pragma solidity ^0.8.0;

contract Voting {

// 合约创建者

address public owner;

// 投票人结构体

struct Voter {

bool isVoter;

uint256 voteCount;

}

// 候选人结构体

struct Candidate {

uint256 id;

string name;

uint256 voteCount;

}

// 存储候选人ID和其信息的映射

mapping(uint256 => Candidate) public candidates;

// 存储投票人地址和其信息的映射

mapping(address => Voter) public voters;

// 候选人数量

uint256 public candidateCount;

// 事件

event CandidateAdded(uint256 indexed candidateId, string candidateName);

event VoterAdded(address indexed voter);

event Voted(address indexed voter, uint256 indexed candidateId);

// 构造函数，设置合约创建者

constructor() {

owner = msg.sender;

}

// 仅合约创建者可调用的修饰符

modifier onlyOwner() {

require(msg.sender == owner, "Only the contract owner can call this function.");

\_;

}

// 添加候选人

function addCandidate(string memory \_name) public onlyOwner {

candidateCount++;

candidates[candidateCount] = Candidate(candidateCount, \_name, 0);

emit CandidateAdded(candidateCount, \_name);

}

// 添加投票人

function addVoter(address \_voter) public onlyOwner {

require(!voters[\_voter].isVoter, "The address is already a voter.");

voters[\_voter].isVoter = true;

emit VoterAdded(\_voter);

}

// 为指定候选人投票

function vote(uint256 \_candidateId) public {

require(voters[msg.sender].isVoter, "You are not an authorized voter.");

require(!voters[msg.sender].hasVoted, "You have already voted.");

require(candidates[\_candidateId].id != 0, "Invalid candidate ID.");

voters[msg.sender].hasVoted = true;

candidates[\_candidateId].voteCount++;

emit Voted(msg.sender, \_candidateId);

}

// 获取投票数

function getVoteCount(address \_voter) public view returns (uint256) {

return voters[\_voter].voteCount;

}

// 获取所有投票人的投票数

function getAllVotersVoteCount() public view returns (address[] memory, uint256[] memory) {

uint256 numberOfVoters = voterList.length;

address[] memory voterAddresses = new address[](numberOfVoters);

uint256[] memory voterVoteCounts = new uint256[](numberOfVoters);

for (uint256 i = 0; i < numberOfVoters; i++) {

voterAddresses[i] = voterList[i];

voterVoteCounts[i] = voters[voterList[i]].voteCount;

}

return (voterAddresses, voterVoteCounts);

}

}

|  |
| --- |
| 投票合约 |
| Input: candidateId  Output: VotedResult |
| If voters[msg.sender].isVoter != true then  |return “Not authorized voter”  Else if voters[msg.sender].hasVoted == true then  |return “you have already voted”  Else if candidates[\_candidateId].id == 0 then  | return “Invalid candidate Id”  Else  Voters[msg.sender].hasVoted = true 将投票标志为设置为true  Candidates[\_candidateId].voteCount++ 将被投票人投票数加  Emit Voted(msg.sender,\_candidateId) 触发投票成功事件 |
|  |
|  |
|  |
|  |
|  |