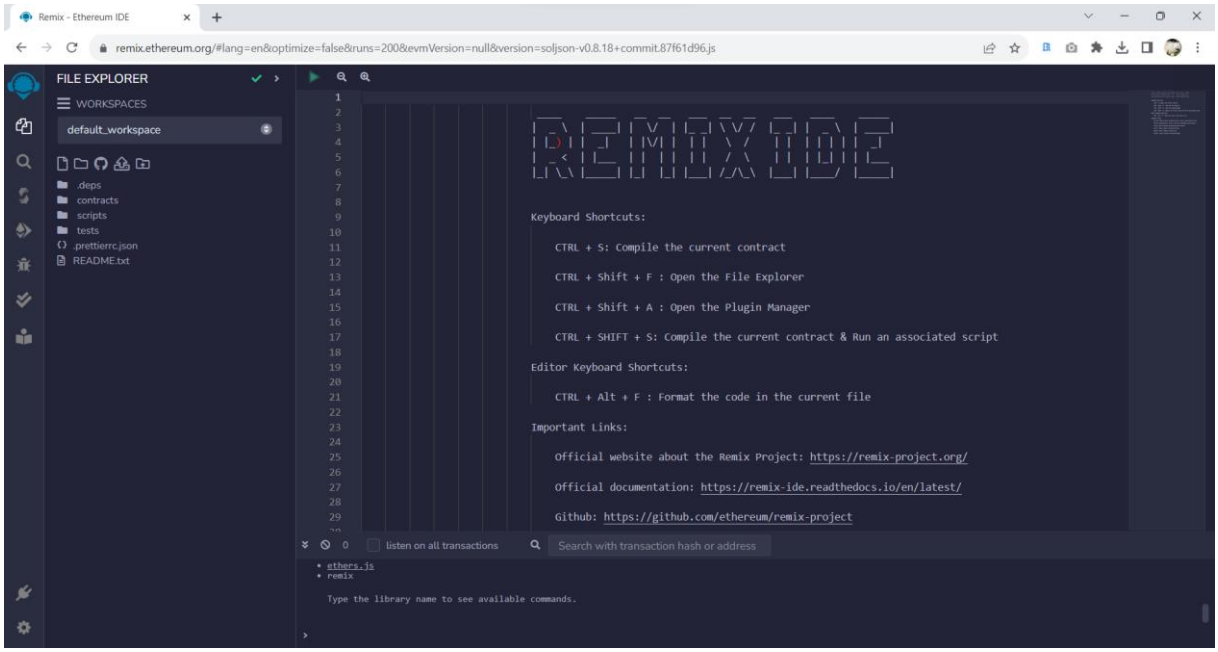
BUỔI 3: SOLIDITY (1)

1. Sử dụng công cụ online online Remix IDE

https://remix.ethereum.org

File sẽ có phần mở rộng **.sol**



2. HelloWorld

- Tạo một file HelloWorld.sol

- Nhập vào đoạn code:

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.8.2 <0.9.0;

contract HelloWorld {
    string private message = "Hello World";
    function getMessage() view public returns (string memory) {
        return message;
    }
}
```

- Thực hiện compile

- Thực hiện deploy xem kết quả

3. Cấu trúc của một file Sol

3.1. Pragma

```
pragma solidity ^0.8.2;
```

Xác định phiên bản trình biên dịch sử dụng.

^: trình biên dịch sử dụng cần phải từ 0.8.2 trở lên.

```
pragma solidity >=0.8.2 <0.9.0;
```

Trình biên dịch nên từ 0.8.2 đến phiên bản trước 0.9.0

3.2. License

Xác định License đối với phần mềm

```
// SPDX-License-Identifier: MIT
```

Câu hỏi: Tìm hiểu và mô tả 04 loại license mã nguồn mở phổ biến.

3.3. Comment

```
//Chú thích 1 dòng
/*
Chú thích nhiều dòng
*/
```

3.4. Cấu trúc của contract

Một contract bao gồm:

- Biến trạng theo
- Định nghĩa cấu trúc (struct)
- Định nghĩa modifier
- Khai báo sự kiện
- Định nghĩa liệt kê (enumeration)
- Định nghĩa hàm

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.8.2 <0.9.0;

//Chú thích 1 dòng
/*
Chú thích nhiều dòng
*/

contract HelloWorld {
    //Định nghĩa biết trạng thái
    string private message = "Hello World";
    int public stateIntVariable;
    address helloIdentifier;
    HelloStruct he;

    //Định nghĩa struct
```

```
    struct HelloStruct {
        string name;
        uint age;
    }

    //Khai báo modifier
    modifier onlyBy() {
        if (msg.sender == helloIdentifier) {
            _;
        }
    }

    //Khai báo event
    event ageRead(address, int);

    //Khai báo liệt kê
    enum gender {male, female}

    //Khai báo hàm
    function getMessage() view public returns (string memory) {
        return message;
    }
}
```

4. Kiểu giá trị

Phạm vi các biến

- internal: mặc định; có thể truy cập từ hợp đồng hiện tại hoặc từ hợp đồng kế thừa. Không thể được truy cập từ ngoài để sửa đổi nhưng có thể đọc được
- public: có thể truy cập từ bên ngoài; hàm getter tự động được tạo
- private: chỉ có thể truy cập từ bên trong.
- constant: biến không thể thay đổi; phải được gán giá trị khai khai báo.

