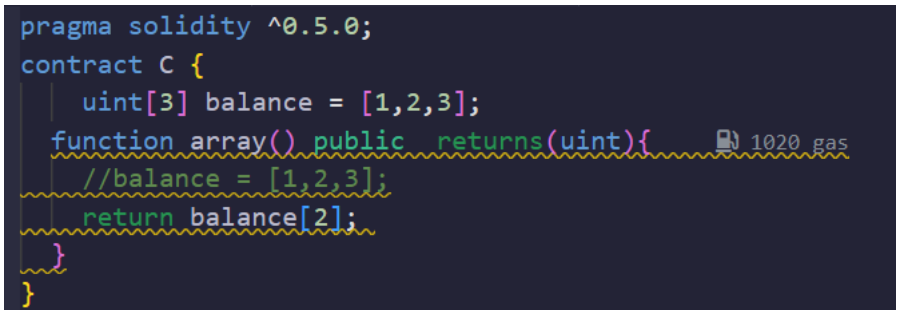
数组的补充教学：

我们做剪刀石头布游戏的时候，需要数组的支持。让我们敲起键盘，活动一下手吧！

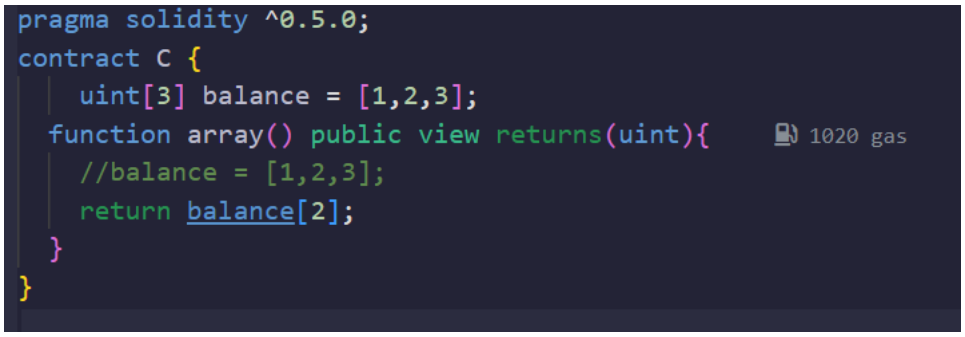
1、例 1 是定义一个数组 balance。这里例子中用 array()返回 balance[2]。请用 remix 编译，部署，运行 array()，查看返回的结果。

例 1:

```
pragma solidity ^0.5.0;
contract C {
    uint[3] balance = [1,2,3];
    function array() public returns(uint){
        //balance = [1,2,3];
        return balance[2];
    }
}
```



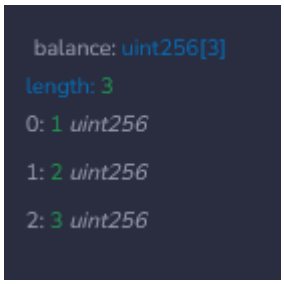
这里首先会有一个 Warninig,加上一个 view 后即可



返回结果:

```
{
    "0": "uint256: 3"
}
```

“0”是第 1 个输出；“uint256”是第 1 个输出的类型；“3”是输出的值；

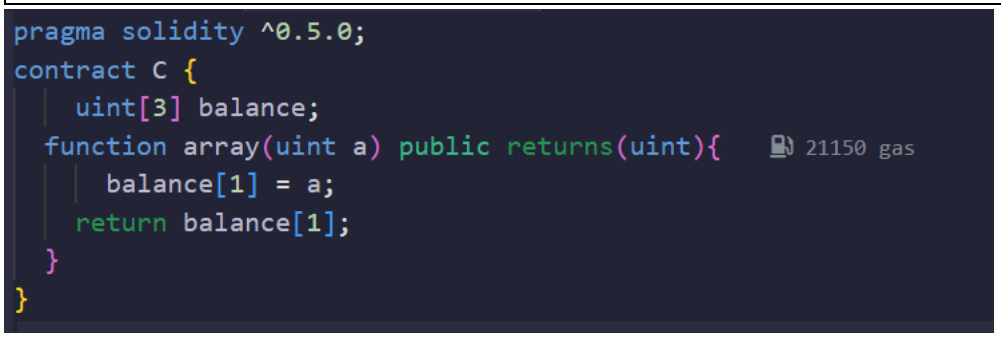


*请分析合约中的 uint[3]的数字"3"和 balance[2]的数字"2"，分别代表了什么意思?

