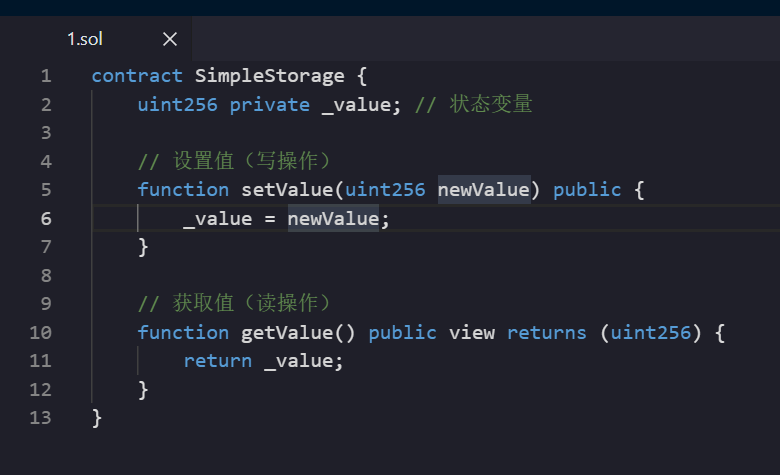
1. **Solidity 智能合约结构**

代码如下：



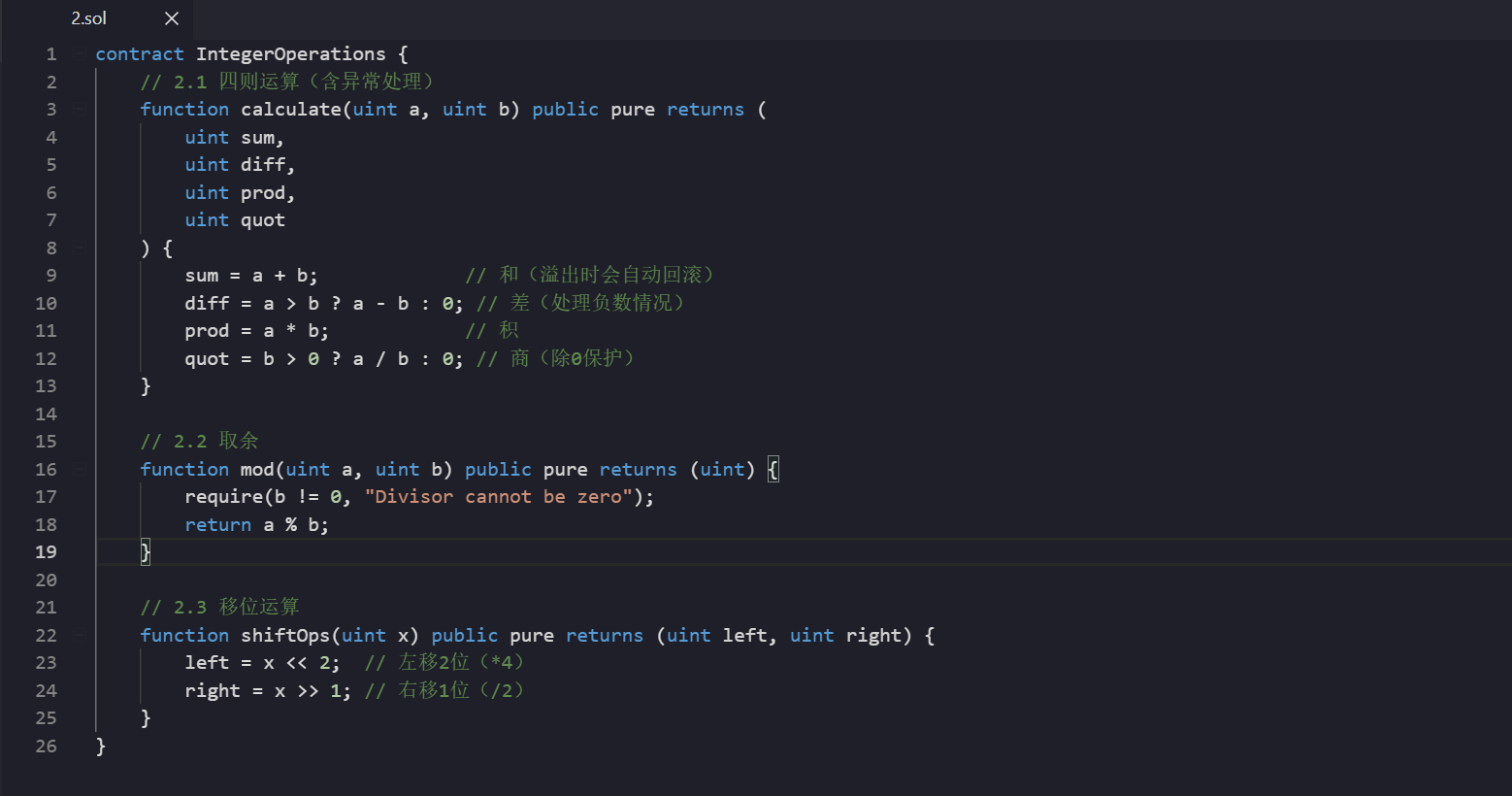
截图如下：



调用setValue输入100后，调用getValue返回值为先前设置的100

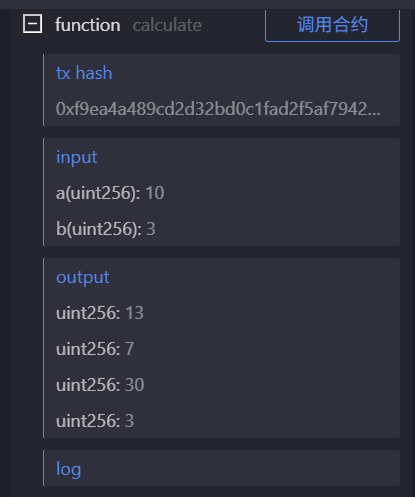
1. **整数**

代码：



正常执行结果：

1.计算加减乘除：



2.取余：

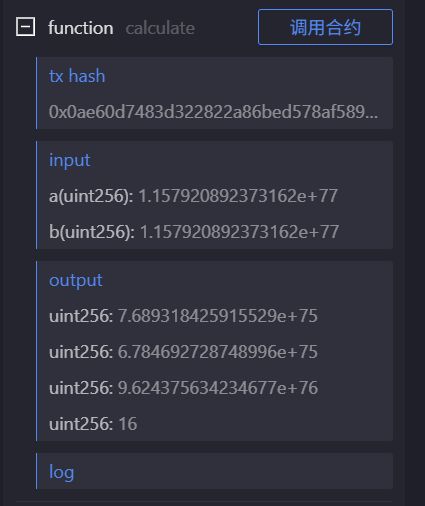


3.：左移右移



