **owner** либо **admin**. Установка роли **admin** осуществляется через функцию **setAdmin**, вызов которой ограничен ролью **owner**, посредством модификатора **onlyOwner**, реализованного в контракте **Ownable**.

Функционал контракта **Ownable** реализует ограничение доступа к функциям контракта посредством модификатора **onlyOwner**.

Модификатор **onlyOwner** ограничивает вызов функции ролью **owner**. Роль **owner** устанавливается при деплое контракта в значение адреса, с которого происходит деплой. Установка нового значения **owner** осуществляется посредством вызова функции **transferOwnership** через параметр address **newOwner**, значение **newOwner** не может быть нулевым.

Переменные:

1. name = "BitWork Fund Conrtact":
   название;

   Примечание: возможная опечатка в слове Contract

2. CourseContract public courseContract:
   контракт **courseContract;**
3. ethusd:
   курс ETH / USD;
4. mapping (address => uint) public pending:
   суммы на ожидании для каждого адреса;
5. fundOneInvestorPersent = 50:
   50% - накопительная сумма процентов по итогам всего срока депозита по первому тарифу;
6. fundOneSponsorOnePersent = 4:
   дополнительные проценты выплачиваемые спонсору первого уровня в тарифе для 6

месяцев;

7. fundOneSponsorTwoPersent = 3:
   дополнительные проценты выплачиваемые спонсору второго уровня в тарифе для 6 месяцев;
8. fundOneSponsorThreePersent = 3:
   дополнительные проценты выплачиваемые спонсору третьего уровня в тарифе для 6 месяцев;
9. fundOneSponsorFourPersent = 0:
   дополнительные проценты выплачиваемые спонсору четвертого уровня в тарифе для 6 месяцев;
10. fundTwoInvestorPersent = 100:
    100% - накопительная сумма процентов по итогам всего срока депозита по второму тарифу;

Возможная избыточность функционала: так как второй тариф экономически менее выгоден, при условии, что можо вложить два раза подряд на 6 месяцев и прибыль составит 125%, пользователи не будут к нему обращаться.

11. fundTwoSponsorOnePersent = 8:
    дополнительные проценты выплачиваемые спонсору первого уровня в тарифе для 12 месяцев;
12. fundTwoSponsorTwoPersent = 6:
    дополнительные проценты выплачиваемые спонсору второго уровня в тарифе для 12 месяцев;
13. fundTwoSponsorThreePersent = 3:
    дополнительные проценты выплачиваемые спонсору третьего уровня в тарифе для 12 месяцев;
14. fundTwoSponsorFourPersent = 3:
    дополнительные проценты выплачиваемые спонсору четвертого уровня в тарифе для 12 месяцев;

Примечание: у некоторых переменных одинаковые значения, следовательно их можно объединить для экономии, так как функции по регулированию бонусной программы отсутствует.

15. totalPaymentsUSD = 0:
    итоговая выплата USD;

16. totalPaymentsWei = 0:
    итоговая выплата WEI;
17. totalInvestmentsUSD = 0:
    итоговые инвестиции USD;

18. totalInvestmentsWei = 0:
    итоговые инвестиции WEI;
19. maxFirstFundAmountUSD = 10000:
    лимит USD для депозитов по первому тарифу;

Примечание: данное ограничение легко можно обойти, разбивая крупную сумму на меньшие, и совершая депозиты с разных анонимных адресов ethereum.

20. divisionInaccuracyProtection = 20:
    погрешность;
21. period = 30 days:
    временной отрезок для деления срока;
22. fundPaymentsAreStopped = false:
    состояние работы фонда (принимает ли платежи, false – да, true –нет);
23. testTimeShift = 0:
    тестовая переменная для сдвига значения now;

*Примечание: рекомендуется удалить для устранения возможности вмешательства владельца контракта во временные условия депозитов, так как в контракте присутствует функция **setTestTimeShift**, доступная владельцу и изменяющая сдвиг времени депозита по времени;*

24. Entry:
    struct содержащий в себе адрес участника (**person**), сумму депозита (**value**), итоговый процент к концу срока(**persent**), тариф (1 или 2) (**fund**), является ли инвестором участник (**true** or **false**) (**isInvestor**), время вклада(**startTime**), сколько уже снято в USD (**withdrawn**).

Примечание: грамматическая ошибка в параметре persent, правильно – percent.

 # Функционал:

При создании контракта все значения переменных устанавливаются как описано выше, а также создается массив из Entry под названием **deposits**;

Конструктор контракта (function **FundContract**) выполняется единоразово при создании контракта и принимает один параметр – адрес контракта **courseContract** - и привязывает этот контракт к **FundContract**.

