LAB 6

LÊ CHÍ HUY – 197CT31347

# Demo storage to storage value type assignment

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

contract DemoStorageToStorageValueTypeAssignment {

    uint stateVar1 = 20;

    uint stateVar2 = 40;

    function getUInt() public returns (uint)

    {

        stateVar1 = stateVar2;

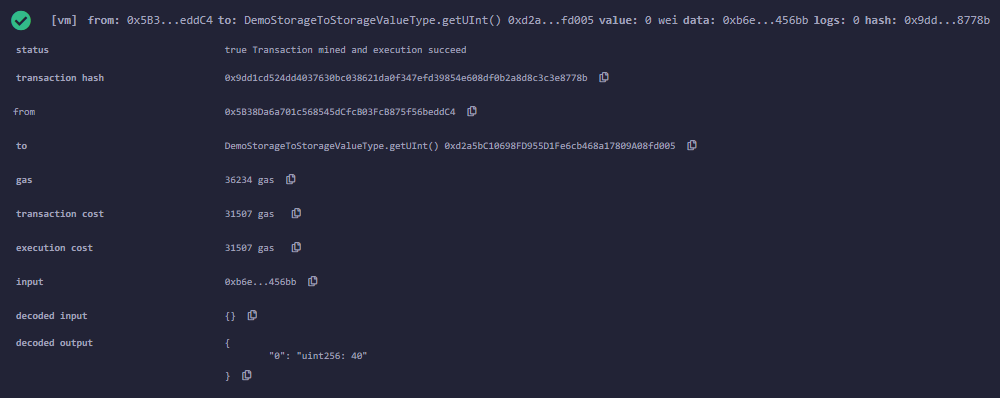
        stateVar2 = 50;

        return stateVar1;

    }

}

Kết quả:



# Demo storage to storage reference type assignment

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

contract DemoStorageToStorageReferenceTypeAssignment {

    uint[2] stateArray1 = [uint(1), 2];

    uint[2] stateArray2 = [uint(3), 4];

    function getUInt() public returns (uint)

    {

        stateArray1 = stateArray2;

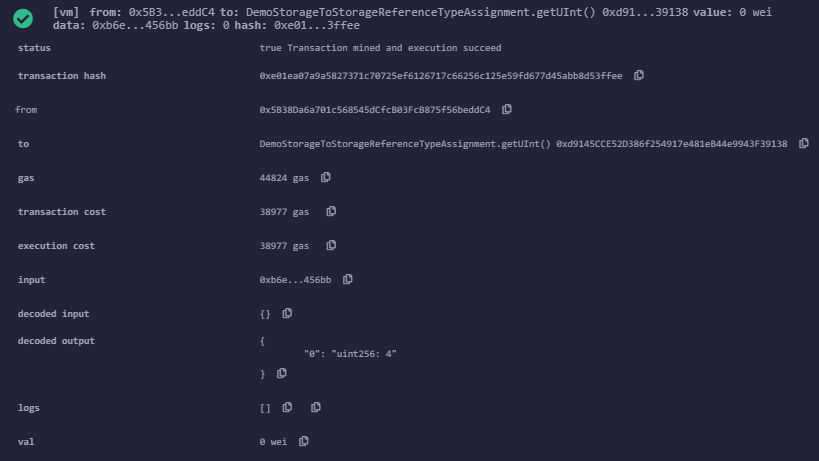
        stateArray2[1] = 5;

        return stateArray1[1];

    }

}

Kết quả:



# Demo memory to storage reference type assignment

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

contract DemoMemoryToStorageReferenceTypeAssignment {

    uint[2] stateArray;

    function getUInt() public returns (uint)

    {

        uint[2] memory localArray = [uint(1), 2];

        stateArray = localArray;

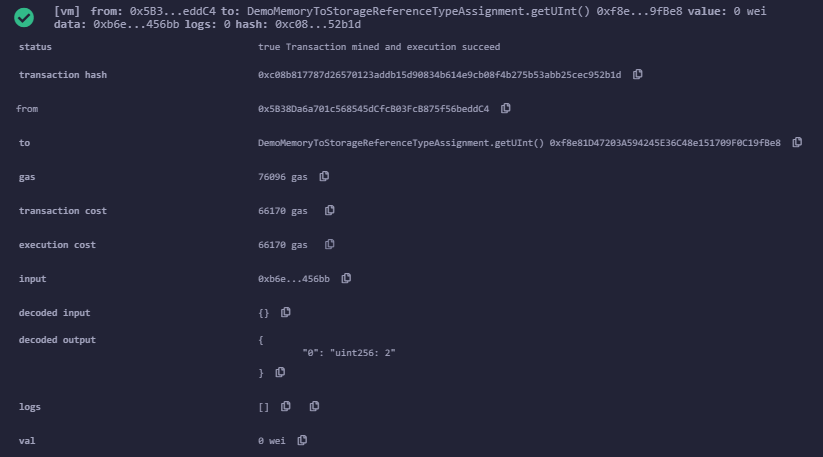
        localArray[1] = 10;

        return stateArray[1];

    }

}

Kết quả:



# Demo storage to memory value type assignment

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

contract DemoStorageToMemoryValueTypeAssignment {

    uint stateVar = 20;

    function getUInt() public returns (uint)

    {

        uint localVar = 40;

        localVar = stateVar;

        stateVar = 50;

        return localVar;

    }

}

Kết quả:



# Demo memory to memory reference type assignment

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

contract DemoMemoryToMemoryReferenceTypeAssignment {

    uint stateVar = 20;

    function getUInt() public returns (uint)

    {

        uint[] memory someVar = new uint[](1);

        someVar[0] = 23;

        uint[] memory otherVar = someVar;

