1、合约的本质是：在合适的条件下触发交易。

2、局部变量（本地变量）包括：函数中的变量、输入参数和输出参数。

3、构造函数（constructor）：在合约创建时，调用相关代码完成初始化。构造函数不支持重载，在构造函数中不能使用 this，因为其还不存在。然后，构造函数最好不要带参数，默认用 public 修饰，用 internal 修饰时，合约为抽象合约。

4、solidity 注释：

单行注释： //.....

多行注释： /**

*

*/

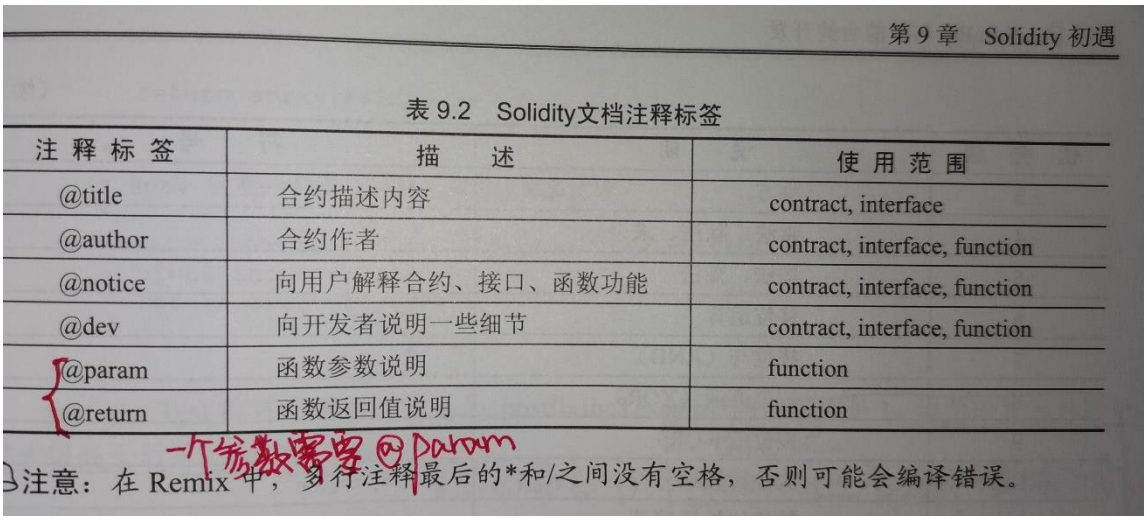
用户文档：直接在方法上使用多行注释或使用单行注释时用三个反斜杠

```
/// 3 斜杠用户文档
function threeSlashUserDoc() public pure{
}
/**
 * 多行注释用户文档
 */
function multipleUserDoc() public pure{
}
```

开发文档：需要加一些注释标签，也有两种方式，多行注释或三斜杠

```
/// @param id 参数说明
/// @dev 方法说明
/// @return 返回值说明
function threeSlashDevDoc(uint8 id) public pure returns(bool){
    return id > 0;
}
 /**
  * @param id 参数说明
  * @dev 方法说明
```

```
    * @return  返回值说明
    */
function multipleDevDoc(uint8 id) public pure returns(bool result){
    result = id > 0;
}
```