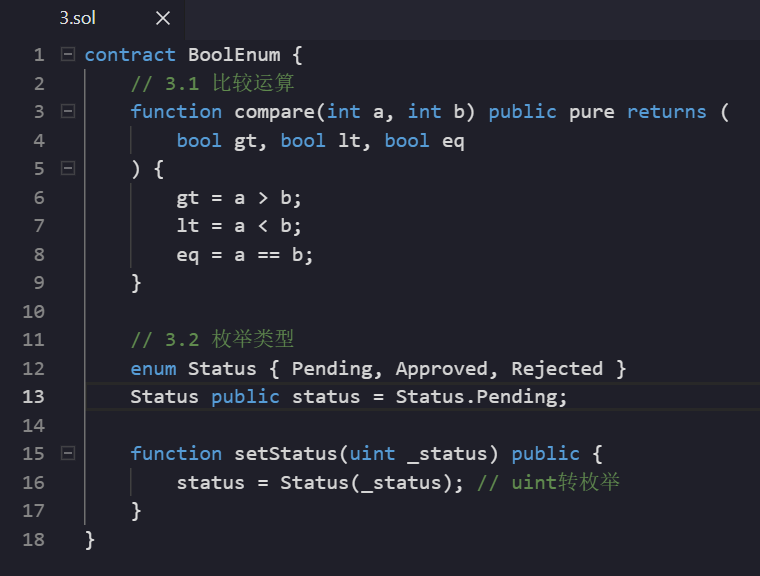
异常执行结果：

输入超过uint256范围的值后运算发生溢出：



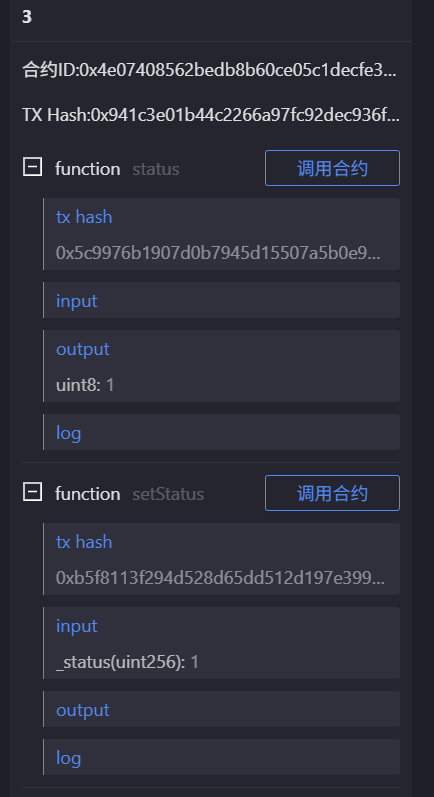
1. **布尔类型与枚举类型**

代码：

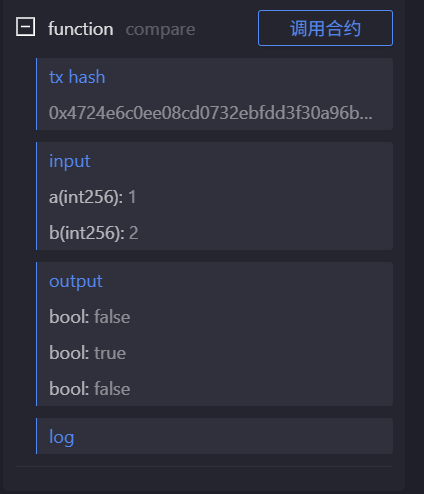


运行：

1.输入状态为1后，可以看到发生了默认转换，输出也为1



2.比较a、b大小并返回比较结果



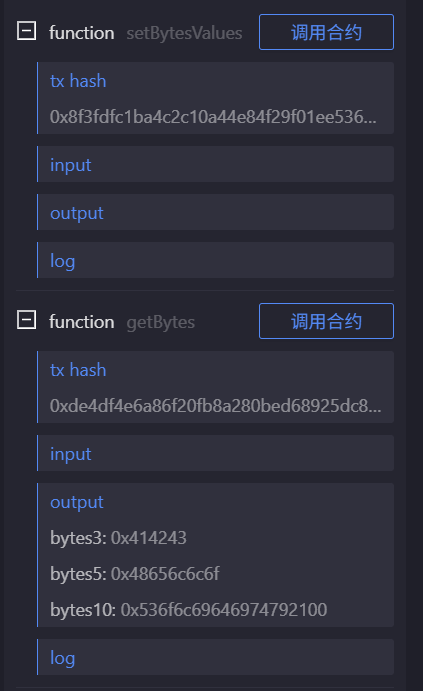
1. **字符串和定长字节数组**

代码：



运行截图：

定长字节数组的创建、赋值、取值：

