

数据结构

```
struct project {
    uint256 id;
    string name;
    address creator;
    mapping(address => contributor) contributors;

    address [] allContributors;           // 项目所有的贡献者
    uint256 weiPerContribution;           // 每获取一贡献度需要贡献的金额
    uint256 minEntry;                     // 项目最低准入
    uint256 totalContribution;             // 项目总贡献度
    uint256 profit;                       // 项目收益余额
    uint256 codeNums;
    uint256[] codePrice;
}

struct contributor {
    address adr;
    uint256 contribution;                 // 总贡献度
    uint256 balance;                      // 未执行贡献度
    uint256 credit;                       // 信誉分
    uint256 creditRating;                 // 信用等级
    uint256 lastBounsTime;
    uint256 obtainedBonus;                // 已获得Bonus
    bool isIn;
    uint256 joinTime;                     // 用户加入项目的时间

    bool[] codeEntry;
}
```

- 在project对象中持有一个所有贡献者地址的列表，以及project的一些信息，例如id、name
- minEntry是指用户开始项目时，需要的最小贡献度，实际可以通过公式获得，在此简化为固定值
- codeNums是指当前项目内包含的代码块数量，代码兑换以代码块为单位，具体代码块大小、内容均简化不做考量
- codePrice数组用于存放每个代码块所需要的贡献值，在此也简化为一个值，可以通过函数修改
- contributor的codeEntry数组用于标识该项目参与者是否具有下标代码块的访问权限

方法功能

- createProject方法用于创建项目
 - 在合约中的全局变量projectId是当前创建的项目数量，每一次调用该方法都会+1
 - 创建项目时需指明名称、贡献度单价、和加入项目所需要的最小贡献度余量（后续称为最小准入
 - 同时会将调用该方法的账户作为项目创始人并将其加入项目

```
function createProject(string memory _name, uint256 weiPerContribution, uint256 min_entry)
{
    uint256 id = projectId;
    projects[id].id = id;
    projects[id].name = _name;
    projects[id].creator = msg.sender;
    projects[id].weiPerContribution = weiPerContribution;
    projects[id].minEntry = min_entry;

    //把创建者加入项目
    projects[id].allContributors.push(msg.sender);
    projects[id].contributors[msg.sender].adr = msg.sender;
    projects[id].contributors[msg.sender].isIn = true;
    projects[id].contributors[msg.sender].joinTime = block.timestamp;
    projectId++;
}
```

- getProjectInfo方法用于获取目标id项目信息
 - 该方法主要用于查看项目创建是否无误

```
//获得项目信息
function getProjectInfo(uint256 project_id) public view returns
(uint id, string memory name, address creator, address [] memory allContributors,
uint256 weiPerContribution, uint256 minEntry, uint256 totalContribution,
uint256 profit ,uint codeNums){
    project storage example=projects[project_id];
    return (example.id, example.name, example.creator, example.allContributors,
    example.weiPerContribution, example.minEntry, example.totalContribution,
    example.profit,example.codeNums);
}
```

- buyContribution方法用于购买指定项目id下的贡献度
 - 该方法仅限当前账户已经加入该项目时使用
 - 调用时发送以太币，会转化为当前项目下的贡献度
 - 该方法会返回该账户加入项目的初始贡献度

```
//购买贡献度
function buyContribution(uint256 project_id) public payable returns(uint){
    project storage pro= projects[project_id];
    require(pro.contributors[msg.sender].isIn,"join the project first");
    uint256 contriToBuy = msg.value / pro.weiPerContribution;
    payable(address(this)).transfer(msg.value);
    pro.totalContribution += contriToBuy;
    projects[project_id].contributors[msg.sender].contribution += contriToBuy;
    projects[project_id].contributors[msg.sender].balance += contriToBuy;
    return contriToBuy;
}
```