- **Функция newDeposit:**

Принимает шесть параметров:

- адрес участника (**person**),
- сумму депозита (**value**),
- итоговый процент к концу срока(**persent**)
- тариф (1 или 2) (**fund**),
- является ли инвестором участник (**true** or **false**) (**isInvestor**),
- время вклада(**startTime**)

Выполняет следующее:

1.Создает struct **newEntry** и включает его в массив **deposits**;

- **Fallback функция автоматически включаемая при отправке эфиров на контракт:**

Выполняет следующее:

1. Проверяет корректность установленного курса (**ethusd** должен быть выше 0)
2. Если отправитель эфира - владелец контракта либо админ, то рассчитывает сколько на контракте стало денег для выплат, и, если достаточно (с учетом погрешности), вызывает функцию **makePayments**();
3. В других случаях, проверяется что прием платажей не приостановлен (**fundPaymentsAreStopped**), сумма отправляется владельцу, сумма итоговых инвестиций пополняется;
4. Из контракта **courseContract** запрашивается информация о спонсоре

*Примечание: (избыточный функционал) эта функция есть в начале **createDepositRecords**() которая вызовется в пункте 6)*

5. Если спонсор отсутствует то сумма записывается в ожидание
6. Вызывается функция **createDepositRecords** с параметрами адреса отправителя и полученной суммой(цифрой а не реальными эфирами);

- **Функция createDepositRecords:**

Примечание: records во множественном числе, по смыслу должно быть в единственном

Вызывается только внутри контрактов.
Принимает два параметра: адрес инвестора, сумму инвестиций.
Выполняет следующее:

1. Из контракта **courseContract** запрашивается информация о спонсоре .
2. Если сумма платежа меньше максимума для первого тарифа, то:
   - Создается **struct** инвестора содержащий в себе всю необходимую информацию (путем вызова уже описанной функции **newDeposit** с шестью параметрами).
   - Проверяется наличие спонсора первого уровня, если такового нет, то им становится владелец контракта, далее для спонсора создается соответствующий **struct** (путем вызова уже описанной функции **newDeposit** с шестью параметрами).
3. Проверяется наличие спонсоров остальных уровней (по порядку) и проделывается тот же алгоритм.

Проблема: ошибка в коде (строка 467) инициализация и запрос спонсора третьего уровня должна быть на две строки выше

Проблема: делается проверка на спонсора четвертого уровня который по условиям не получит ничего

4. Если сумма платежа больше максимума для первого тарифа то проделывается описанный выше алгоритм для записи платежа по второму тарифу.

Проблема: для спонсоров 2, 3 и 4 уровня сделана такая же ошибка перемены мест строк, т.е. владелец устанавливается как спонсор при определенном условии, а потом эта переменная запрашивается из другого контракта, что делает предыдущее действие бессмысленным

Проблема: логика вышеописанной функции разделяет платежи по тарифам исходя из суммы, а не из предпочтения клиента, что делает невозможным вызвать эту функцию корректно в случае если клиент хочет вложить сумму больше максимума первого тарифа на срок 6 месяцев, и наоборот, если клиент хочет вложить маленькую сумму на 12 месяцев.

- **Функция makePayments:**

Вызывается только внутри контрактов.
Не принимает параметров.
Выполняет следующее:

1. Для каждой записи об инвестиции, платеже (массив **deposits**) рассчитывается сумма к выплате в данный момент (путем вызова функции **totalPersonPaymentsShouldBeDoneToTheMoment**), и если возможно производится выплата.

Проблема: ошибка в коде строка 515 функция totalPersonPa.... вызывается с пробелом между названием функции и скобками с параметрами

- **Функция getAmountToPayAtTheMoment**:

Принимает один параметр - время (**time**).
Доступна только админу и владельцу.
Не требует отправки транзакции и соответственно газа.
Возвращает сумму к оплате (именно цифру а не эфир).

Выполняет следующее:

1. Проверяет указанное время на корректность для данной функции (нельзя брать прошедший период, т.е. **time**>=**now**)
2. Для каждой записи об инвестиции, платеже (массив **deposits**) рассчитывается сумма к выплате в заданный момент (путем вызова функции **totalPersonPaymentsShouldBeDoneToTheMoment**)
3. Возвращает сумму к оплате (именно цифру, а не эфир)

– **Функция totalPersonPaymentsShouldBeDoneToTheMoment:**

Принимает два аргумента: порядковый номер (индекс) нужной записи в массиве **deposits**, нужное время (**time**).

Не требует отправки транзакции и соответственно газа.

Возвращает сумму к оплате (именно цифру, а не эфир).

Выполняет следующее:

1. Проверяет указанное время на корректность для данной функции (нельзя брать прошедший период, т.е. **time**>=**now**).
2. Для соответствующей записи из массива **deposits** рассчитывается сумма к выплате к нужному моменту по указанному в массиве тарифу.

Проблема: для первого фонда по истечению 6 месяцев сумма процентов равняется 100 (должно быть 50)

Проблема: для первого фонда ежемесячно начисляется 20 процентов (должно быть 10)

Проблема: в формуле итоговой суммы к выплате процент вычисляется дважды подряд

*Примечание: одна из переменных называется **monthes** - грамматическая ошибка, должно быть **months***

– **Функция getAmountToPayNow:**

Не принимает параметров.

Не требует отправки транзакции и соответственно газа.

Доступна только владельцу и админу.

Возвращает сумму к оплате (именно цифру а не эфир).

Выполняет следующее:

1. Вызывает функцию **getAmountToPayAtTheMoment** с параметром времени "Сейчас" (now).