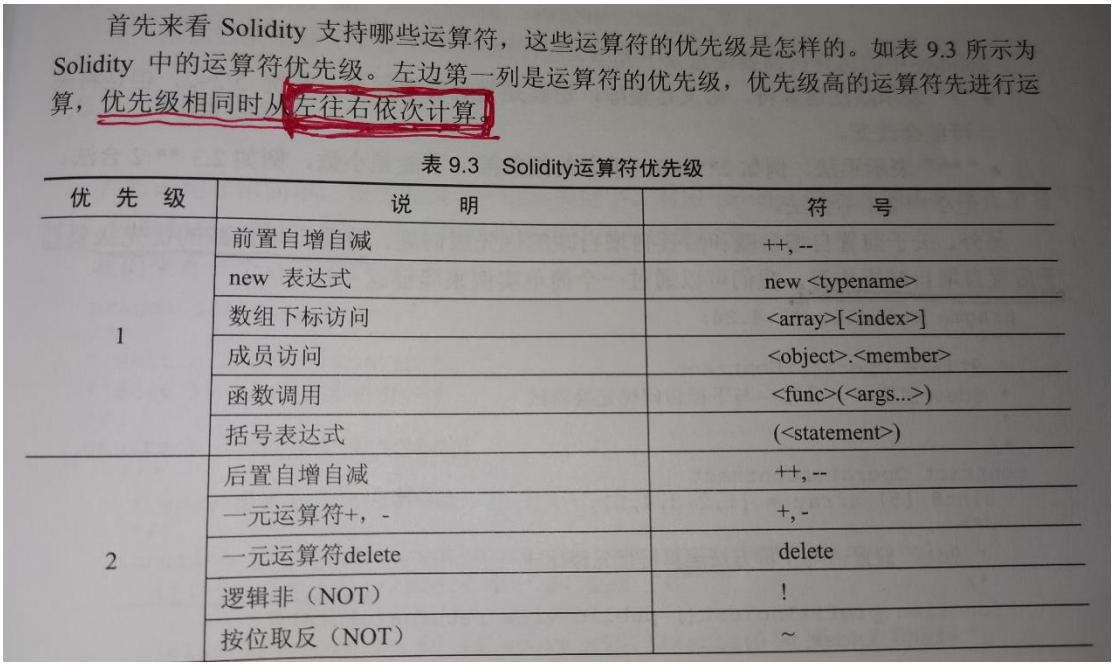
第 9 章　Solidity 初遇

表 9.2　Solidity文档注释标签

注 释 标 签	描　　述	使 用 范 围
@title	合约描述内容	contract, interface
@author	合约作者	contract, interface, function
@notice	向用户解释合约、接口、函数功能	contract, interface, function
@dev	向开发者说明一些细节	contract, interface, function
@param	函数参数说明	function
@return	函数返回值说明	function

注意：在 Remix 中，多行注释最后的*和/之间没有空格，否则可能会编译错误。

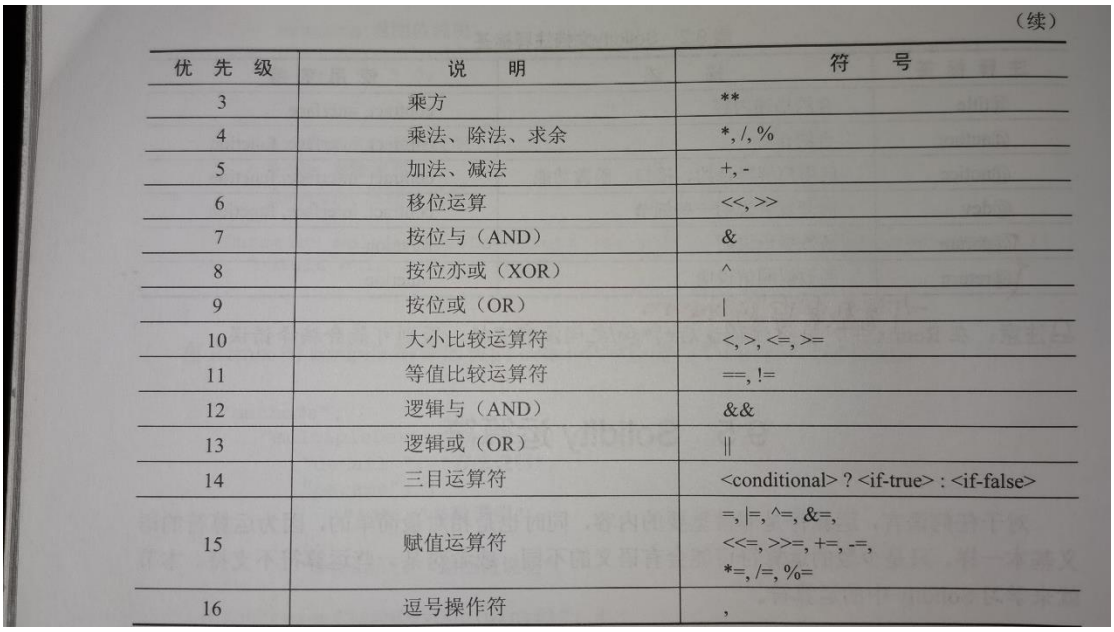
5、Solidity 运算符优先级



首先来看 Solidity 支持哪些运算符，这些运算符的优先级是怎样的。如表 9.3 所示为 Solidity 中的运算符优先级。左边第一列是运算符的优先级，优先级高的运算符先进行运算，优先级相同时从左往右依次计算。