Kiểu	Giá trị
bool	true/false
int/uint	Số nguyên không dấu, kích cỡ khác nhau
int8, int256	Kiểu số nguyên có dấu 8 bit, 256 bit
uint8, uint256	Kiểu số nguyên không dấu 8 bit, 256 bit
address	Địa chỉ 20 byte (160 bit)
…	

enum: kiểu người dùng định nghĩa.

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.8.2 <0.9.0;

contract ExampleType {
    function getIntNum() public pure returns (uint32) {
        uint32 x = 12;
        return x;
    }

    function getBool() public pure returns (bool) {
        bool x = false;
        return x;
    }

    function getByte() public pure returns (bytes1) {
        bytes1 x = "a";
        return x;
    }

    function getAddress() public pure returns (address) {
        address x = 0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48;
        return x;
    }
}
```

5. Phép toán

Tương tự như các ngôn ngữ khác: + - * / % ** ++ --

Solidity *không có kiểu float*

Yêu cầu: viết các hàm add, substract, multiply, divide

```
function add(uint256 x, uint256 y) public pure returns(uint256) {
        //Hàm trả về tổng x và y
    }

    function substract(uint256 x, uint256 y) public pure returns (uint256) {
        //Hàm trả về hiệu x và y
    }

    function multiply(uint256 x, uint256 y) public pure returns (uint256) {
        //Hàm trả về tích x nhân y
    }

    function divide(uint256 x, uint256 y) public pure returns (uint256) {
        //Hàm trả về thương x chia y
    }
```

Cho biết kết quả: substract(2, 4), divide(5, 2). Giải thích kết quả đạt được.

Xây dựng thêm các hàm lũy thừa, chia lấy phần dư

6. Các câu lệnh

```
if (điều kiện) {}
else if (đk) {}
else {}
```

```
while (đk) {}
```

```
for (uint i = 0; i < 10; i++) {}
```

```
function isOdd(uint256 x) public pure returns(bool) {
        if (x % 2 == 0) {
            return true;
        } else {
            return false;
        }
    }

    function addNumbers(uint256 n) public pure returns(uint256) {
        uint256 sum = 0;
        for (uint256 i = 0; i < n; i++) {
            sum += i;
        }
        return sum;
    }

    function multiplyNumbers() public pure returns(uint256) {
        uint256 i = 1;
        uint256 mul = i;
        while(mul < 1028) {
            mul *= i;
            i += 1;
        }
        return mul;
    }
```

Xây dựng hàm kiểm tra một số nguyên dương là một số nguyên tố hay không

```
function isPrime(uint256 number) public view returns (bool) {

    }
}
```

Xây dựng hàm tính dãy số Fibonacci 0, 1, 1, 2, 3, 5, …. (không dùng phương pháp đệ quy

```
function fibonacci(uint256 n) public view returns (uint256) {
        // your code here
    }
```

7. Mảng và chuỗi

```
 function useArrayForUint256(uint256[] calldata input) public pure returns
(uint256[] memory) {
        return input;
    }
```

Tìm hiểu calldata và memory là gì?

- Chỉ mục mảng bắt đầu là 0
- Số lượng phần tử của mảng: sử dụng length

Chuỗi (string)

- Hoạt động gần giống với mảng
- Không thể lấy ký tự trong chuỗi bằng chỉ mục
- Không có thuộc tính length

```
function helloUser(string calldata name) public pure returns (string memory) {
        return string.concat("hello ", name);
    }
```

Bài tập: hoàn thiện các hàm bên dưới

```
// SPDX-License-Identifier: MIT

pragma solidity >=0.8.2 <0.9.0;

contract Exercise {
    function threeFive(uint256 n) public pure returns (string memory) {
        //Nếu n chia hết cho 3 thì trả về "three"
        //Nếu n chia hết cho 5 thì trả về "five"
        //Nếu n chia hết cho 3 và 5 thì trả về "threefive"
        //Ngược lại trả về chuỗi rỗng
    }
```

```
    function sumArray(uint256[] calldata _arr) public pure returns (uint256) {
        //Trả về tổng mảng arr
    }

    function filterEven(uint256[] memory _arr) public pure returns (uint256[]
memory) {
        //trả về mảng chỉ gồm các số chẵn lấy ra từ mảng _arr
    }

    function isSortedArray(uint256[] calldata _arr) public pure returns (bool)
{
        //trả về true nếu mảng _arr đã được sắp xếp theo thứ tự tăng dần
        //ngược lại trả về false
    }

    function meanArray(uint256[] calldata _arr) public pure returns (uint256)
{
        //Trả về giá trị trung bình của mảng _arr
    }
}
```