**Programmazione in Solidity**

La programmazione in Solidity, il linguaggio di contratto intelligente per Ethereum e altre piattaforme blockchain, segue i principi di programmazione tipici, ma con alcune peculiarità dovute alla natura decentralizzata e alla sicurezza richiesta. Di seguito, ti fornisco una panoramica della logica generale nella programmazione in Solidity:

1. \*\*Definizione dei contratti\*\*: I contratti sono le entità principali in Solidity. Sono simili a classi in altri linguaggi di programmazione e contengono lo stato e le funzioni che gestiscono lo stato. I contratti sono definiti utilizzando la parola chiave `contract`.

2. \*\*Gestione dello stato\*\*: Solidity consente di definire variabili di stato all'interno dei contratti. Queste variabili rappresentano lo stato del contratto e possono essere di diversi tipi di dati, come interi, stringhe, array e strutture. Il loro valore viene memorizzato sulla blockchain.

3. \*\*Funzioni\*\*: I contratti Solidity possono contenere funzioni che possono essere chiamate per interagire con lo stato del contratto. Le funzioni possono essere di diversi tipi, ad esempio pubbliche, private, interne o esterne. Le funzioni possono modificare lo stato del contratto o restituire valori.

4. \*\*Modificatori\*\*: I modificatori sono utilizzati per modificare il comportamento delle funzioni all'interno dei contratti. Possono essere utilizzati per applicare logica comune a più funzioni, verificare condizioni o restrizioni prima dell'esecuzione di una funzione.

5. \*\*Eventi\*\*: Gli eventi sono utilizzati per registrare e comunicare gli eventi che si verificano all'interno del contratto. Possono essere emessi durante l'esecuzione di determinate operazioni e registrano i dettagli dell'evento sulla blockchain.

6. \*\*Pagamenti\*\*: Solidity consente il trasferimento di criptovaluta da un contratto all'altro o da un utente a un contratto e viceversa. Questi pagamenti possono essere gestiti utilizzando funzioni di pagamento o specificando modificatori di pagamento.

7. \*\*Gestione degli errori\*\*: Solidity offre diverse modalità per gestire gli errori e le eccezioni. Questo include l'uso di assert, require e revert per interrompere l'esecuzione del contratto in caso di errori o violazioni delle condizioni.

8. \*\*Test e debugging\*\*: È importante testare e debuggare i contratti Solidity prima di distribuirli sulla blockchain. Questo può essere fatto utilizzando framework di test come Truffle o utilizzando ambienti di sviluppo come Remix.

Questi sono solo alcuni dei concetti fondamentali della programmazione in Solidity. La programmazione in Solidity presenta anche altre caratteristiche e sfide uniche, come la sicurezza dei contratti e la gestione delle risorse sulla blockchain.

**Costruttore**

Il costruttore è una funzione speciale presente in un contratto Solidity che viene eseguita una sola volta quando il contratto viene creato. La sua principale funzione è inizializzare lo stato del contratto o eseguire altre operazioni necessarie all'avvio del contratto.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract MyContract

{

uint256 public myVariable;

// Costruttore che inizializza myVariable con un valore iniziale

constructor(uint256 initialValue)

{

myVariable = initialValue;

}

}

**Iniziare a Programmare:**

In questa parte degli appunti, ho preso esercizi randomici dal web e gli ho implementati in linguaggio solidity grazie all’utilizzo di REMIX.

* **Esercizio 1:**

Mostrare a video la scritta “Hello Word”

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

* **Esercizio 2:**

Creare delle funzioni per operazioni matematiche

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

* **Esercizio 3:**

Creare uno script in cui si può registrare un valore e memorizzarlo, per poi visualizzarlo grazie ad un'altra funzione.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente