

『0010』 - Solidity Types - 整型(Integer)

孔壹学院：国内区块链职业教育领先品牌

作者：黎跃春，区块链、高可用架构工程师

微信：liyc1215 QQ群：348924182 博客：<http://liyuechun.org>

`int/uint`：变长的有符号或无符号整型。变量支持的步长以 8 递增，支持从 `uint8` 到 `uint256`，以及 `int8` 到 `int256`。需要注意的是，`uint` 和 `int` 默认代表的是 `uint256` 和 `int256`。

什么是有符号整型，什么是无符号整型

无符号整型（`uint`）是计算机编程中的一种数值资料型别。**有符号整型**（`int`）可以表示任何规定范围内的整数，**无符号整型**只能表示非负数（0及正数）。

有符号整型能够表示负数的代价是其能够存储正数的范围的缩小，因为其约一半的数值范围要用来表示负数。如：`uint8` 的存储范围为 0 ~ 255，而 `int8` 的范围为 -127 ~ 127

如果用二进制表示：

- **uint8**: `0b 00000000` ~ `0b 11111111`，每一位都存储值，范围为 0 ~ 255
- **int8**: `0b 11111111` ~ `0b 01111111`，最左一位表示符号，1 表示负，0 表示正，范围为 -127 ~ 127

支持的运算符

- 比较：`<=`，`<`，`==`，`!=`，`>=`，`>`，返回值为 `bool` 类型。
- 位运算符：`&`，`|`，`(^ 异或)`，`(~ 非)`。
- 数学运算：`+`，`-`，一元运算 `+`，`*`，`/`，`(% 求余)`，`(** 次方)`，`(<< 左移)`，`(>> 右移)`。

Solidity目前沒有支持 `double/float`，如果是 `7/2` 会得到 3，即无条件舍去。但如果运算符是字面量，则不会截断(后面会进一步提到)。另外除0会抛异常，我们来看看下面的这个例子：

一、加 +，减 -，乘 *，除 /

```
pragma solidity ^0.4.4;
```

```

contract Math {

    function mul(int a, int b) constant returns (int) {

        int c = a * b;
        return c;
    }

    function div(int a, int b) constant returns (int) {

        int c = a / b;
        return c;
    }

    function sub(int a, int b) constant returns (int) {

        return a - b;
    }

    function add(int a, int b) constant returns (int) {

        int c = a + b;
        return c;
    }
}

```

The screenshot displays the Remix IDE interface. On the left, the 'browser' tab shows the Solidity code for the 'Math' contract, which includes functions for multiplication, division, subtraction, and addition. The code is as follows:

```

pragma solidity ^0.4.4;

contract Math {
    function mul(int a, int b) constant returns (int) {
        int c = a * b;
        return c;
    }
    function div(int a, int b) constant returns (int) {
        int c = a / b;
        return c;
    }
    function sub(int a, int b) constant returns (int) {
        return a - b;
    }
    function add(int a, int b) constant returns (int) {
        int c = a + b;
        return c;
    }
}

```

On the right, the 'JavaScript VM' environment is shown. It includes fields for 'Environment' (set to JavaScript VM), 'Account' (0xca3...a733c), 'Gas limit' (3000000), and 'Value' (0). Below these, the 'browser/ballot.sol:Math' contract is selected. The 'At Address' field is empty, and the 'Create' button is visible. A section titled '0 pending transactions' is also present. At the bottom, a table shows the state of the contract's memory:

browser/ballot.sol:Math at 0x692...77b3a (memory)	
sub	3,5
int256: -2	
mul	3,5
int256: 15	
div	7,3
int256: 2	
add	3,5
int256: 8	

安全 | <https://remix.ethereum.org/#version=soljson-v0.4.13+commit.fb4cb1a.js>

browser/ballot.sol

```
1 pragma solidity ^0.4.4;
2
3 contract Math {
4
5     function mul(int a, int b) constant returns (int) {
6
7         int c = a * b;
8         return c;
9     }
10
11    function div(int a, int b) constant returns (int) {
12
13        int c = a / b;
14        return c;
15    }
16
17    function sub(int a, int b) constant returns (int) {
18
19        return a - b;
20    }
21
22    function add(int a, int b) constant returns (int) {
23
24        int c = a + b;
25        return c;
26    }
27 }
```

3/0抛出异常

[2] only remix transactions, script Listen on network

[call] from: -, to: browser/ballot.sol:Math.sub(int256,int256) 0x692...77b3a, data: adefc...0 Debug

{

"int256": "0"

}

call to browser/ballot.sol:Math.div

[call] from: -, to: browser/ballot.sol:Math.div(int256,int256) 0x692...77b3a, data: 43509...0 Debug

{

"int256": "0"

}

call to browser/ballot.sol:Math.div errored: VM error: invalid opcode.
The constructor should be payable if you send value.
Debug the transaction to get more information.