- joinProject方法用于将当前账户加入指定id项目
 - 调用该方法的账户需要在调用时发送以太币，且不低于（贡献度单价*最低准入）
 - 该方法会返回当前项目下总贡献度

```

//加入项目
function joinProject(uint256 project_id) public payable returns(address[] memory){
    uint256 contriToBuy = msg.value / projects[project_id].weiPerContribution;
    require(contriToBuy >= projects[project_id].minEntry,"buy more contribution");
    payable(address(this)).transfer(msg.value);
    projects[project_id].allContributors.push(msg.sender);
    projects[project_id].totalContribution += contriToBuy;
    projects[project_id].contributors[msg.sender].adr = msg.sender;
    projects[project_id].contributors[msg.sender].isIn = true;
    projects[project_id].contributors[msg.sender].joinTime = block.timestamp;
    projects[project_id].contributors[msg.sender].contribution = contriToBuy;
    projects[project_id].contributors[msg.sender].balance = contriToBuy;
    for(uint i = 0;i < projects[project_id].codeNums;i++)
    projects[project_id].contributors[msg.sender].codeEntry.push(false);
    return projects[project_id].allContributors;
}

```

- initCodePrice和changePrice用于初始化和改变项目中代码块所需贡献度
 - 初始化时需指定当前项目下有多少个代码块，并且默认每个代码块所需贡献度一致
 - 会将当前项目下所有贡献者的所有代码块访问权限置为不可访问，并给项目代码块数量赋值
 - 改动合法代码块所需贡献度时会将其更改为指定价格

```

//初始化代码访问价格
function initCodePrice(uint256 project_id,uint256 nums,uint256 price) public {
    for(uint i = 0;i < nums;i++){
        projects[project_id].codePrice.push(price);
        address[] memory contributorsAddr = projects[project_id].allContributors;
        for(uint j=0;j<contributorsAddr.length;j++){
            //当前贡献者地址
            address curContriAddr = contributorsAddr[j];
            projects[project_id].contributors[curContriAddr].codeEntry.push(false);
        }
    }
    projects[project_id].codeNums = nums;
}

//修改代码访问价格
function changePrice(uint256 project_id,uint256 pos, uint256 price) public {
    require(projects[project_id].codePrice.length > pos,"overflow");
    projects[project_id].codePrice[pos] = price;
}

```

- addCode方法可以给项目增加代码块
 - 每一次调用增加一个代码块，并且指定所需贡献度
 - 每一次调用会默认将所有贡献者置为不可访问
 - 修改对应项目属性

```
//添加代码
function addCode(uint price,uint project_id) public {
    project storage pro=projects[project_id];
    address[] memory contributorsAddr = pro.allContributors;
    pro.codeNums++;
    pro.codePrice.push(price);
    for(uint i=0;i<contributorsAddr.length;i++){
        //当前贡献者地址
        address curContriAddr = contributorsAddr[i];
        pro.contributors[curContriAddr].codeEntry.push(false);
    }
}
```

- buyCodeEntry方法用于交换代码访问权限
 - 会事先进行判断，包括判断该账户是否在项目中、目标访问代码块是否存在、用于兑换访问权限的贡献度是否足够，均满足之后即可成功兑换

```
//交换代码访问权限
function buyCodeEntry(uint256 project_id,uint256 goalCode) public {
    require(projects[project_id].contributors[msg.sender].isIn,"not in project");
    require(goalCode < projects[project_id].codeNums,"access invalid");
    require(projects[project_id].contributors[msg.sender].balance >=
    projects[project_id].codePrice[goalCode],"your contribution is not enough");
    projects[project_id].contributors[msg.sender].balance -=
    projects[project_id].codePrice[goalCode];
    projects[project_id].contributors[msg.sender].codeEntry[goalCode] = true;
}
```

- accessCode方法用于检验是否交换访问权限成功
 - 若检验成功，会返回当前账户的可兑换贡献度余量

```
//查看是否购买成功
function accessCode(uint256 project_id,uint256 goalCode) public view returns(uint256 balance){
    require(projects[project_id].contributors[msg.sender].isIn,"not in project");
    require(goalCode < projects[project_id].codeNums,"access invalid");
    require(projects[project_id].contributors[msg.sender].codeEntry[goalCode],"not accessible");
    return projects[project_id].contributors[msg.sender].balance;
}
```

具体过程

- 使用remix
- 首先使用账号0创建一个项目

createProject

_name:

weiperContribution:

min_entry:

Calldata ...

