Практичне завдання №2
Створення простого ERC20 токена

Мета завдання

- Зрозуміти архітектуру ERC20 токена.
- Навчитися писати смарт-контракти на мові Solidity з нуля.
- Освоїти роботу з Remix IDE для розробки та тестування смарт-контрактів.
- Вивчити процес налаштування MetaMask для роботи з Ethereum Sepolia Testnet.
- Дізнатися, як отримувати тестові ETH та розгортати смарт-контракти на Sepolia Testnet.
- Отримати навички взаємодії зі смарт-контрактами після їх розгортання.

Архітектура простого ERC20 токена

1. Основні компоненти ERC20 токена

ERC20 є стандартом для створення взаємозамінних токенів на платформі Ethereum. Основні компоненти ERC20 токена включають:

- **State Variables (Станові змінні):**
  - `name`: Назва токена (наприклад, "SimpleToken").
  - `symbol`: Символ токена (наприклад, "STK").
  - `decimals`: Кількість десяткових знаків (зазвичай 18).
  - `totalSupply`: Загальна кількість токенів, що існують.
  - `balanceOf`: Маппінг, який зберігає баланс кожного користувача.
  - `allowance`: Маппінг, який дозволяє одній адресі витрачати токени від імені іншої адреси.
- **Events (Події):**
  - `Transfer`: Випускається при передачі токенів від одного користувача до іншого.
  - `Approval`: Випускається при затвердженні витрат токенів однією адресою від імені іншої.
- **Functions (Функції):**
  - `constructor`: Ініціалізує токен, встановлює початкову кількість токенів та призначає їх творцю контракту.
  - `transfer`: Переводить токени від власника до іншої адреси.
  - `approve`: Дозволяє іншій адресі витрачати певну кількість токенів від імені власника.
  - `transferFrom`: Переводить токени від однієї адреси до іншої, використовуючи дозволені витрати.
  - `balanceOf`: Повертає баланс певної адреси.
  - `allowance`: Повертає кількість токенів, які одна адреса може витратити від імені іншої.

2. Детальний опис важливих частин контракту

2.1. State Variables (Станові змінні)

Назва та Символ токена:

```
string public name = "SimpleToken";
string public symbol = "STK";
uint8 public decimals = 18;
```

- Ці змінні визначають основні характеристики токена, такі як його назва, символ та кількість десяткових знаків, що впливає на точність обліку токенів.

Total Supply та Balance Of:

```
uint256 public totalSupply;
mapping(address => uint256) public balanceOf;
```

- `totalSupply` відображає загальну кількість створених токенів. Маппінг `balanceOf` зберігає баланс кожного користувача за адресою.

Allowance:

```
mapping(address => mapping(address => uint256)) public allowance;
```

- Цей маппінг дозволяє одній адресі (спендер) витрачати токени від імені іншої адреси (овнера), що є основою для функції `transferFrom`.

2.2. Events (Події)

Transfer:

```
event Transfer(address indexed from, address indexed to, uint256 value);
```

- Випускається кожного разу, коли відбувається передача токенів. Допомагає відстежувати трансакції на блокчейні.

Approval:

```
event Approval(address indexed owner, address indexed spender, uint256
value);
```

- Випускається при затвердженні дозволу на витрати токенів однією адресою від імені іншої.

2.3. Constructor (Конструктор)

Ініціалізація контракту:

```
constructor(uint256 _initialSupply) {
    totalSupply = _initialSupply * 10 ** uint256(decimals);
    balanceOf[msg.sender] = totalSupply;
    emit Transfer(address(0), msg.sender, totalSupply);
}
```

- Конструктор встановлює загальну кількість токенів, присвоює їх творцю контракту (`msg.sender`) та випускає подію `Transfer` від нульової адреси, що сигналізує про створення токенів.

2.4. Functions (Функції)

Transfer:
`function transfer(address _to, uint256 _value) public returns (bool success)`

- Функція для переведення токенів від власника до іншої адреси. Вона перевіряє наявність достатнього балансу, оновлює баланси відправника та отримувача та випускає подію `Transfer`.