Environment: JavaScript VM

Account: 0xca3...a733c (99.9999999999998206)

Gas limit: 3000000

Value: 0

browser/ballot.sol:Math

At Address: Enter contract's address - i.e. 0x60606...

Create

0 pending transactions

browser/ballot.sol:Math at 0x692...77b3a (memory)

sub	int256 a, int256 b
int256: 0	
mul	int256 a, int256 b
div	3,0
add	int256 a, int256 b

二、求余 %

```
pragma solidity ^0.4.4;

contract Math {

    function m(int a, int b) constant returns (int) {

        int c = a % b;
        return c;
    }
}
```

The screenshot shows the Remix IDE interface. On the left, the Solidity code editor displays a contract named `Math` with a function `m` that takes two integers `a` and `b` and returns their remainder `a % b`. The code is as follows:

```
1 pragma solidity ^0.4.4;
2
3 contract Math {
4
5     function m(int a, int b) constant returns (int) {
6
7         int c = a % b;
8         return c;
9     }
10
11 }
```

On the right, the 'Environment' tab of the JavaScript VM is active. It shows the following configuration:

- Environment: JavaScript VM
- Account: 0xca3...a733c (99.9999999999998957)
- Gas limit: 3000000
- Value: 0

Below the environment settings, the 'At Address' field is empty, and the 'Create' button is visible. The '0 pending transactions' section is also empty. The 'browser/ballot.sol:Math at 0x692...77b3a (memory)' section shows the state of the contract's memory:

Variable	Value
m	10,3
int256: 1	

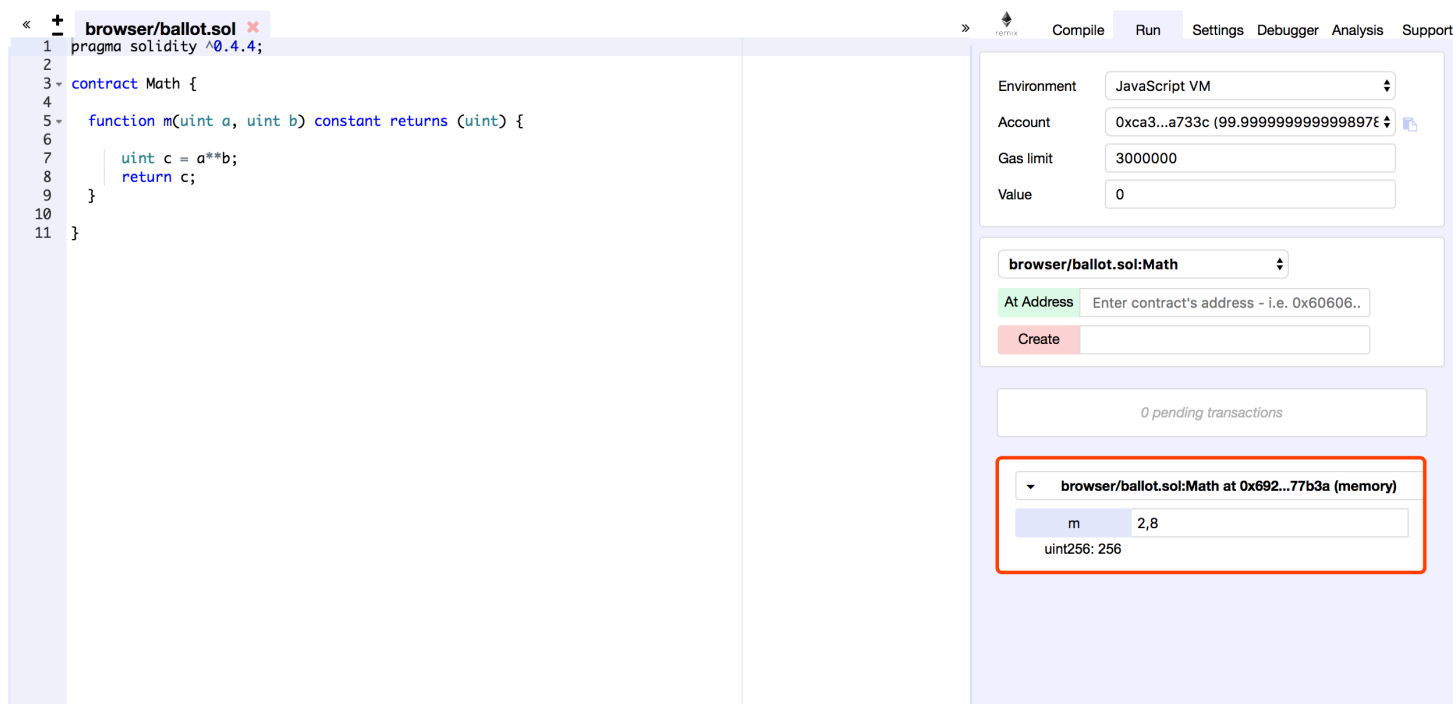
三、次方

```
pragma solidity ^0.4.4;

contract Math {

    function m(uint a, uint b) constant returns (uint) {

        uint c = a**b;
        return c;
    }
}
```