表 9.3　Solidity运算符优先级

优　先　级	说　　明	符　　号
1	前置自增自减	++, --
	new 表达式	new <typename>
	数组下标访问	<array>[<index>]
	成员访问	<object>.<member>
	函数调用	<func>(<args...>)
	括号表达式	(<statement>)
2	后置自增自减	++, --
	一元运算符+，-	+, -
	一元运算符delete	delete
	逻辑非（NOT）	!
	按位取反（NOT）	~



（续）

优 先 级	说 明	符 号
3	乘方	**
4	乘法、除法、求余	*, /, %
5	加法、减法	+, -
6	移位运算	<<, >>
7	按位与（AND）	&
8	按位亦或（XOR）	^
9	按位或（OR）	|
10	大小比较运算符	<, >, <=, >=
11	等值比较运算符	==, !=
12	逻辑与（AND）	&&
13	逻辑或（OR）	||
14	三目运算符	<conditional> ? <if-true> : <if-false>
15	赋值运算符	=, |=, ^=, &=, <<=, >>=, +=, -=, *=, /=, %=
16	逗号操作符	,

6、控制结构

不支持 Switch、goto

7、可见性修饰符

Public 类型的状态变量和函数：本合约、外部合约、子合约都可以访问。

Internal 修饰时状态变量，外部和子合约都可以访问，其修饰的函数，只能在合约内访问。

Private 修饰的状态变量和函数，都只能在合约内部访问，子合约和外部都无法访问。

External 只能用于修饰函数，在合约内部不能直接调用，如果使用，则 this.的方式，合约外部可以调用。

Constant 修饰状态变量，该变量为常量。

View 修饰函数，替换 constant，不修改状态变量，不管函数中有没有修改状态变量的语句。

Pure 就是函数，既不修改也不读取状态变量。

8、数据类型

值类型：当用在函数参数或者赋值的时候，始终执行的都是复制操作。

引用类型：当用在函数参数或者赋值的时候，不一定都是赋引用。根据数据位置的不同有可能执行的是复制操作，也可能是赋引用。

布尔类型不支持其他类型显式或隐式的转换成布尔类型。

布尔类型在||和&&运算时的短路运算规则。

9、字符串字面量

字符串字面量用单引号或双引号，支持转义字符，是一个动态字节数组，按字节存放，存储 UTF -8 编码。如：

/n：

/xNN：表示十六进制，在内存中存放的也是十六进制

/uNNNN：Unicode 的码位，

串没有填充'\0'作为结尾。

Solidity 中的字符串字面量也支持转义字符，例如\n、\xNN 和\uNNNN 等。\xNN 表示的是十六进制值，在内存中存放的也是十六进制值。\uNNNN 表示的是 Unicode 的码位，存储的时候存储的是对应的 UTF-8 编码。下面看一个实例：

```
pragma solidity ^0.4.24;
/**
 * @title StringLiteralContract
 * @dev 字符串字面量测试
 */
contract StringLiteralContract {
    string public A = "\x41";                          //使用\x 方式
    string public unicodeZhong = "\u4e2d";             //使用 Unicode 方式
    string public zhong = "中";
    /**
     * 获取\x41 对应值
     */
    function getANum() public view returns(uint8) {
        return uint8(bytes(A)[0]);//强制转换为 bytes 类型，然后取低字节转换为 uint8
    }
}
```