Approve:
`function approve(address _spender, uint256 _value) public returns (bool success)`

- Дозволяє іншій адресі (`_spender`) витратити певну кількість токенів від імені власника. Випускає подію `Approval`.

TransferFrom:
`function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)`

- Дозволяє переведення токенів від однієї адреси до іншої з використанням дозволених витрат. Перевіряє наявність дозволу та достатнього балансу, оновлює відповідні значення та випускає подію `Transfer`.

3. Інструкції для виконання практичного завдання

Нижче наведено покрокові інструкції для студентів щодо створення, тестування та розгортання простого ERC20 токена без використання бібліотек.

3.1: Написання смарт-контракту ERC20 токена з нуля у Remix IDE

1. **Відкрийте Remix IDE:**
   - Перейдіть за посиланням https://remix.ethereum.org/
2. **Створіть новий файл:**
   - У панелі ліворуч натисніть на значок "Файли" (File Explorer).
   - Натисніть на значок "+", введіть назву файлу, наприклад, `SimpleToken.sol`.
3. **Напишіть код смарт-контракту:**
   - Використовуючи попередній опис архітектури, напишіть контракт, включаючи станові змінні, події та функції.
   - **Порада:** Почніть з визначення основних змінних (`name`, `symbol`, `decimals`, `totalSupply`), потім додайте маппінги (`balanceOf`, `allowance`), події (`Transfer`, `Approval`), конструктор та основні функції (`transfer`, `approve`, `transferFrom`).
4. **Компіляція контракту:**
   - Перейдіть на вкладку "Solidity Compiler" (іконка з компілятором) у Remix.
   - Переконайтесь, що обрана версія компілятора відповідає `pragma solidity ^0.8.0;` (наприклад, `0.8.0` або пізніша).
   - Натисніть кнопку "Compile SimpleToken.sol".
   - Переконайтесь, що компіляція пройшла успішно без помилок.

3.2: Отримання тестових ETH через Sepolia Faucet

1. **Встановіть MetaMask:**
    - Завантажте MetaMask з https://metamask.io/ та встановіть розширення для вашого браузера.
    - Створіть новий гаманець або імпортуйте існуючий, слідуючи інструкціям MetaMask.
2. **Налаштування MetaMask для Sepolia Testnet:**
    - Відкрийте MetaMask та натисніть на випадаючий список мереж у верхній частині вікна.
    - Виберіть "Sepolia Test Network". Якщо мережа відсутня, додайте її вручну:
        - Натисніть "Add Network".
        - Заповніть поля наступним чином:
            - **Network Name:** Sepolia Testnet
            - **New RPC URL:** `https://sepolia.infura.io/v3/YOUR_INFURA_PROJECT_ID` *(замініть `YOUR_INFURA_PROJECT_ID` на ваш Infura Project ID або використайте інший RPC URL для Sepolia)*
            - **Chain ID:** `11155111`
            - **Currency Symbol:** ETH
            - **Block Explorer URL:** `https://sepolia.etherscan.io`
        - Натисніть "Save".
3. **Отримання тестових ETH:**
    - Перейдіть на
        - https://cloud.google.com/application/web3/faucet/ethereum/sepolia
        - https://www.alchemy.com/faucets/ethereum-sepolia
    - Введіть вашу Sepolia адресу з MetaMask (копіюйте з MetaMask).
    - Натисніть кнопку для отримання тестового ETH.
    - Дочекайтесь підтвердження, що тестові ETH були надіслані на ваш гаманець.