Unit[3]表示数组的长度为 3，balance[2]表示数组的索引为 2

2、可以测试一下 array()中给 balance[1]赋值。

```
pragma solidity ^0.5.0;
contract C {
    uint[3] balance;
    function array(uint a) public returns(uint){
        balance[1] = a;
        return balance[1];
    }
}
```



程序代码

```
call  [call] from: 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db to: C.array() data: 0xb0e...c2ae1

from                0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
to                  C.array() 0xd9145CCE52D386f254917e481eB44e9943F39138
执行成本            2320 gas (当被一个合约调用是才需要费用)
输入                0xb0e...c2ae1
解码输入            {}
解码输出            {
                        "0": "uint256: 3"
                    }
日志                []
```

```
balance: uint256[3]
length: 3
0: 0 uint256
1: 0 uint256
2: 0 uint256
```

3、再试试二维数组

a.声明

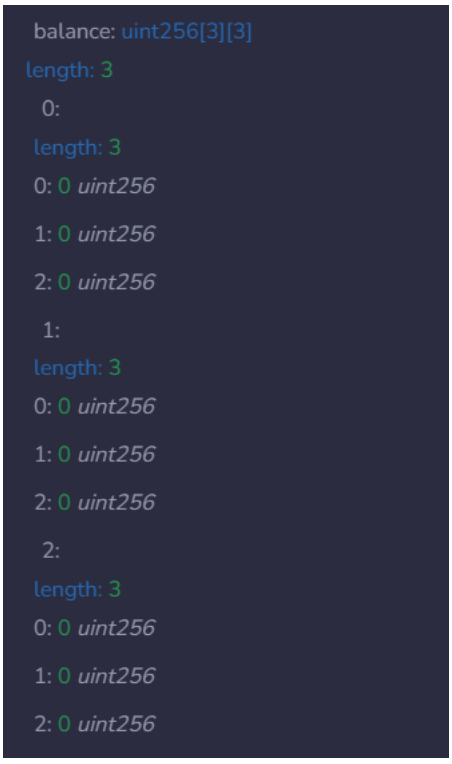
```
pragma solidity ^0.5.0;
contract C {
    uint[3][3] balance;

    function array(uint a) public returns(uint){
        balance[1][1] = a;
        return balance[1][1];
    }
}
```

程序代码

```
pragma solidity ^0.5.0;
contract C {     RETURN costs 0 gas - this line costs 99 gas - 62378 gas left
    uint[3][3] balance;

  function array(uint a) public returns(uint){     21224 gas
      balance[1][1] = a;
    return balance[1][1];
  }
}
```

部署

输出结果



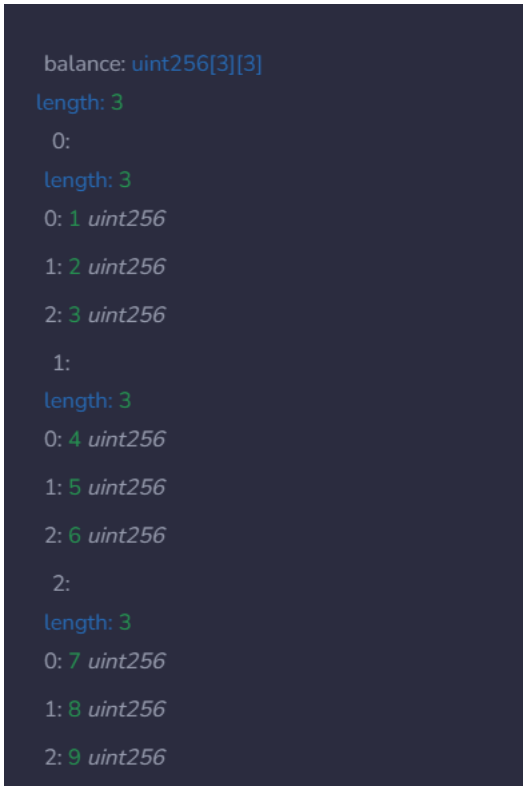
b. 了解二维数组的结构;