如图 10.4 所示，字符串字面量其实存储的就是 UTF-8 编码，这个在字节数组中会详细介绍。

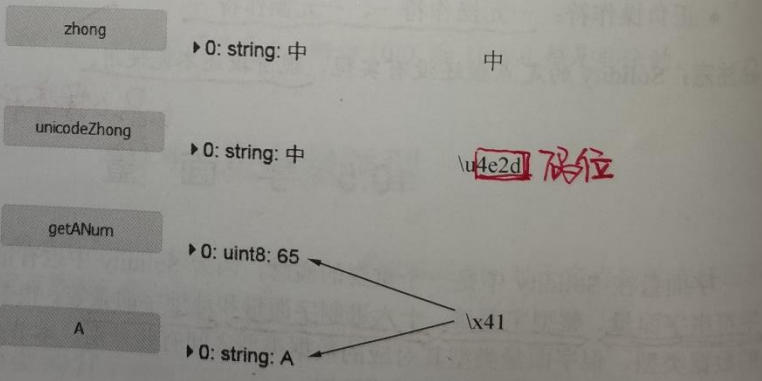


图 10.4　字符串字面量

从上面的例子中可以看到，字符串字面量“中”存放的就是 UTF-8 编码，对于 Unicode 存放的就是对应码位上的字符的 UTF-8 编码(\u4e2d 是中的 uincode 码位)。\x 表示的是十六进制，注意不能超出范围，最大是"\x7F"。

10、十六进制字面量

使用 hex 关键字作为前缀的字符串字面量，必须是十六进制格式。如：

Hex“7e7d”

\x 只能表示一个字节，hex 可以表示多个字节。

11、枚举类型（enum）

枚举类型可以强制转换成整型，但不能隐式的转换。

12、映射（mapping）

定义方式：mapping(key ➔value)

Key: 除了 mapping、动态数组、合约、枚举、结构体之外的所有类型。

Value：可以是任何类型

如果 key 存在，则获取对应的值

如果 key 不存在，则获取的值为对应 value 类型的零值。

Mapping 只能用于状态变量，或者 internal 函数的 storage 标注类型的引用。

13、struct

Struct 不能作为参数，作为返回值也只能在 internal 函数中

结构体赋值有两种方式：一，按定义的顺序传入参数，二：{}包裹起来，指定每一个元素对应的值。

14、以太坊地址

账户地址：

合约地址：

Transfer 函数会抛出异常，send 函数不会抛出异常，只返回 false。

一个合约最多可以有一个不命名的函数，其没有参数，也没有返回值。

回退函数(fallback): 当交易发送以太币给合约却不调用合约内的任何方法时，回退函数就被执行了。如果此种情况下，没有回退函数，合约就会抛出异常。

没有回退函数的合约：可以作为挖矿收益的地址和合约析构函数退回的以太币。

```
/**
*@dev 使用 call 和函数签名的方式调用 add 函数，附带参数
*/
```

```
function testCallAddTwo(ToCallContract tcc) public returns(bool){
    return address(tcc).call(bytes4(keccak256 ("add(uint8,uint8)")),1,2);
    }
/**
*@dev 测试直接通过函数名方式
*/
function testCallAddFour(ToCallContract tcc) public returns(bool){
    return address(tcc).call("add",1,2);
    }
```

15、函数类型

Msg.data 的前 4 个字节就是函数签名，计算函数签名是，数据类型的别名需要转换成相应的类型，如：uint 换成 uint256.

Function (uint8) pure returns (uint8) f

红色部分是类型，f 是该函数类型的一个变量

16、数据类型转换

隐式转换：一般是低类型向该类型转换，至于存放在高字节还是低字节，那是另外一个问题。

显式转换：大类型转换成小类型，或不同类型之间转换，通常是固定大小字节数组之间的转换，保留低位数据

使用 var 不定义类型，根据值进行类型推断，会引起隐式类型转换问题：

1、不能定义数组和 mapping

2、不能定义函数参数

3、不能定义状态变量

18、delete 运算符

把变量重置为默认值。对于固定长度数组和动态长度数组，操作结果还是有区别。

固定长度数组：长度保持不变，数据重置

动态长度数组：长度重置为 0

对于 mapping：无效操作，如:delete idName，会得到一个警告。但可以

作用在一个指定的 key 上，如:delete idName[key]，将对应值重置。

Struct：重置结构体中的每一个元素。

19、EVM 结构：

易变得数据区：

不变的数据区：代码和 storage

20、事件

事件被调用时，事件的参数被记录到交易日志中，与合约关联，永久存储在链上，但日志和事件在合约中不能被访问。

与事件相关的关键词：

Emit：触发一个事件

Anonymous：标注事件，表示不把事件签名作为第一个主题，所以被其标注的事件最多可以有 4 个 indexed 参数。

Indexed：标注参数，指定还参数会创建一个主题，作为索引。

event LogEvent(string indexed first,uint indexed seconde,uint8 third,uint8 indexed fourth);

event LogAnonymousEvent(string indexed first,uint indexed seconde,uint8 third,uint8 indexed fourth) **anonymous**;