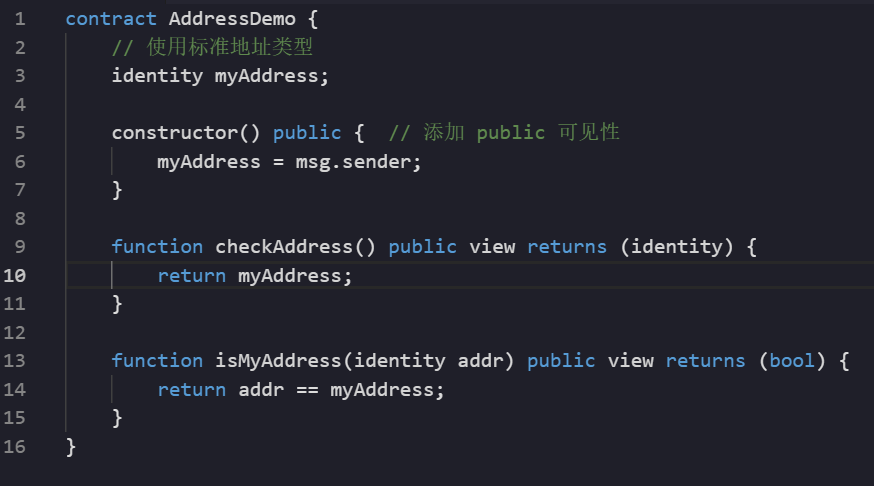


2.字符串直接转换为定长字节数组：



1. **地址类型**

代码如下：

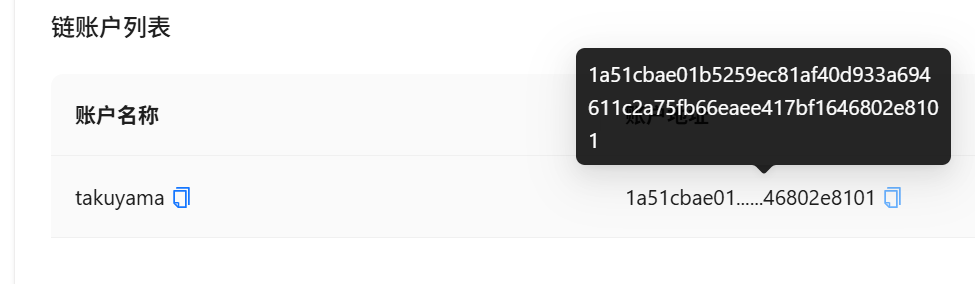


运行截图如下：

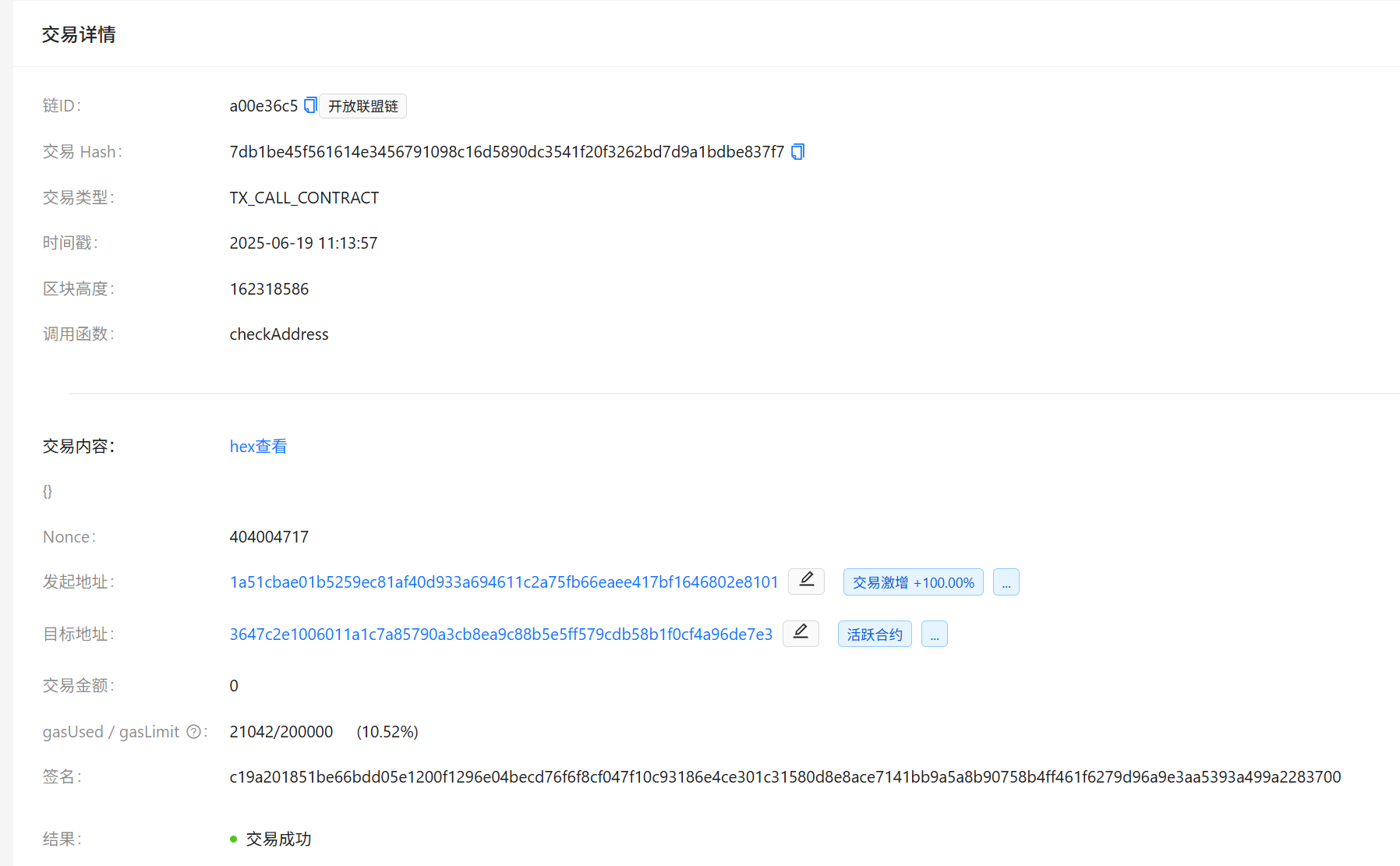
1.获取地址并比较：

两者都是0x1a51cbae01b5259ec81af40d933a694611c2a75fb66eaee417bf1646802e8101



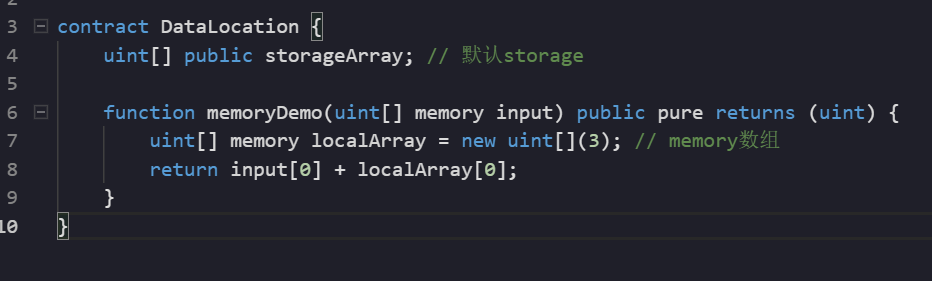


2.在区块链浏览器上查看交易信息：



