基础知识类21-25

面试问答题（中英文）

基础知识类（21-25）

21、什么是浮动利率和固定利率？

What are variable and fixed interest rates?

答：

固定利率是指在贷款期间内，贷款利率保持不变。而浮动利率是指贷款利率会随着市场利率的变化而变化。浮动利率通常基于某个基准利率，例如央行基准利率或LIBOR（伦敦银行间同业拆借利率）。如果基准利率上升，贷款利率也会上升，反之亦然。

send函数的gas限制也为2300 gas，但它返回一个布尔值，指示转账是否成功。如果转账失败，它将返回false。但是，如果接收方合约没有实现fallback函数，或者fallback函数消耗的gas超过了2300，那么转账将失败并回滚所有更改。

A fixed interest rate means that the interest rate on the loan remains the same for the duration of the loan. A floating rate is when the interest rate on the loan changes in response to changes in market interest rates. Floating interest rates are usually based on a benchmark rate, such as the central bank's prime rate or LIBOR (London Interbank Offered Rate). If the prime rate rises, the lending rate will also rise, and vice versa.

The send function also has a gas limit of 2300 gas, but it returns a boolean indicating whether the transfer was successful. If the transfer fails, it returns false, but if the recipient contract does not implement a fallback function, or if the fallback function consumes more than 2300 gas, then the transfer fails and all changes are rolled back.

22、transfer 和 send 之间有什么区别？为什么不应该使用它们？

What is the difference between transfer and send? Why should they not be used?

答：

transfer函数的gas限制为2300 gas，这意味着如果接收方合约没有实现fallback函数，或者fallback函数消耗的gas超过了2300，那么转账将失败并回滚所有更改。这可以防止重入攻击，但也可

能导致一些问题，例如无法向某些合约发送以太币。send函数的gas限制也为2300 gas，但它返回一个布尔值，指示转账是否成功。如果转账失败，它将返回false。但是，如果接收方合约没有实现fallback函数，或者fallback函数消耗的gas超过了2300，那么转账将失败并回滚所有更改。

由于这些限制，transfer和send函数已经被认为是不安全的，因此不应该使用它们。相反，您应该使用call函数来转移以太币。call函数没有gas限制，可以向任何地址发送以太币，并且可以指定要发送的gas数量。但是，您应该小心使用call函数，因为它可能会导致一些安全问题，例如重入攻击。

The transfer function has a gas limit of 2300 gas, which means that if the recipient contract does not implement a fallback function, or if the fallback function consumes more than 2300 gas, the transfer will fail and all changes will be rolled back. This prevents reentry attacks, but can also lead to problems such as not being able to send Ether to some contracts. the send function also has a gas limit of 2300 gas, but it returns a boolean indicating whether the transfer was successful or not. If the transfer fails, it returns false. however, if the recipient contract does not implement the fallback function, or if the fallback function consumes more than 2300 gas, the transfer will fail and all changes will be rolled back.

Due to these limitations, the transfer and send functions have been deemed unsafe and therefore they should not be used. Instead, you should use the call function to transfer Ether. call functions do not have a gas limit, can send Ether to any address, and can specify the amount of gas to send. However, you should be careful with the call function as it can lead to some security issues, such as reentry attacks.

23、如何在 Solidity 中编写节省gas的高效循环?

How to write efficient loops that save gas in Solidity?

答：

1.避免无限循环。

2.避免重复计算：多次计算相同的值，应该将该值存储在变量中，并在需要时重复使用该变量。

3.避免使用昂贵的操作：例如除法和取模。尝试使用更便宜的操作来代替它们。

4.避免使用大型数组：如果可能的话，您应该使用映射或其他数据结构来代替数组。

5.避免使用复杂的嵌套循环。

6.使用constant和immutable关键字：如果您需要在循环中使用常量或不可变变量，请使用constant和immutable关键字来声明它们。

1. Avoid infinite loops.

2. Avoid repeated calculations: to calculate the same value multiple times, you should store the value in a variable and reuse the variable when needed.

3. Avoid using expensive operations: such as division and modulo. Try using cheaper operations instead of them.

4. Avoid large arrays: If possible, you should use maps or other data structures instead of arrays.

5. avoid complex nested loops.

6. Use the constant and immutable keywords: if you need to use constants or immutable variables in a loop, declare them using the constant and immutable keywords.

24、uint8、uint32、uint64、uint128、uint256 都是有效的 uint 大小。还有其他的吗?

uint8, uint32, uint64, uint128, uint256 are all valid uint sizes. Are there others?

答:

这些类型分别表示8位、32位、64位、128位和256位的无符号整数。除了这些类型之外，Solidity还支持其他整数类型，例如int8、int16、int32、int64、int128和int256。这些类型分别表示8位、16位、32位、64位、128位和256位的有符号整数。

These types represent 8-bit, 32-bit, 64-bit, 128-bit, and 256-bit unsigned integers, respectively. In addition to these types, Solidity supports other integer types such as int8, int16, int32, int64, int128, and int256, which represent 8-bit, 16-bit, 32-bit, 64-bit, 128-bit, and 256-bit signed integers.

25、以太坊如何确定 EIP-1559 中的 BASEFEE?

How does Ethernet determine BASEFEE in EIP-1559?

答:

在以太坊的EIP-1559协议中，BASEFEE是由以太坊网络根据交易需求和区块大小动态调整的。BASEFEE的计算方式是通过一个名为“基础费用追踪器”的算法来实现的。该算法会根据当前区块的交易需求和区块大小来动态调整BASEFEE，以确保交易能够在合理的时间内得到确认。

具体来说，基础费用追踪器会根据当前区块的交易需求和区块大小计算出一个目标基础费用。如果当前的BASEFEE高于目标基础费用，那么BASEFEE将会下降。如果当前的BASEFEE低于目标基础费用，那么BASEFEE将会上升。这种动态调整机制可以确保BASEFEE始终能够反映当前的交易需求和区块大小，从而提高以太坊的交易效率和可靠性。

In Ethernet's EIP-1559 protocol, BASEFEE is dynamically adjusted by the Ethernet network based on transaction demand and block size, and is calculated using an algorithm called the "Base Fee Tracker". The algorithm dynamically adjusts BASEFEE based on the current block's transaction

demand and block size to ensure that transactions are recognized in a reasonable amount of time.

Specifically, the BASEFEE tracker calculates a target BASEFEE based on the transaction demand and block size of the current block. If the current BASEFEE is higher than the target base fee, then the BASEFEE will decrease. If the current BASEFEE is lower than the target base fee, then the BASEFEE will increase. This dynamic adjustment mechanism ensures that BASEFEE always reflects the current transaction demand and block size, thus improving the efficiency and reliability of Ethernet transactions.