主题：把事件索引化的数值，只要触发一个事件，就生成一个或多个主题，最多可以有 4 个。

第一个主题：事件签名

剩下 3 个主题：索引化的参数数值，如果是字符串、字节数组或数组，则主题是其 keccak256 哈希值。

21 log 函数

Log0、log1、log2、log3、log4 五个函数都是 anonymous，第一个参数是非 indexed 参数。

22.以太币单位

常用范围有三个：ether、Gwei、wei。’

23、错误处理

Assert 断言函数，程序执行计算后，检查结果是否符合预期。

Require 一般是用于检查前置条件，有两个重载函数，一个不带错误信息的，另一个可以传递条件不满足的错误信息。

Revert，用于错误处理的函数。有两个重载函数

24、ABI 编码

合约 ABI（Application Binary Interface），应用二进制接口，

表 12.6　函数ABI结构

属　　性	说　　明
type	function或者constructor可以省略
name	函数名称
constant	如果为true，那么函数不能修改区块链状态
payable	默认false，如果为true表示可以接收以太币
stateMutability	pure、view、constant、nonpayable、payable之一
inputs	对象数组，对象包含两个属性，即name和type属性
outputs	对象数组，对象包含两个属性，即name和type属性

表 12.5　事件ABI结构

属　　性	说　　明
type	始终为event
name	事件名称
inputs	对象数组，对象包含3个属性，name、type与indexed
anonymous	如果事件使用了anonymous修饰符

至表 12.9 所示为 Solidity 中的数据类型分类，表 12.7 为基本类型，表 12.8 为固定大小数组类型，表 12.9 为非固定大小类型。

表 12.7　基本类型

类　型	说　明
uint<M>	M位无符号整数，0<M<=256&&M % 8 ==0
int<M>	M位有符号整数，0<M<=256&&M % 8 ==0
address	等价于uint160
uint,int	uint256，int256的同义词或者说别名
bool	等价于uint8，被限制为0和1
fixed<M> x<N>	M位无符号定点数，8 <= M <= 256 && M % 8 ==0 && 0 < N <= 80
ufixed<M> x<N>	M位有符号定点数，8 <= M <= 256 && M % 8 ==0 && 0 < N <= 80
fixed,ufixed	fixed128x18与ufixed128x18的别名
byte<M>	M字节二进制类型，0<M<=32
function	20字节地址加上4字节函数selector，编码之后等同byte24

表 12.8　固定大小数组类型

类　型	说　明
<type>[M]	M个元素的固定大小数组

表 12.9　非固定大小类型

类　型	说　明
bytes	动态大小字节序列
string	动态大小使用UTF-8编码的Unicode字符串
<type>[]	可变大小数组

另外，多个类型还可以组合为一个元组，使用小括号包裹，元素使用逗号分隔：

(T1,T2,…Tn)其中 n>=0

元组中可以包含元组，也可以包含数组，也可以不包含任何元素（当 n=0 的情况）。

Solidity 数据类型分类：1、动态类型

2、静态类型

表 12.10 中为动态类型，其余都是静态类型。

表 12.10　ABI编码动态数据类型

类　型	说　明
bytes	动态大小字节数组
string	字符串
T[]	对于任意类型T,T[]都属于动态类型
T[n]	T是动态类型T[n]才为动态类型，n>=0
(T1,T2,T3…Tn)	只要元组中有动态类型，(T1,T2,T3…Tn)就是动态类型

