Лабораторна робота 4
ФУНКЦІЇ ТА МОДИФІКАТОРИ В SOLIDITY
ТИПИ ФУНКЦІЇ В SOLIDITY

Теоретичні відомості

Вступ

1. **External** – функції з таким типом можна викликати з інших контрактів, але не можно викликати з інших функції цього контракту

2. **Public** – має всі особливості типу external, але разом із цим функції цього типу можна викликати з інших функцій цього контракту

3. **Internal** – так звані внутрішні функції. Їх можна викликати з інших функцій цього контракту та контрактів нащадків, але не можна викликати з інших контрактів

4. **Private** – тип, ідентичний до internal, за винятком того, що ці функції недоступні для контрактів нащадків

5. **View** – додатковий модифікатор функції, який показує те, що ця функція не змінює стейт блокчейну, тобто тільки читає інформацію з блокчейна. За функції з цим модифікатором не потрібно платити комісію, за винятком ситуацій, коли ці функції викликаються в функції без модифікатора view.

6. **Pure** – також додатковий модифікатор, який має всі особливості view, крім того, що такі функції нічого не читають із блокейна.

Порядок написання функції

Оголошувати нову функцію можна за допомогою ключового слова function, після нього йде назва функції. Назви функції найчастіше йдуть в camelCase (наприклад, myNewFunction). Після назви йдуть параметри функції, потім її тип, додатковий модифікатор типу, якщо він є, далі

ключові слова override, virtual (якщо є), потім кастомні модифікатори, в кінці пишеться ключове слово returns та в круглих скобках що функція повертає. Останньої частини може не бути, якщо вона нічого не повертає (рис. 4.1).

```
function myFunc(uint256 someParameter) external view virtual returns (uint256, bytes32) {}
```

Рисунок 4.1 – Приклад порядку в об’явленні функції

Модифікатори в Solidity

Модифікатор в Solidity створюється за допомогою ключового слова modifier. Після ключового слова потрібно вказати назву модифікатора та відкрити фігурні скобки. Всередині модифікатора може бути будь-який код, але обов’язковою частиною є строка “_;”. Дана строка показує місце, де буде виконуватися код функції (рисунок 4.2).

```
modifier myModifier {
    // some code before function execution
    _;
    // some code after function execution
}
```

Рисунок 4.2 – Приклад структури модифікатора

.

Індивідуальне завдання

Є два контракти **FirstContract** та **SecondContract**. Потрібно реалізувати наступну логіку:

1. Для **FirstContract**

a. В функції **callSetInternalNumber** потрібно викликати функцію **setInternalNumber** на адресі **secondAddress_**

b. В функції **callSetFirstContract** потрібно викликати функцію **setFirstContract** на адресі **secondAddress_**

2. Для **SecondContract**

a. В конструкторі встановлювати змінну **owner** в адресу того, хто деплоє цей контракт

b. Створити модифікатор **onlyOwner**, який дозволяє тільки адресі в змінній **owner** викликати цю функцію

c. Функція **setFirstContract** повинна встановлювати змінну **firstContract** в ту адресу, яка приходить в параметрах функції. Ця функція повинна мати модифікатор **onlyOwner**

d. Створити модифікатор **onlyFirstContract** за аналогією до **onlyOwner**

e. Функція **setInternalNumber** повинна встановлювати змінну **_internalNumber** в те число, що приходить в параметрах функції. Ця функція повинна мати модифікатор **onlyFirstContract**

f. Функція **publicFunction** повинна просто встановлювати **publicNumber** в те число, що приходить в параметрах функції

g. Функція **externalFunction** повинна встановлювати змінну **publicNumber**, але **ЗАБОРОНЕНО** це робити напряму.

h. Функція **getInternalNumber** повинна повертати вміст змінної **_internalNumber**

3. В IDE Remix протестувати всі доступні функції та продемонструвати їх роботу у звіті.