1. **数据位置**

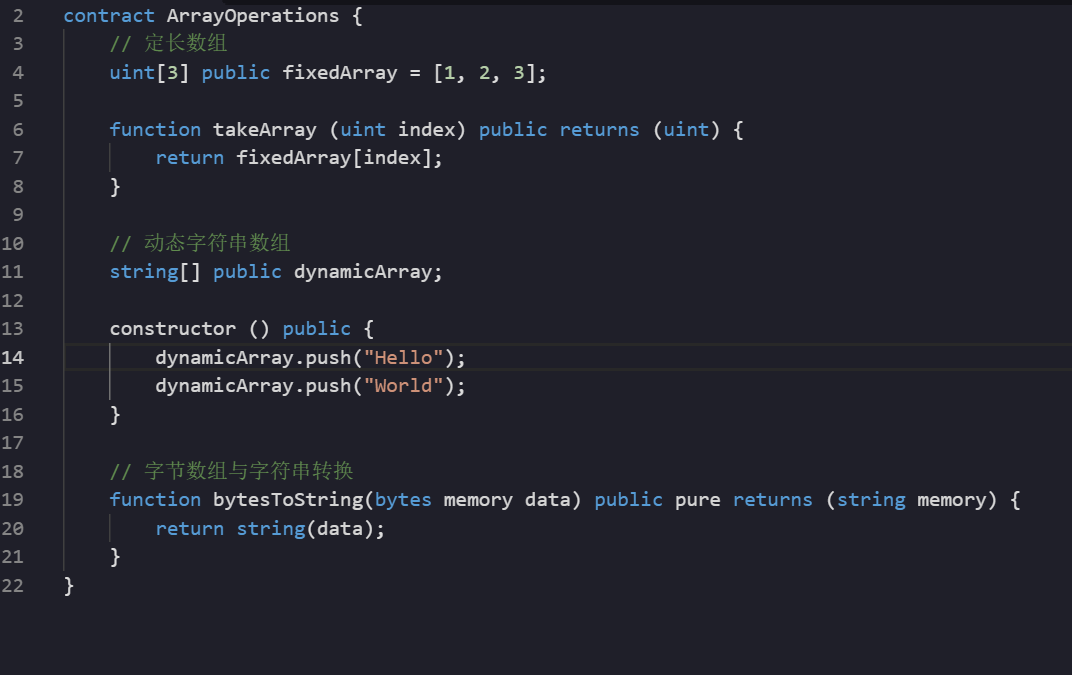
代码如下：

****

Storage与Memory的区别为，storage永久存储在区块链状态数据库中，修改会消耗大量 gas；而memory存储在临时内存中，函数结束后数据消失，gas开销低

1. **数组**

代码如下：

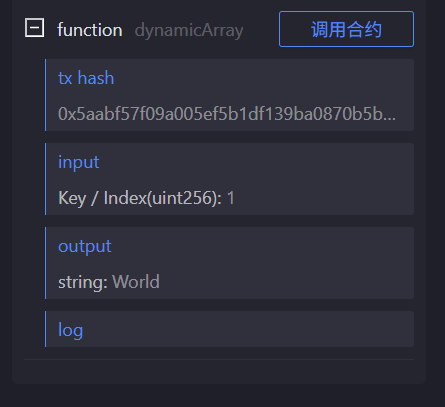
****

运行截图如下：

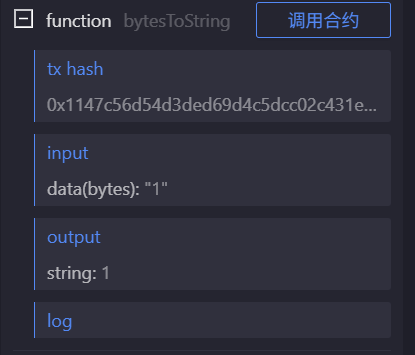
1.创建定长数组并获得元素：



2. 创建动态字符串数组，并获得元素“World”

