实践经验类(6-9)

面试问答题（中英文）

实践经验类 Practical experience category（6-9）

6、乘以或除以二的倍数的 gas 高效替代方法是什么?

What is the gas efficient alternative to multiplying or dividing by a multiple of two?

答:

在Solidity中，将一个数乘以或除以2的倍数可以通过移位运算来实现，这比使用乘法或除法运算更高效。具体来说，将一个数左移n位相当于将它乘以2的n次方，而将一个数右移n位相当于将它除以2的n次方。例如，将一个数除以8可以通过将它右移3位来实现，而将一个数乘以16可以通过将它左移4位来实现。在Solidity 0.8.3及更高版本中，编译器会自动将乘法和除法运算转换为移位运算，从而提高代码的效率。

In Solidity, multiplying or dividing a number by a multiple of two can be accomplished by shifting operations, which is more efficient than using multiply or divide operations. Specifically, shifting a number n places to the left is equivalent to multiplying it by 2 to the nth power, while shifting a number n places to the right is equivalent to dividing it by 2 to the nth power. For example, dividing a number by 8 can be accomplished by shifting it 3 places to the right, while multiplying a number by 16 can be accomplished by shifting it 4 places to the left. In Solidity 0.8.3 and higher, the compiler automatically converts multiplication and division operations to shifts, thus improving the efficiency of the code.

7、哪些操作会部分退还 gas?

What operations partially refund gas?

答:

STORE 操作：如果将一个存储槽从非零值更改为零值，则会退还一部分gas。

SLOAD 操作：如果从存储中读取一个非零值，则会退还一部分gas。

CALL 操作：如果调用的合约执行成功，则会退还一部分gas。

RETURN 操作：如果从函数中返回，则会退还一部分gas。

SELFDESTRUCT 操作：如果自毁合约，则会退还其余的gas。

需要注意的是，这些操作的退还gas的数量是固定的，具体取决于操作的类型和执行的环境。

The STORE operation: a portion of gas is refunded if a storage slot is changed from a non-zero value to a zero value.

SLOAD operation: if a non-zero value is read from storage, a portion of gas is refunded.

CALL operation: a portion of gas is refunded if the called contract executes successfully.

RETURN operation: a portion of gas is refunded if returned from a function.

SELFDESTRUCT operation: the rest of the gas is refunded if the contract self-destructs.

Note that the amount of returned gas for these operations is fixed, depending on the type of operation and the environment in which it is executed.

8、如果代理对 A 进行 delegate call，而 A 执行 address(this).balance，返回的是代理的余额还是 A 的余额?

If the agent makes a delegate call to A and A executes address(this).balance, is the balance returned for the agent or for A?

答：

返回的是代理的余额。因为在 delegate call 中，调用的A会共享代理合约的存储空间，因此 address(this) 将返回代理合约的地址，而不是被调用合约的地址。

The balance of the proxy is returned. Because in a delegate call, the calling A shares the storage space of the proxy contract, address(this) will return the address of the proxy contract, not the address of the called contract.

9、为什么 Solidity 不支持浮点数运算?

Why does Solidity not support floating point arithmetic?

答：

Solidity不支持浮点数运算是因为浮点数运算需要大量的计算资源，而以太坊虚拟机的计算资源是有限的。此外，浮点数运算可能会导致精度丢失和舍入错误，这可能会影响智能合约的正确性和安全性。为了避免这些问题，Solidity使用整数运算来代替浮点数运算。如果您需要进行浮点数运算，可以使用一些库，如FixedPoint.sol，来模拟浮点数运算。这些库使用整数运算来实现浮点数运算，从而避免了精度丢失和舍入错误的问题。

Solidity does not support floating-point arithmetic because floating-point arithmetic requires a large amount of computational resources, which are limited in the Ethernet virtual machine. In addition, floating point arithmetic can lead to precision loss and rounding errors,

which can affect the correctness and security of smart contracts. To avoid these problems, Solidity uses integer operations instead of floating-point operations. If you need to perform floating point operations, you can use libraries such as Fixed Point.sol to simulate floating point operations. These libraries use integer arithmetic to implement floating point operations, thus avoiding precision loss and rounding errors.