山东大学 区块链原理 课程作业报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202200150 | 姓名： 王泓然 | | 班级： 经济学院 |
| 作业题目：空投合约 | | | |
| 作业学时：4 | | 完成日期： 2024.12.27 | |
| 原理分析与步骤：  Solidity是一种面向对象的高级编程语言，专门用于编写和部署智能合约，特别是在以太坊区块链上。智能合约是自动执行的合约，合约条款直接写入代码中，能够在满足特定条件时自动执行，无需第三方干预。  Solidity的特点  1.静态类型系统：Solidity使用静态类型系统，能够在编译时检测类型错误，提高了合约的安全性。  2.面向对象：支持继承、库和复杂的用户定义类型，使代码更加模块化和可重用。  3.内嵌支付功能：通过关键字`payable`，开发人员可以直接在语言层面支持支付功能。  4.与以太坊集成：Solidity紧密集成于以太坊平台，可以无缝编写与以太坊区块链交互的智能合约。  智能合约，是一段写在区块链上的代码，一旦某个事件触发合约中的条款，代码即自动执行。也就是说，满足条件就执行，不需要人为操控。  ERC-20 标准为以太坊区块链上的加密代币功能制定了一个全面的框架，将操作分类为 getter、函数和事件，以确保生态系统内集成的一致性和易用性。  Getters 的目的是在不改变区块链状态的情况下检索和显示数据。 ERC-20 中概述的主要包括：  总供应量：此函数报告已发行的代币总数，提供对特定代币流通规模的深入了解。  Balance Of ：返回特定账户的代币余额，让用户轻松验证其持有情况。  津贴：这一独特功能促进了委托支出，其中一个账户可以授权另一个账户代表其支出指定的代币金额。例如，如果用户 A 授权用户 B 使用 50 个代币，则用户 B 可以使用这些代币进行交易，最多可达分配的金额，但不能超过。  函数是面向操作的命令，可实现代币管理和传输：  转账：这一核心功能用于将代币从一个账户转移到另一个账户，这是代币流通的一个基本方面。  批准：它允许代币持有者指定另一个帐户的支出限额，从而实现以太坊网络内的自动支付和津贴等场景。  转账来源：基于“批准”功能，允许第三方在批准的限额内在账户之间转移代币，从而简化涉及多方的交易。  事件是[智能合约](https://plisio.net/zh/blog/smart-contracts-their-role-and-operation-in-blockchain)发出的信号，表明已发生重大操作，提供透明度和可追溯性：  转移事件：每当转移代币时都会触发该事件，该事件会记录交易，提供代币移动的可见性和验证。  批准事件：当一个帐户批准另一个帐户花费特定代币金额时，会发出此事件，作为对委托权限的公开确认。  除了核心功能之外，值得注意的是 ERC-20 在促进以太坊上去中心化应用程序 (dApp) 之间的互操作性方面的重要性。通过遵守一套标准化规则，ERC-20 代币可以轻松集成到钱包、交易所和其他 dApp 中，从而增强整个生态系统的流动性和实用性。此外，该标准为创新的金融应用程序和协议铺平了道路，通过支持从简单的转账到复杂的智能合约执行等广泛的交易，为去中心化金融（DeFi）的发展做出了重大贡献。作为其基础作用的证明，ERC-20 标准继续影响新代币标准和区块链技术的开发，突显其对以太坊区块链和更广泛的加密领域的关键影响。  空投是加密货币社区中的一种营销策略，项目经理会将令牌免费分发给特定的用户组。为了获得空投项目的资格，用户通常需要完成一些简单的任务，如测试产品、分享新闻、向朋友介绍项目等。项目可以通过空投来获得种子用户，用户也可以获得一笔财富，这是两者中最好的选择。因为每次都有很多用户收到空投信，所以项目经理不可能逐个转账。使用智能合约分批分发ERC20标记，可以显著提高空投的效率。 | | | |
| 结论分析与体会：  **实验思路**  **详见代码备注**  **合约测试** | | | |

附录：程序源代码

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.4;

import "./IERC20.sol";  // 导入IERC20接口，IERC20接口是ERC20代币的标准接口

/// @notice 向多个地址转账ERC20代币的空投合约

contract Airdrop {

    /// @notice 向多个地址转账ERC20代币

    /// @param \_token 转账的ERC20代币合约地址

    /// @param \_addresses 空投地址数组

    /// @param \_amounts 每个地址的空投代币数量数组

    function multiTransferToken(

        address \_token,

        address[] calldata \_addresses,

        uint256[] calldata \_amounts

    ) external {

        // 声明IERC20合约变量，初始化ERC20代币合约

        IERC20 token = IERC20(\_token);

        // 校验：\_addresses和\_amounts数组的长度必须相同

        require(\_addresses.length == \_amounts.length, "Lengths of Addresses and Amounts are NOT EQUAL");

        uint256 totalAmount = getSum(\_amounts);  // 获取空投的代币总量

        // 校验：合约的余额足够进行空投

        require(token.balanceOf(address(this)) >= totalAmount, "Insufficient token balance");

        // 循环发送代币到各个地址

        for (uint256 i = 0; i < \_addresses.length; i++) {

            // 调用ERC20的transfer函数向每个地址转账指定数量的代币

            token.transfer(\_addresses[i], \_amounts[i]);

        }

    }

    /// @notice 计算数组的总和

    /// @param \_arr 数字数组

    /// @return sum 数组的总和

    function getSum(uint256[] calldata \_arr) public pure returns(uint sum) {

        // 遍历数组并求和

        for (uint256 i = 0; i < \_arr.length; i++) {

            sum += \_arr[i];

        }

    }

}