法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

关注 小象学院

区块链编程：Solidity以太坊智能合约

王亮

第四课 solidity编程:智能合约实现

4.12 函数及基本控制结构

函数类型

- 函数也是一种类型，且属于值类型
- 可以将一个函数赋值给一个变量，一个函数类型的变量
- 还可以将一个函数作为参数进行传递
- 也可以在函数调用中返回一个函数
- 函数类型有两类，可分为internal和external

函数类型

- 内部函数(internal):不能在当前合约的上下文环境以外的地方执行，内部函数只能在当前合约内被使用。如在当前的代码块内，包括内部库函数，和继承的函数中。
- 外部函数（External）:外部函数由地址和函数方法签名两部分组成。可作为外部函数调用的参数，或者由外部函数调用返回。

函数类型的定义

- function (param types) {internal|external} [pure|constant|view|payable] [returns (return types)] varName;
- 若不写类型，默认的函数类型是internal的
- 如果函数没有返回结果，则必须省略returns关键字

函数类型的定义

```
//varName 是一个函数类型，默认为internal类型
function(uint) returns (uint) varName;

//simpleFunction是正常的合约函数，可见性默认为public
function simpleFunction(uint parameter) returns (uint) {
    return parameter;
}

//test是正常的合约函数，varName可以被赋值
function test() returns (uint) {
    varName = simpleFunction;
    return varName(11111);
}
```

函数的定义

- function f(<parameter types>) {internal|external} [pure|constant|view|payable] [returns (<return types>)] { // function body}。

```
contract SimpleFunc {
    function hello(uint i) {
        //todo
    }
}
```

入参和返回值

- 同javascript一样，函数有输入参数，但与之不同的是，函数可能有任意数量的返回参数
- 入参(Input Parameter)与变量的定义方式一致，稍微不同的是，不会用到的参数可以省略变量名称
- 出参（Output Paramets）在returns关键字后定义，语法类似变量的定义方式

入参和返回值

```
//返回和与乘积，变量名可由returns定义
function arithmetics(uint _a, uint _b)
    public
    pure
    returns (uint oSum, uint oProduct)
{
    oSum = _a + _b;
    oProduct = _a * _b;
}

//返回值也可以通过return来定义
function arithmeticsR(uint _a, uint _b) returns (uint, uint) {
    uint oSum = _a + _b;
    uint oProduct = _a * _b;
    return (oSum, oProduct);
}
```

入参和返回值

Solidity内置支持元组(tuple)，这种内置结构可以同时返回多个结果。

```
function f() public pure returns (uint, bool, uint) {
    return (7, true, 2);
}

function g() public {
    //一般使用var定义参数接收返回值，而不是明确定义类型
    var (x, b, y) = f();

    //可以忽略部分返回值
    (x,) = f;
}
```

函数控制结构

- 支持if，else，while，do，for，break，continue，return，?:。
- 条件判断中的括号不可省略，但在单行语句中的大括号可以省略。
- 无Boolean类型的自动转换，比如if(1){...}在Solidity中是无效的。
- 没有switch。

函数调用

- 内部调用:同一个合约中，函数互相调用。调用在EVM中被翻译成简单的跳转指令。
- 外部调用: 一个合约调用另外一个合约的函数，或者通过web3调用合约函数。调用通过消息调用完成(bytes24类型, 20字节合约地址+的4字节的函数方法签名)。

函数调用例子

```
contract InfoFeed {
    //内部调用，可以递归调用
    function g(uint a) returns (uint ret) { return f(); }
    function f() returns (uint ret) { return g(7) + f(); }

    function info() payable returns (uint ret) { return 42; }
}


contract Consumer {
    InfoFeed feed;
    //设置合约地址
    function setFeed(address addr) { feed = InfoFeed(addr); }

    //调用
    function callFeed() returns(uint) {
        return feed.info.value(10).gas(800)();
    }
}
```

命名参数调用和匿名函数参数

- 函数调用的参数，可以通过指定名字的方式调用，但可以以任意的顺序，使用方式为{}包含。参数的类型和数量要与定义一致。

```
function add(uint val1, uint val2) returns (uint) {
    return val1 + val2;
}

function g() returns (uint) {
    //按照参数命名赋值
    return add({val2: 2, val1: 1});
}
```

省略函数参数名称

- 没有使用的参数名可以省略

```
contract C2 {
    //省略参数名称
    function func(uint k, uint) public pure returns(uint) {
        return k;
    }
}
```

省略函数参数名称

- 没有使用的参数名可以省略

```
contract C2 {
    //省略参数名称
    function func(uint k, uint) public pure returns(uint) {
        return k;
    }
}
```

变量作用范围

- 一个变量在声明后都有初始值为字节表示的全0
- Solidity使用了javascript的变量作用范围的规则
- 函数内定义的变量，在整个函数中均可用，无论它在哪里定义

变量作用范围例子

```
function scoping() {
    uint i = 0;
    while (i++ < 1) {
        uint same1 = 0;
    }

    while (i++ < 2) {
        uint same1 = 0;// 报错，重复声明了same1
    }
}

function crossFunction() {
    uint same1 = 0;//正确，没有影响
}
```

流程演示

联系我们

小象学院：互联网新技术在线教育领航者

- 微信公众号：**小象学院**