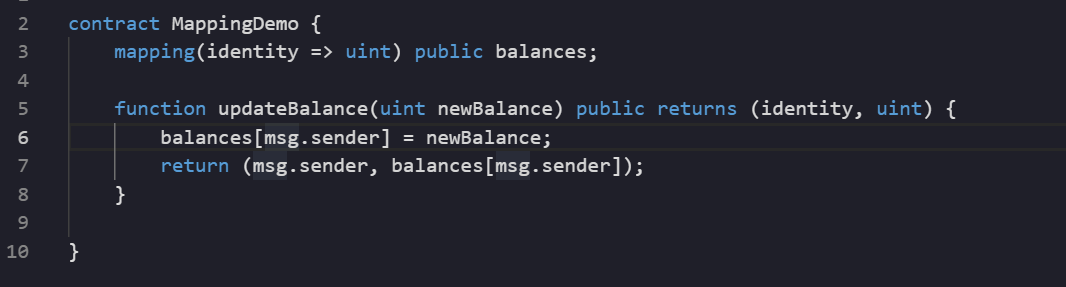


3.变长字节数组与字符串的转换：



1. **映射**

代码如下：



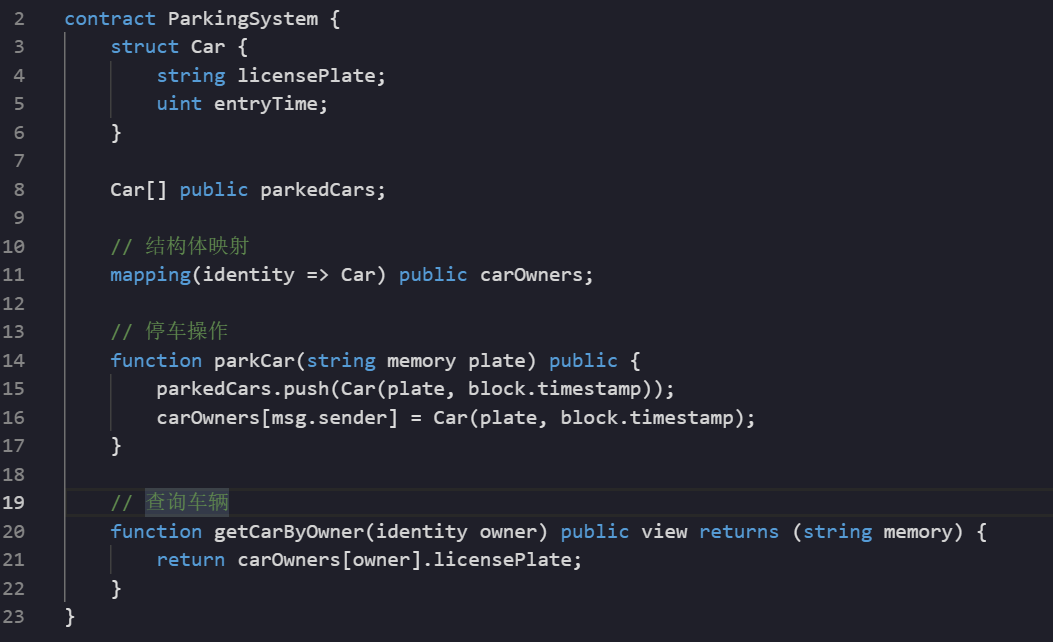
运行截图如下：

可以看到当前地址下成功添加了100的整数



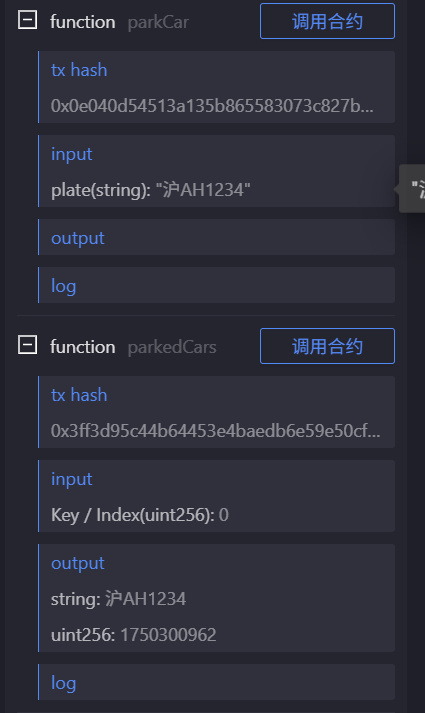
1. **结构体**

代码如下：

****

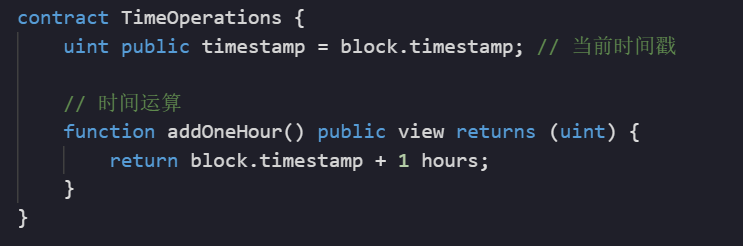
运行截图如下：

先调用parkCar函数输入号牌“沪AH1234”，再调用parkedCars查询到对应信息



1. **时间单位**

代码如下：

****

运行截图如下：

1.获得当前时间戳：

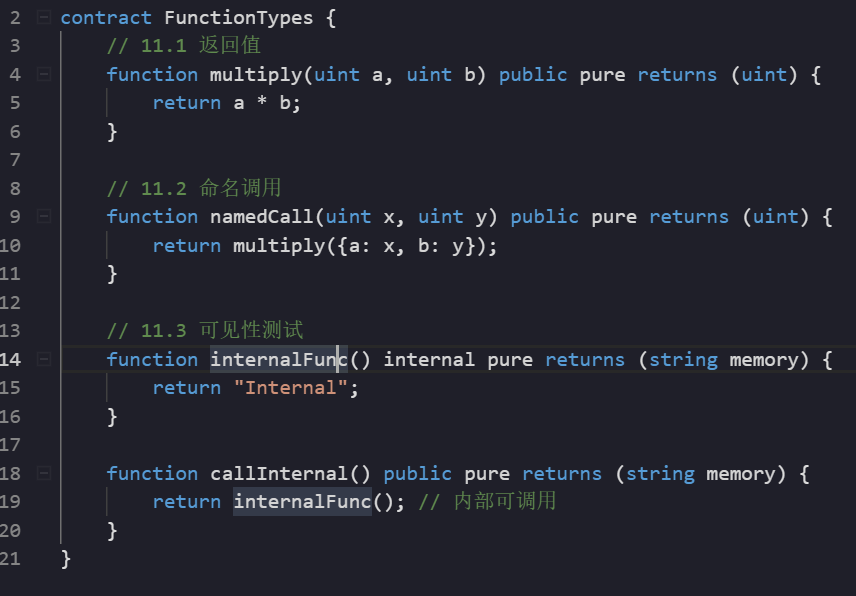
****

2.时间运算

