

solidity自学：第二天

目录

[固定长度字节数组转动态长度字节数组](#)
[bytes与string的相互转换](#)
[for循环](#)
[固定数组详解](#)
[可变长度数组深入](#)
[可变长度二维数组](#)
[以太坊地址的本质](#)

固定长度字节数组转动态长度字节数组

```

1 pragma solidity ^0.4.0;
2
3 contract fixToDynamicArray{
4     bytes12 name = 0x7a68656e676a69616e78756e;
5
6     function fixBytesTodynamicBytes() view returns(bytes){
7         //return bytes(name); !不可以这样子转换
8         bytes memory newName = new bytes(name.length);
9         for(uint i = 0;i < name.length;i++){//不可以是int,而是int
10             newName[i] = name[i];
11         }
12         return newName;
13     }
14 }
```

bytes与string的相互转换

```

1 pragma solidity ^0.4.0;
2 contract bytesToString{
3     bytes name = new bytes(2);
4
5     function Init(){
6         name[0] = 0x7a;
7         name[1] = 0x68;
8     }
9
10    function bytesToString_() returns(string){
11        return string(name);
12    }//行得通，返回“zh”
13 }
```

for循环

形参的传入！

高版本的solidity不能自动填充bytes32，
输入不是32个字节会报错。把输入的形参 (bytes32)改成bytes就可以正常跑了

```

1 pragma solidity ^0.4.4;
2 contract bytes32ToString{
3     bytes2 name = 0x7a68;
4
5     function changeIt() returns(string){
6         //return string(name); ! 这样是不可以的
7         //固定长度字节数组可以转换为bytes可变字节数组
8         //bytes可变字节数组可以转换为string
9     }
10    function bytes32ToString_(bytes
```



LEVI_104

1/4

```

11 |         bytes memory newName = new bytes(_newName.length);
12 |         for(uint i = 0;i < _newName.length;i++){
13 |             newName[i] = _newName[i];
14 |
15 |         return string(newName);
16 |     //! ! 解决方案1：高版本的solidity不能自动填充bytes32,
17 |     //    输入不是32个字节会报错。把输入的形参(bytes32)改成bytes就可以正常跑了
18 |     //!!解决方案2：将输入的0x7a68用0补全至64位，就可以正常运行了
19 |
20 }

```

固定数组详解

没什么好讲的，跟Java差不多

```

1 pragma solidity ^0.4.4;
2 contract fixArray{
3     uint[5] arr = [1,2,3,4,5];
4
5     function Init(){
6         arr[0] = 100;
7         arr[1] = 200;
8     }
9     function getArrayContent() view returns(uint[5]){
10        return arr;
11    }
12    function getGrade() view returns(uint){
13        uint grade = 0;
14        for(uint i = 0;i < 5;i++){
15            grade = grade + arr[i];
16        }
17        return grade;
18    }
19    function changeLength() returns(uint){
20        //return arr.length = 10; ! 不可以
21    }
22    function push(){
23        //arr.push(6); ! 不可以
24    }
25 }