[vm] from: 0x5B3...eddC4 to: contributionUse.createProject(string,uint256,uint256) 0x5c4...165CF
value: 0 wei data: 0xf3a...00000 logs: 0 hash: 0xaa1...1254b

- 初始化项目内代码块数量和价格
 - 设为两个代码块，每个代码块价格为1贡献度，可通过changePrice方法更改

initCodePrice

project_id:

nums:

price:

Calldata ...

[vm] from: 0x5B3...eddC4 to: contributionUse.initCodePrice(uint256,uint256,uint256) 0xEc2...cF142
value: 0 wei data: 0xe2f...00001 logs: 0 hash: 0xdc1...ec9a2

- 切换账号1加入该项目，因为是第一个项目，项目id为0
 - 同时发送2Wei用于购买准入

joinProject

project_id: 0

Calldata ...

transact

[vm] from: 0xAb8...35cb2 to: contributionUse.joinProject(uint256) 0x5c4...165CF value: 2 wei data: 0xdf9...00000 logs: 0 hash: 0xd31...f7757 Debug

- 目前账户足够兑换代码访问权限

- 先兑换第一块代码块

buyCodeEntry

project_id: 0

goalCode: 0

Calldata ...

transact

[vm] from: 0xAb8...35cb2 to: contributionUse.buyCodeEntry(uint256,uint256) 0xEc2...cF142 value: 0 wei data: 0xbd7...00000 logs: 0 hash: 0x2fe...194c3 Debug

- 调用accessCode方法检测一下是否兑换成功

accessCode

project_id: 0

goalCode: 0

Calldata ...

call

- 成功返回balance

```
decoded output
{
  "0": "uint256: balance 1"
}

logs
[]
```

- 切换回账号0，加入一个所需贡献度为2的代码块（这个代码块下标为2）

addCode

price: 2

project_id: 0



Calldata ...

transact

[vm] from: 0x5B3...eddC4 to: contributionUse.addCode(uint256,uint256) 0xEc2...cF142 value: 0 wei data: 0xa60...00000 logs: 0 hash: 0x794...44e1b



- 切换为账号1，此时试图获取该代码块访问权限，因当前余量贡献度为1，不够2，不能兑换成功
 - 显示贡献度余额不够

- 账号1再去购买一点贡献度
 - 发送1Wei使用buyContribution方法，此时账户1可兑换贡献度余量为2

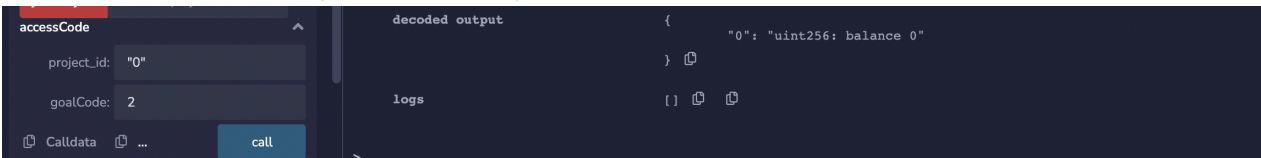
-  [vm] from: 0xAb8...35cb2 to: contributionUse.buyContribution(uint256) 0xEc2...cF142 value: 1 wei data: 0x72c...00000 logs: 0 hash: 0xd5e...d22ee Debug 

- 使用账号1再次获取新代码块访问权限

- 兑换成功

-  [vm] from: 0xAb8...35cb2 to: contributionUse.buyCodeEntry(uint256,uint256) 0xEc2...cF142 value: 0 wei data: 0xbd7...00002 logs: 0 hash: 0x690...4b779 Debug 

- 使用accessCode方法检查，的确兑换成功，并且此时可兑换贡献度余量用完

-  The image shows a web interface for the 'accessCode' method. On the left, there are input fields for 'project_id' (containing '0') and 'goalCode' (containing '2'). Below these are buttons for 'Calldata', '...', and 'call'. On the right, the 'decoded output' is shown as a JSON object: {"0": "uint256: balance 0"}. Below that, the 'logs' section is empty, showing an array of logs.