****

1. **Solidity 函数**

代码如下：



运行截图如下：

1. 分别用函数与命名函数计算2\*3：

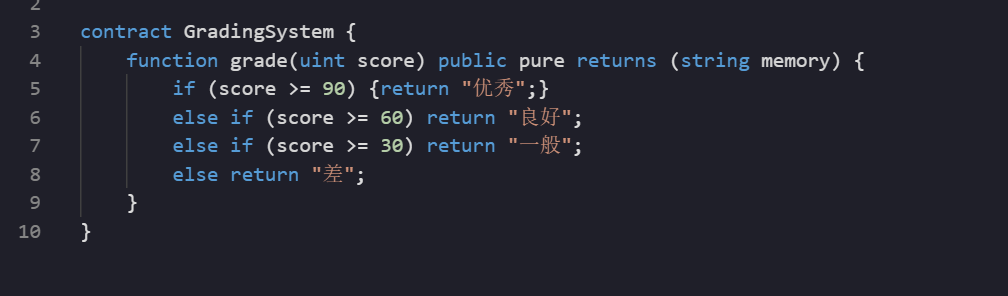


2.测试internal函数：



1. **Solidity 条件语句**

代码如下：

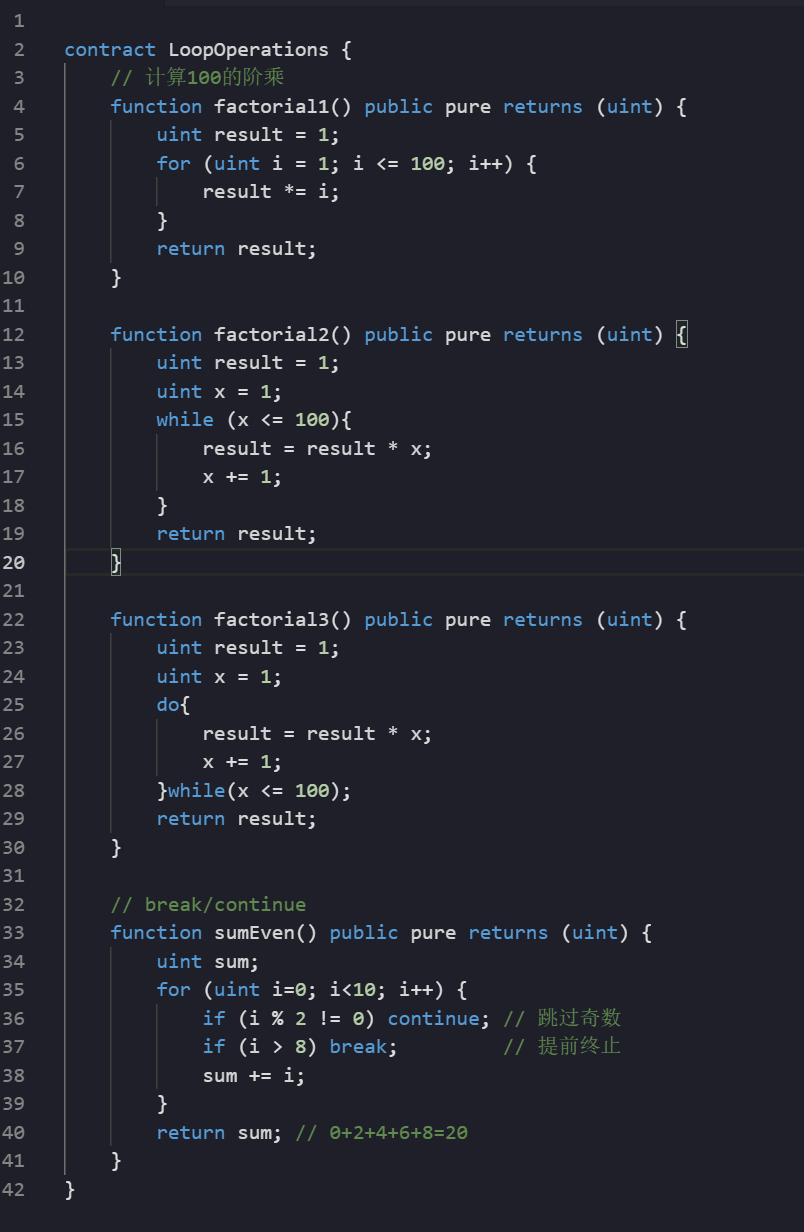
****

运行截图如下：



1. **Solidity 循环语句**

代码如下：

****

运行截图如下：

1.三种循环计算100的阶乘：





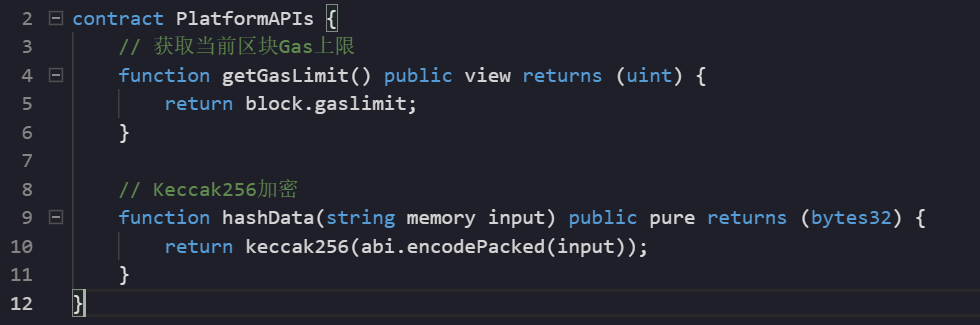


2.break与continue语句：



1. **蚂蚁链常用平台接口函数**

代码如下：

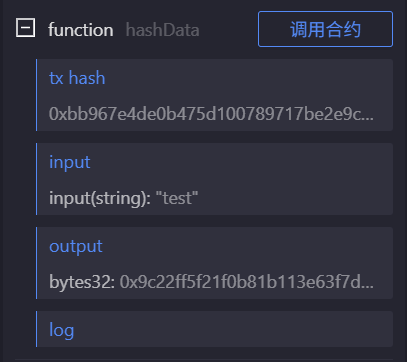