```

可变长度数组深入

```

1 pragma solidity ^0.4.4;
2 contract dynamicArray{
3     uint[] grade;//定义一个可变长度的数组
4
5     function getContent() view returns(uint[]){
6         return grade;
7     }
8     function getLength() view returns(uint){
9         return grade.length;
10    }
11    function add() view returns(uint){
12        uint sum = 0;
13        for(uint i = 0;i < grade.length;i++){
14            sum += grade[i];
15        }
16        return sum;
17    }
18    function changeLength_(){
19        grade.length = 10;
20    }
21 }

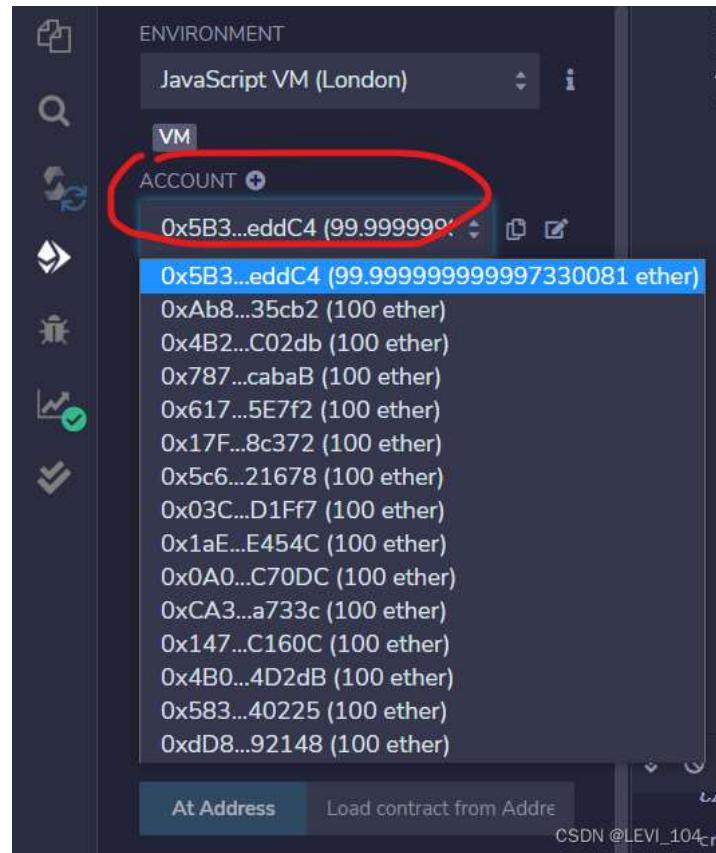
```

可变长度二维数组

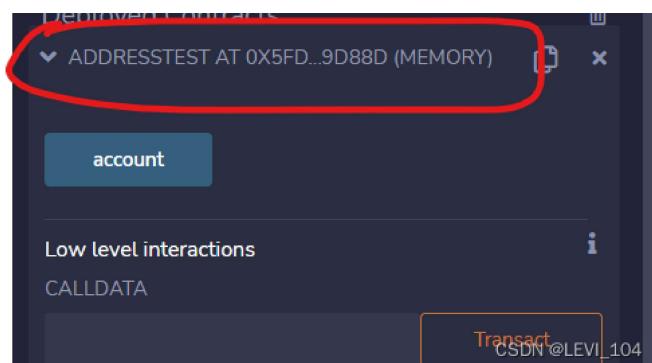


这里就不做阐述（我也没看那几个视频），举一反三—维数组，都差不多的

以太坊地址的本质



圈住的地方就是我的账户，我的地址0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 (160位uint160)



圈住的是本合约在区块链中的地址

```

1 pragma solidity ^0.4.4;
2
3 contract addressTest{
4     address public account = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4; //默认值: 0x000000000000000000000000000000000000000000000000000000000000000
5     //本合约用户地址: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
6     //本合约地址: 0x5FD6eB55D12E759a21C09eF703fe0CBa1DC9d88D
7     //因此, address使用uint160来存储的
8
9     address account1 = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4; //用户地址
10    address account2 = 0xDA0bab807633f07f013f94DD0E6A4F96F8742B53; //合约地址
11
12    function changeIt() view returns(uint160){
13        return uint160(account);
14        //返回: 520786028573371803640530888255888666801131675076
15    }
16
17    function changeIt2() view returns(address){
18        return address(520786028573371803640530888255888666801131675076
19        //返回: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

```

```
20 |     }21 |
21 |     function check1() view returns(bool){
22 |         return account > account2;
23 |     }//F
24 |     function check2() view returns(bool){
25 |         return account >= account2;
26 |     }//F
27 |     function check3() view returns(bool){
28 |         return account < account2;
29 |     }//T
30 |     function check4() view returns(bool){
31 |         return account <= account2;
32 |     }//T
33 | //这说明了，合约地址 > 用户地址
34 |
35 }
```

address是uint160存储；合约地址 > 用户地址

“相关推荐”对你有帮助么？



关于我们 招贤纳士 商务合作 寻求报道 ☎ 400-660-0108 📩 kefu@csdn.net 💬 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心
家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 ©1999-2022北京创新乐知网络技术有限公司 版权与免责声明 版权申诉
出版物许可证 营业执照