3.3: Розгортання смарт-контракту на Sepolia Testnet

1. **Підключення Remix IDE до Sepolia через MetaMask:**
    - Переконайтесь, що MetaMask налаштований на Sepolia Testnet та має достатньо тестових ETH.
    - У Remix IDE перейдіть на вкладку "Deploy & Run Transactions" (іконка з гаманцем).
    - У випадаючому списку "Environment" виберіть "Injected Web3". Remix автоматично підключиться до вашого MetaMask.
    - Переконайтесь, що обрана мережа у MetaMask — Sepolia Testnet.
2. **Налаштування параметрів розгортання:**
    - У полі "Contract" виберіть `SimpleToken - SimpleToken.sol`.
    - У полі "Deploy" введіть початкову кількість токенів, наприклад, `1000000` (це буде 1,000,000 токенів з урахуванням 18 десяткових знаків).
3. **Розгортання контракту:**
    - Натисніть кнопку "Deploy".
    - MetaMask відкриється з підтвердженням транзакції. Перевірте деталі та натисніть "Confirm".

- Дочекайтесь підтвердження транзакції. Після успішного розгортання контракт з'явиться у списку "Deployed Contracts".

3.4: Взаємодія зі смарт-контрактом

1. **Перевірка балансу:**
   - У вкладці "Deployed Contracts" знайдіть ваш контракт `SimpleToken`.
   - Натисніть на функцію `balanceOf`.
   - Введіть вашу адресу (або залиште поле порожнім, щоб використовувати вашу адресу).
   - Натисніть кнопку "transact" для отримання балансу токенів.
2. **Передача токенів:**
   - Натисніть на функцію `transfer`.
   - Введіть адресу отримувача та кількість токенів для переведення (наприклад, `100`).
   - Натисніть кнопку "transact".
   - Підтвердіть транзакцію через MetaMask.
   - Дочекайтесь підтвердження транзакції.
   - Перевірте баланс отримувача за допомогою функції `balanceOf`.
3. **Затвердження дозволу та передача токенів від імені іншого користувача:**
   - **Затвердження дозволу:**
     - Натисніть на функцію `approve`.
     - Введіть адресу витрачальника (наприклад, адресу іншого вашого гаманця) та кількість токенів для витрати.
     - Натисніть кнопку "transact" та підтвердіть транзакцію через MetaMask.
   - **Передача токенів через `transferFrom`:**
     - Перейдіть до іншого гаманця, який має дозволи витрати.
     - Підключіть цей гаманець до Remix IDE (зазвичай через MetaMask).
     - Натисніть на функцію `transferFrom`.
     - Введіть адресу власника токенів, адресу отримувача та кількість токенів для переведення.
     - Натисніть кнопку "transact" та підтвердіть транзакцію через MetaMask.

Додаткове завдання: Публікація смарт-контракту на Sepolia Etherscan та перевірка його роботи

1. **Отримання адреси контракту:**
   - Після розгортання контракту у Remix IDE скопіюйте адресу вашого смарт-контракту. Вона доступна у списку "Deployed Contracts" або у MetaMask під час розгортання.
2. **Перевірка контракту на Sepolia Etherscan:**
   - Перейдіть на Sepolia Etherscan.
   - Вставте адресу вашого контракту у поле пошуку та натисніть Enter.
   - Якщо контракт ще не перевірено, виконайте наступні кроки для його перевірки:
     1. Натисніть кнопку "Verify and Publish".
     2. Виберіть версію компілятора Solidity, яка відповідає `pragma solidity ^0.8.0;` (наприклад, `v0.8.0+commit.c7dfd78e`).
     3. Виберіть оптимізацію (зазвичай `No` для простих контрактів).

4. Вставте вихідний код контракту у відповідне поле.
5. Натисніть кнопку "Verify and Publish".

- Після успішної перевірки ви зможете переглянути всі функції контракту та транзакції через Etherscan.

3. **Перевірка роботи контракту через Etherscan:**
   - На сторінці контракту у Etherscan ви зможете переглядати всі виклики функцій, транзакції передачі токенів та інші дії, пов'язані з вашим контрактом.
   - Ви також можете використовувати вкладку "Read Contract" та "Write Contract" для взаємодії з контрактом безпосередньо через Etherscan.