运行截图如下：

1.获取gas上限并输出



2.调用加密函数：



1. **CloudIDE 使用调试技巧**

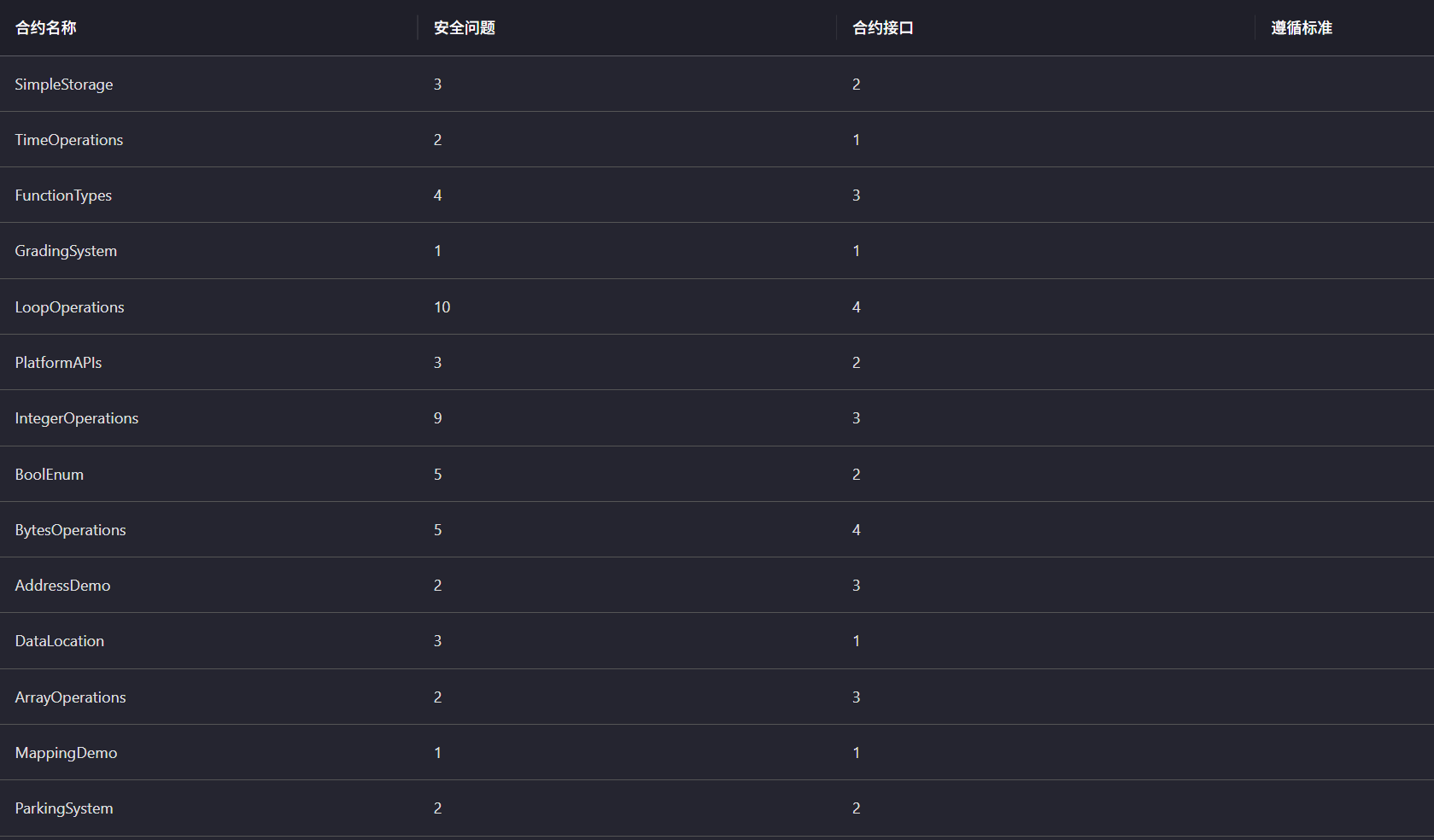
1.本地模拟器部署并查看输出结果：



2.单步调试：

****

3.合约检测任务与查看



****

**下面是各部分的代码：**

1.

contract SimpleStorage {

uint256 private \_value; // 状态变量

// 设置值（写操作）

function setValue(uint256 newValue) public {

\_value = newValue;

}

// 获取值（读操作）

function getValue() public view returns (uint256) {

return \_value;

}

}

2.