四、与 &, | 或, 非 ~, ^ 异或

```
pragma solidity ^0.4.4;

contract Math {

    function yu() constant returns (uint) {

        uint a = 3; // 0b0011
        uint b = 4; // 0b0100

        uint c = a & b; // 0b0000
        return c; // 0
    }

    function huo() constant returns (uint) {

        uint a = 3; // 0b0011
        uint b = 4; // 0b0100

        uint c = a | b; // 0b0111
        return c; // 7
    }

    function fei() constant returns (uint8) {

        uint8 a = 3; // 0b00000011
        uint8 c = ~a; // 0b11111100
        return c; // 0
    }
}
```

```
function yihuo() constant returns (uint) {

    uint a = 3; // 0b0011
    uint b = 4; // 0b0100

    uint c = a ^ b; // 0b0111
    return c; // 252
}
```

The screenshot displays the Remix IDE interface. On the left, the Solidity code editor shows a contract named `Math` with four functions: `yu()`, `huo()`, `fei()`, and `yihuo()`. Each function contains a calculation involving `uint` variables. Red arrows point from specific lines of code in the editor to the memory view on the right. The memory view, titled `browser/ballot.sol:Math at 0x692...77b3a (memory)`, shows the state of memory slots for each function: `yu` (uint256: 0), `yihuo` (uint256: 7), `huo` (uint256: 7), and `fei` (uint8: 252). The bottom panel shows the debugger with two transactions: `call to browser/ballot.sol:Math.huo` and `call to browser/ballot.sol:Math.fei`, both with `Debug` buttons.

五、位移

```
pragma solidity ^0.4.4;

contract Math {

    function leftShift() constant returns (uint8) {

        uint8 a = 8; // 0b00001000
        uint8 c = a << 2; // 0b00100000
        return c; // 32
    }
```

```
function rightShift() constant returns (uint8) {
```

```
    uint8 a = 8; // 0b00001000  
    uint8 c = a >> 2; // 0b00000010  
    return c; // 2
```

```
}
```

```
}
```

The screenshot displays the Remix IDE interface. On the left, the Solidity code editor shows a contract named `Math` with two functions: `leftShift` and `rightShift`. The `leftShift` function takes a `uint8` value `a` and shifts it left by 2 bits, returning `32`. The `rightShift` function takes a `uint8` value `a` and shifts it right by 2 bits, returning `2`. Two red arrows point from the function calls in the right-hand pane to the corresponding function definitions in the code editor.

On the right, the 'Run' tab is active, showing the execution environment. The 'Environment' is set to 'JavaScript VM'. The 'Account' is `0xca3...a733c (100 ether)`. The 'Gas limit' is `3000000` and the 'Value' is `0`. Below this, the 'browser/ballot.sol:Math' contract is selected, and the 'At Address' field is empty. The 'Create' button is visible.

At the bottom, the 'Logs' pane shows the execution results. It displays the creation of the `browser/ballot.sol:Math` contract, followed by two function calls: `rightShift` and `leftShift`. The `rightShift` call returns `uint8: 2`, and the `leftShift` call returns `uint8: 32`.

- `a << n` 表示a的二进制位向左移动 `n` 位，在保证位数没有溢出的情况下等价于 `a` 乘以2的`n` 次方。
- `a >> n` 表示a的二进制位向右移动 `n` 位，在保证位数没有溢出的情况下等价于 `a` 除以2的`n` 次方。

整数字面量

整数字面量，由包含 `0-9` 的数字序列组成，默认被解释成十进制。在Solidity中不支持八进制，前导0会被默认忽略，如`0100`，会被认为是`100`，【PS：十六进制可以这么写，`0x11`】。

小数由 `.` 组成，在他的左边或右边至少要包含一个数字。如 `1.`，`.1`，`1.3` 均是有效的小数。

[illegible]