表 12.11　ABI基本编码规则

类　　型	编 码 方 式
uint<M>	value对应的大端（即低地址放高字节，高地址存放低字节）十六进制编码，高位(左边)填充0字节
int<M>	value对应的补码，高位(左边)填充ff(负数)或者0字节(正数)
address	等同于uint160
bool	等同于uint8，如果是true则对应uint8的1，如果是false则对应0
bytes	字节长度len当作uint256编码，连接上value的编码，低位填充
string	string使用UTF-8编码，然后按bytes的方式编码
bytes<M>	value尾部填充0字节
fixed<M>x<N>	X * 10**N对应int256的编码
fixed	等同于fixed128x18
ufixed<M>x<N>	X * 10**对应uint256的编码
ufixed	等同于ufixed128x18

uint、int、address、bool、fixed 和 ufixed 都是转换为对应的整型处理，所以处理方式基本一致。对于 bytes、string、bytes<M>都可以处理为字节，填充方式是低位填充，填充的 0 字节为 32 字节的倍数。

25、ABI 编码应用

ABI 是用来与合约进行交互的标准方式。简单的讲，就是调用合约时，数据编码和解码的规则。在 remix 中，调试面板中的 input 显示的内容就是数据的 ABI 编码结果。

26、函数重载

合约中可以有同名的函数，但输入参数必须有不一样的地方，返回参数不能用于判定函数重载。如下所示：

```
function overloadFunction(uint8 num) public pure { }
function overloadFunction(uint256 num) public pure {}
```

调用重载函数时，应注意类型转换的问题。

27.new 关键字

New 可以用来创建合约，属于创建合约的合约。也就是说，使用 eth_getCode 方法得不到合约的代码，

28、数据位置

每一个复杂的数据类型，都有一个额外的标注，叫数据位置，表示数据是在 memory 或 storage 中，它关系到变量之间的赋值行为，必须显式的声明。

函数参数，包括输入和输出参数，都是 memory，存储在 EVM 的 memory 中。函数局部变量都是 storage；状态变量必须是 storage。

0x00- 0x3f（64 字节）：哈希方法的暂存空间

0x40- 0x5f（32 字节）：当前分配的内存大小（又名空闲内存指针），也就是下一个分配空间的开始位置。

0x60- 0x7f（32 字节）：存放动态 memory 数组的初始值。也就是说，memory 存储数据是从 0x80 开始的。

Storage 标注的变量会存储在 EVM 的 storage 区域，在交易执行后写入合约的 storage 中，storage 标注的变量有两种存储格式：

一、mapping 和动态大小数组。

1、对于 Bytes 和 string，存储方式比较简单，长度小于 31 字节时，slot 中高位存数据，低位存放长度值的 2 倍；长度超过 32 字节时，对应 slot 中存放 2*length+1，数据存放在 keecak256(slot)中，