- **Функция getAmountToPayTommorow:**

Не принимает параметров.
Не требует отправки транзакции и соответственно газа.
Доступна только владельцу и админу.
Возвращает сумму к оплате (именно цифру а не эфир).
Выполняет следующее:

1. Вызывает функцию **getAmountToPayAtTheMoment** с параметром времени "Сейчас + 1 дней".

- **Функция getAmountToPayNextWeek:**

Не принимает параметров.
Не требует отправки транзакции и соответственно газа.
Доступна только владельцу и админу.
Возвращает сумму к оплате (именно цифру а не эфир).
Выполняет следующее:

1. Вызывает функцию **getAmountToPayAtTheMoment** с параметром времени "Сейчас + 7 дней".

- **Функция getAmountToPayNextMonth:**

Не принимает параметров.
Не требует отправки транзакции и соответственно газа.
Доступна только владельцу и админу.
Возвращает сумму к оплате (именно цифру а не эфир).

Выполняет следующее:

1. Вызывает функцию **getAmountToPayAtTheMoment** с параметром времени "Сейчас + 30 дней".

- **Функция getContractBalanceWei:**

  Не принимает параметров.
  Не требует отправки транзакции и соответственно газа.
  Возвращает сумму доступную на балансе контракта в wei.

- **Функция getContractBalanceUSD:**

  Не принимает параметров.
  Не требует отправки транзакции и соответственно газа.
  Возвращает сумму доступную на балансе контракта в usd.

- **Функция withdraw:**

  Не принимает параметров.
  Доступна только владельцу.
  Выполняет следующее:
  1. Переводит весь доступный баланс контракта владельцу.

- **Функция getTotalInvestmentsUSD:**

  Не принимает параметров.
  Не требует отправки транзакции и соответственно газа.
  Доступна только владельцу и админу.
  Возвращает сумму вложенных средств в usd (цифру, не деньги).

- **Функция getTotalInvestmentsWei:**

  Не принимает параметров.
  Не требует отправки транзакции и соответственно газа.
  Доступна только владельцу и админу.
  Возвращает сумму вложенных средств в wei (цифру, не эфир).

- **Функция getTotalPaymentsUSD:**

  Не принимает параметров.
  Не требует отправки транзакции и соответственно газа.
  Доступна только владельцу и админу.
  Возвращает сумму выплаченных средств в usd (цифру, не деньги).

- **Функция getTotalPaymentsWei:**

  Не принимает параметров.
  Не требует отправки транзакции и соответственно газа.
  Доступна только владельцу и админу.
  Возвращает сумму выплаченных средств в wei (цифру, не эфир).

- **Функция setETHPrice:**

  Принимает один параметр: новая цена eth в долларах.
  Доступна для вызова только из контракта courseContract.

  Примечание: функцию можно было сделать internal.

  Изменяет цену eth.

  Проблема: функция не запускает никаких пересчетов, следовательно после обновления курса все значения usd сохраненные в контракте будут устаревшими.

- **Функция toWei:**

  Принимает один параметр: сумма usd (цифра).
  Не требует отправки транзакции и соответственно газа.
  Возвращает сумму в Wei (цифру).
  Переводит доллары в wei.

- **Функция toUSD:**

  Принимает один параметр: сумма wei (цифра).
  Не требует отправки транзакции и соответственно газа.
  Возвращает сумму в Usd (цифру).
  Переводит wei в доллары .

- **Функция newSponsorInfo:**

  Принимает один параметр: адрес нового участника.
  Доступна только для вызова из контракта **courseContract**

  *Примечание: можно было сделать **internal.***

  Выполняет следующее:

  1. Запускает функцию **createDepositRecords**, которая создает запись в массиве **deposits** для спонсора.

- **Функция stopFundPayments:**

  Не принимает параметров.
  Доступна только владельцу.
  Меняет значение **fundPaymentsAreStopped** на **true**, тем самым отключает возможность вносить пользователям эфир на счет контракта.

– **Функция resumeFundPayments:**

Не принимает параметров.

Доступна только владельцу.

Меняет значение **fundPaymentsAreStopped** на **false**, тем самым включает возможность вносить пользователям эфир на счет контракта.

– **Функция setCourseContract:**

Принимает один параметр: адрес нового контракта **courseContract.**

Доступна только владельцу.

Подключает новый контракт **courseContract.**

– **Функция setTestTimeShift:**

Принимает один параметр: новое добавочное значение в секундах (добавляется к now, в тестовых целях).

Доступна только владельцу.

Устанавливает новое добавочное значение.

Примечание: рекомендуется удалить данную функцию для устранения возможности вмешательства владельца контракта во временные условия депозитов;

#Предупреждение об ответственности

Этот аудит касается только исходных кодов смарт контрактов и не должен рассматриваться как одобрение платформы, команды или компании.

#Авторы

Аудит провёла команда **EthereumWorks.** По вопросам проведения аудитов и разработки смарт контрактов обращайтесь: Telegram - **@SlavaPoe**, Skype - **v.poskonin** (MousePo).