contract IntegerOperations {

// 2.1 四则运算（含异常处理）

function calculate(uint a, uint b) public pure returns (

uint sum,

uint diff,

uint prod,

uint quot

) {

sum = a + b; // 和（溢出时会自动回滚）

diff = a > b ? a - b : 0; // 差（处理负数情况）

prod = a \* b; // 积

quot = b > 0 ? a / b : 0; // 商（除0保护）

}

// 2.2 取余

function mod(uint a, uint b) public pure returns (uint) {

require(b != 0, "Divisor cannot be zero");

return a % b;

}

// 2.3 移位运算

function shiftOps(uint x) public pure returns (uint left, uint right) {

left = x << 2; // 左移2位（\*4）

right = x >> 1; // 右移1位（/2）

}

}

3.

contract BoolEnum {

// 3.1 比较运算

function compare(int a, int b) public pure returns (

bool gt, bool lt, bool eq

) {

gt = a > b;

lt = a < b;

eq = a == b;

}

// 3.2 枚举类型

enum Status { Pending, Approved, Rejected }

Status public status = Status.Pending;

function setStatus(uint \_status) public {

status = Status(\_status); // uint转枚举

}

}

4.

contract BytesOperations {

// 不同长度的定长字节数组

bytes3 public fixedBytes3; // 3字节长度

bytes5 public fixedBytes5; // 5字节长度

bytes10 public fixedBytes10; // 10字节长度

// 设置不同长度的字节数组值

function setBytesValues() public {

fixedBytes3 = "ABC"; // 3字节

fixedBytes5 = "Hello"; // 5字节

fixedBytes10 = "Solidity!"; // 9字节（自动填充到10字节）

}

// 获取字节数组值

function getBytes() public view returns (bytes3, bytes5, bytes10) {

return (fixedBytes3, fixedBytes5, fixedBytes10);

}

// 获取字节数组长度

function getBytesLengths() public pure returns (uint, uint, uint) {

bytes3 b3 = "123";

bytes5 b5 = "abcde";

bytes10 b10 = "0123456789";

return (b3.length, b5.length, b10.length);

}

// 直接转换（编译器自动处理）

function directConversion() public pure returns (bytes3) {

return bytes3("ETH"); // 转换为3字节数组

}

}

5.

contract AddressDemo {

// 使用标准地址类型

identity myAddress;

constructor() public { // 添加 public 可见性

myAddress = msg.sender;

}

function checkAddress() public view returns (identity) {

return myAddress;

}

function isMyAddress(identity addr) public view returns (bool) {

return addr == myAddress;

}

}

6.

contract DataLocation {

uint[] public storageArray; // 默认storage

function memoryDemo(uint[] memory input) public pure returns (uint) {

uint[] memory localArray = new uint[](3); // memory数组

return input[0] + localArray[0];

}

}

7.

contract ArrayOperations {

// 定长数组

uint[3] public fixedArray = [1, 2, 3];

function takeArray() public view returns (uint) {

return fixedArray[0];

}

// 动态字符串数组

string[] public dynamicArray;

constructor () public {

dynamicArray.push("Hello");

dynamicArray.push("World");

}

// 字节数组与字符串转换

function bytesToString(bytes memory data) public pure returns (string memory) {

return string(data);

}

}

8.

contract MappingDemo {

mapping(identity => uint) public balances;

function updateBalance(uint newBalance) public {

balances[msg.sender] = newBalance;

}

}

9.

// 9.4 停车场管理系统

contract ParkingSystem {

struct Car {

string licensePlate;

uint entryTime;

}

// 9.2 结构体数组

Car[] public parkedCars;

// 9.3 结构体映射

mapping(identity => Car) public carOwners;

// 停车操作

function parkCar(string memory plate) public {

parkedCars.push(Car(plate, block.timestamp));

carOwners[msg.sender] = Car(plate, block.timestamp);

}

// 查询车辆

function getCarByOwner(identity owner) public view returns (string memory) {

return carOwners[owner].licensePlate;

}

}

10.

contract TimeOperations {

uint public timestamp = block.timestamp; // 10.1 当前时间戳

// 10.2 时间运算

function addOneHour() public view returns (uint) {

return timestamp + 1 hours;

}

}

11.

contract FunctionTypes {

// 11.1 返回值

function multiply(uint a, uint b) public pure returns (uint) {

return a \* b;

}

// 11.2 命名调用

function namedCall(uint x, uint y) public pure returns (uint) {

return multiply({a: x, b: y});

}

// 11.3 可见性测试

function internalFunc() internal pure returns (string memory) {

return "Internal";

}

function callInternal() public pure returns (string memory) {

return internalFunc(); // 内部可调用

}

}

12.

contract GradingSystem {

function grade(uint score) public pure returns (string memory) {

if (score >= 90) {return "优秀";}

else if (score >= 60) return "良好";

else if (score >= 30) return "一般";

else return "差";

}

}

13.

contract LoopOperations {

// 计算100的阶乘

function factorial1() public pure returns (uint) {

uint result = 1;

for (uint i = 1; i <= 100; i++) {

result \*= i;

}

return result;

}

function factorial2() public pure returns (uint) {

uint result = 1;

uint x = 1;

while (x <= 100){

result = result \* x;

x += 1;

}

return result;

}

function factorial3() public pure returns (uint) {

uint result = 1;

uint x = 1;

do{

result = result \* x;

x += 1;

}while(x <= 100);

return result;

}

// break/continue

function sumEven() public pure returns (uint) {

uint sum;

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

if (i % 2 != 0) continue; // 跳过奇数

if (i > 8) break; // 提前终止

sum += i;

}

return sum; // 0+2+4+6+8=20

}

}

14.

contract PlatformAPIs {

// 14.1 获取当前区块Gas上限

function getGasLimit() public view returns (uint) {

return block.gaslimit;

}

// 14.2 Keccak256加密

function hashData(string memory input) public pure returns (bytes32) {

return keccak256(abi.encodePacked(input));

}

}