2、对于动态数组 T[]，slot 中存放元素个数，数据在 keecak256(slot)中，左填充。

3、对于 mapping，对应的 slot 中没有存储值，数据存储在 keecak256(key+slot)的位置，key 是键，+表示拼接。注意拼接顺序。

二、静态大小类型变量

1、存储槽中的第一个项以低位对齐方式存储。

2、值类型仅使用存储它们所需的字节数。

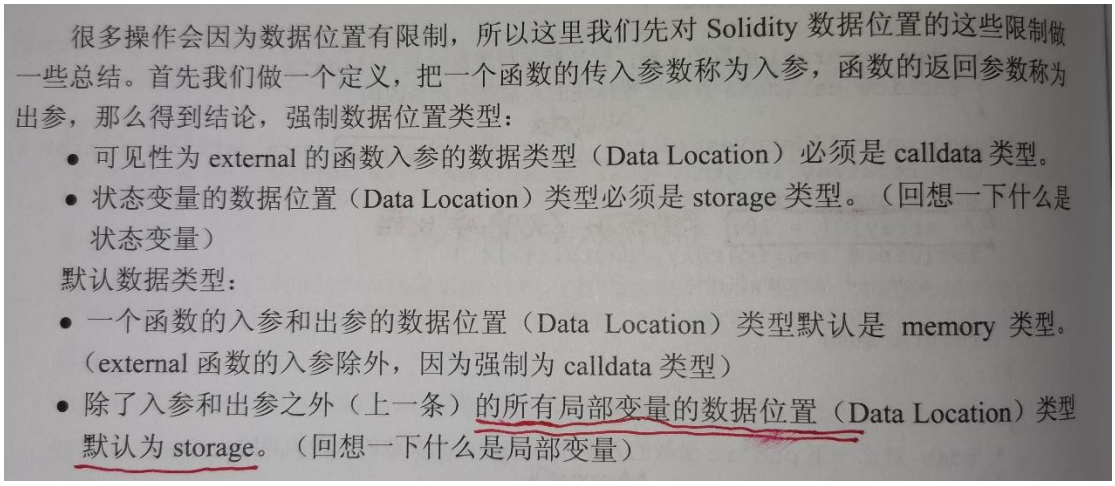
3、如果值类型不适合存储槽的剩余部分，则将其存储在下一个存储槽中。

4、结构和数组数据总是从一个新的槽开始，它们的项目根据这些规则被紧密地打包。

5、结构体或数组数据之后的项目总是开始一个新的存储槽。

29、calldata 数据位置

只用于 external 函数的输入参数，既不能修改也不能持久化。



很多操作会因为数据位置有限制，所以这里我们先对 Solidity 数据位置的这些限制做一些总结。首先我们做一个定义，把一个函数的传入参数称为入参，函数的返回参数称为出参，那么得到结论，强制数据位置类型：

- 可见性为 external 的函数入参的数据类型（Data Location）必须是 calldata 类型。
- 状态变量的数据位置（Data Location）类型必须是 storage 类型。（回想一下什么是状态变量）

默认数据类型：

- 一个函数的入参和出参的数据位置（Data Location）类型默认是 memory 类型。（external 函数的入参除外，因为强制为 calldata 类型）
- 除了入参和出参之外（上一条）的所有局部变量的数据位置（Data Location）类型默认为 storage。（回想一下什么是局部变量）

30、函数修改器

1、用于前置条件检查，

2、后置的一些清理工作

多个函数修改器在一个函数中时，在_之前的部分，执行顺序时从左往右；在_z之后的部分，执行顺序从右向左。

31、合约继承

Solidity 合约继承使用 is 关键字，合约在区块链上部署时，只部署一个合约，子合约会将父合约的代码复制到创建的合约中

向父级合约传递参数有两种参数：

1、继承时传递参数，contract LevelTwoOne is LevelOne(2)

2、构造函数中传递，contract LevelTwoTwo is LevelOne{

```
    constructor() LevelOne(31) public{
    }
```

多重继承：

```
contract LevelThreeOne is LevelOne,LevelTwoOne,LevelTwoTwo{

}
```

后面的合约会重写前面合约中的同名函数，所以被继承的合约应该写在前面。

33、super 关键词

Super 代表继承链中的上一个。

34、抽象合约

如果合约中有方法没有实现，那么他就是抽象合约。如果一个合约继承了一个

抽象合约而没有实现其中的方法，也会标记为抽象合约。

35、接口合约

接口中不能有任何实现了的方法，而抽象合约中可以有实现了的方法，此外，接口还有一些限制：

1、不能继承其他合约与接口

2、不能定义构造函数

3、不能定义变量

4、不能定义结构体

5、不能定义枚举

```
interface Token {
    function totalSupply() external view returns (uint);
    event Transfer(address indexed _from, address indexed _to, uint _value);
}
```

抽象合约而没有实现其中的方法，也会标记为抽象合约。

35、接口合约

接口中不能有任何实现了的方法，而抽象合约中可以有实现了的方法，此外，接口还有一些限制：

1、不能继承其他合约与接口

2、不能定义构造函数

3、不能定义变量

4、不能定义结构体

5、不能定义枚举

```
interface Token {
    function totalSupply() external view returns (uint);
    event Transfer(address indexed _from, address indexed _to, uint _value);
}
```

抽象合约而没有实现其中的方法，也会标记为抽象合约。

35、接口合约

接口中不能有任何实现了的方法，而抽象合约中可以有实现了的方法，此外，接口还有一些限制：

1、不能继承其他合约与接口

2、不能定义构造函数

3、不能定义变量

4、不能定义结构体

5、不能定义枚举

```
interface Token {
    function totalSupply() external view returns (uint);
    event Transfer(address indexed _from, address indexed _to, uint _value);
}
```