这个程序加入了输入的参数，这样可以查看确定二维数组的结构。

```
pragma solidity ^0.5.0;
contract C {
```

```
    uint[3][3] balance = [[1,2,3],[4,5,6],[7,8,9]];

    function array(uint indexx, uint indexy) public returns(uint){
                return balance[indexx][indexy];
    }
}
```

```
pragma solidity ^0.5.0;
contract C {     RETURN costs 0 gas - this line costs 71 gas - 87444 gas left
    uint[3][3] balance = [[1,2,3],[4,5,6],[7,8,9]];

  function array(uint indexx, uint indexy) view  public returns(uint){     1173 gas
          return balance[indexx][indexy];
   }
}
```



3. 获取数组长度：

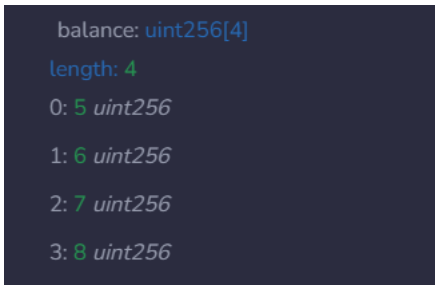
```
pragma solidity ^0.5.0;
contract C {
    uint[4] balance = [5,6,7,8];

    function array(uint indexx) public returns(uint, uint){

                return (balance[indexx], balance.length);
    }
}
```

程序代码，依然加了一个 view

```
pragma solidity ^0.5.0;
contract C {
    uint[4] balance = [5,6,7,8];

  function array(uint indexx) view  public returns(uint, uint){    1138 gas

        return (balance[indexx], balance.length);
  }
}
```

部署

输出结果



4. 数组的简单应用

测试 if 与数组在一起的小功能。

```
pragma solidity ^0.5.0;
contract C {
      uint[4] balance = [5,6,7,8];

      function array(uint indexx) public returns(uint, bool){
                  if(balance[indexx] == 8){

                  return(balance[indexx], true);
```

```
                    }
                    return(balance[indexx], false);
    }
}
```

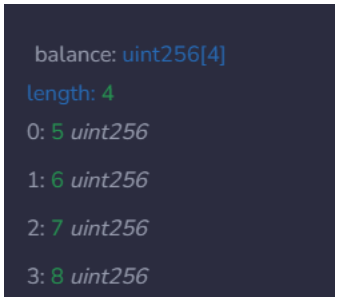
程序代码

```
pragma solidity ^0.5.0;
contract C {
    uint[4] balance = [5,6,7,8];

  function array(uint indexx) view  public returns(uint, bool){    2016 gas
                if(balance[indexx] == 8){

                return(balance[indexx], true);
                }
                return(balance[indexx], false);
  }
}
```

部署

状态	0x1 交易已打包且执行成功
交易哈希	0x82526bb0793a4726acd33e4a96e1c954137870a8d56b8ca4216436045f033561
区块哈希	0xe0a3617adace9c3d48f07feac47b4408f82d638f746af08a55269a05a441694a
区块号	7
合约地址	0x358AA13c52544ECCEF6B0ADD0f801012ADAD5eE3
from	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to	C.(constructor)
gas	gas
交易成本	198295 gas
执行成本	138361 gas
输入	0x608...10032
解码输入	[]
解码输出	-
日志	[]

运行结果


balance: uint256[4]
length: 4
0: 5 uint256
1: 6 uint256
2: 7 uint256
3: 8 uint256