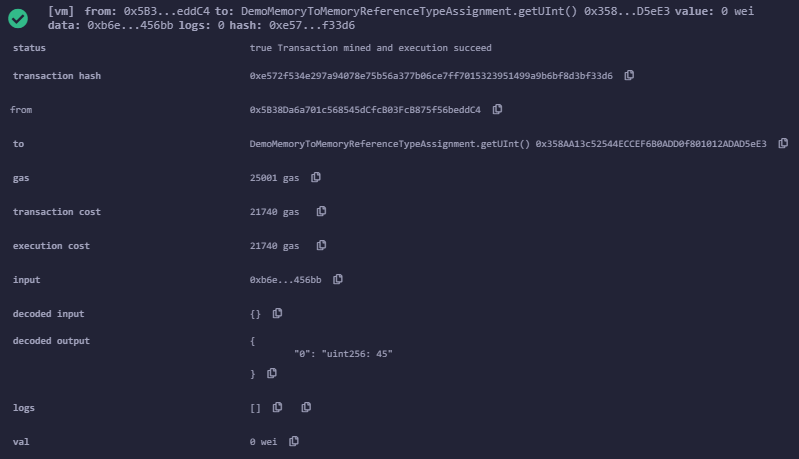
        someVar[0] = 45;

        return (otherVar[0]);

    }

}

Kết quả:



# Smart contract sinh viên

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

contract Class {

    int public stateInt;

    address personIdentifier;

    string std\_id;

    gender \_g;

    Student std;

    uint a;

    struct Student {

        string student\_id;

        string student\_name;

        string DOB;

        bool gender;

    }

    modifier onlyBy() {

        if (keccak256(abi.encodePacked(std\_id)) == keccak256(abi.encodePacked(std.student\_id))) {

            \_;

        }

    }

    event readAge(string, int);

    enum gender { male, female }

    function getAge(string memory s\_id) onlyBy() payable external returns (string memory dob) {

        std = Student("26IT1001234", "Ritesh", "2022/09/03", true);

        emit readAge(s\_id, stateInt);

        \_g = gender.male;

        dob = std.DOB;

        return dob;

    }

}

Kết quả:



# 8.1 Kiểu Integer

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

contract AllAboutInts {

    uint stateUInt = 20;

    uint stateInt = 20;

    function getUInt(uint incomingValue) public

    {

        uint memoryuint = 256;

        uint256 memoryuint256 = 256;

        uint8 memoryuint8 = 8;

        uint256 result = memoryuint8 + memoryuint;

        uint8 assignAfterIncrement = ++memoryuint8;

        uint8 assignBeforeIncrement = memoryuint8++;

    }

    function getInt(int incomingValue) public

    {

        uint memoryInt = 256;

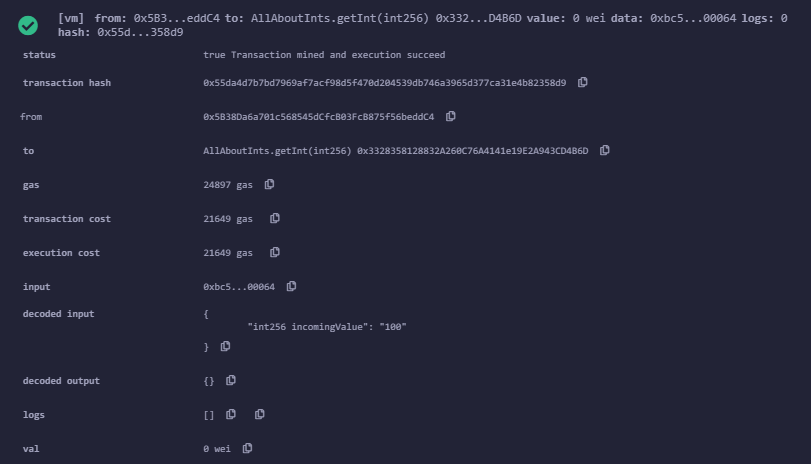
        int256 memoryInt256 = 256;

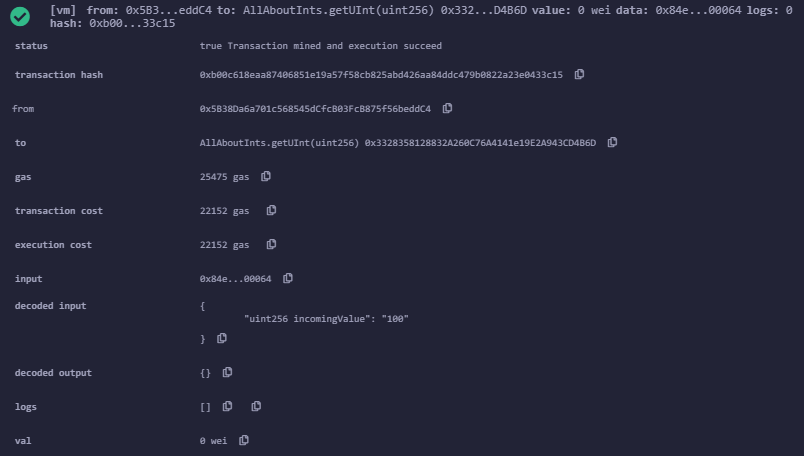
        int8 memoryInt8 = 8;

    }

}

Kết quả:





# 8.2 Kiểu bool

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.7;

contract boolContract {

    bool isPaid = true;

    function manageBool() public returns (bool)

    {

        isPaid = false;

        return isPaid;

    }

    function convertToUint() public returns (uint8)

    {

        isPaid = true;

        return uint8(isPaid);

    }

}