به نام خدا

گزارش تمرین دوم درس برنامه سازی پیشرفته

استاد: دکتر جهانشاهی

تدریسیار: کیان بهزاد

محمدمهدی مالوردی-9723079

توابع کلاس سرور را به صورت زیر تعریف می کنیم:

```cpp
#include "server.h"


Server::Server():
clients{}
{}

std::shared_ptr<Client> Server::add_client(std::string id){


std::random_device rd;
std::mt19937 mt(rd());
std::uniform_int_distribution<> dist(1000, 9999);

std::string temp{id};
for(auto member = clients.begin();member != clients.end();member++){
    if(member->first->get_id() == id){
        temp += std::to_string(dist(mt));
    }

 }
 Client cli(temp, *this);
 std::shared_ptr<Client> cl_shp = std::make_shared<Client>(cli);

 clients.insert({cl_shp,5});
 return cl_shp;

}

std::shared_ptr<Client> Server::get_client(std::string id) const{

std::shared_ptr<Client> cl_shp{nullptr};

    for(auto member = clients.begin();member != clients.end();member++){
        if(member->first->get_id() == id){
            cl_shp = member->first;
        }
    }
 return cl_shp;
}
```

```cpp
double Server::get_wallet(std::string id){

double amount{};

    for(auto member = clients.begin();member != clients.end();member++){
        if(member->first->get_id() == id){
            amount = member->second;
        }
    }
  return amount;


}

bool Server::parse_trx(std::string trx, std::string &sender, std::string &receiver, double &value){

    size_t index1 = trx.find("-");
    size_t index2 = trx.find("-",index1+1);


    if(index1 == std::string::npos || index2 == std::string::npos){
        throw std::runtime_error("Error!!!!");
    }

    sender = trx.substr(0,index1);
    receiver = trx.substr(index1+1,(index2-index1)-1);
    value = std::stod(trx.substr(index2+1));


    return true;

}
```

```cpp
bool Server::add_pending_trx(std::string trx, std::string signature){


std::string sender{},receiver{};
double value{};
parse_trx(trx, sender, receiver, value);
std::shared_ptr<Client> senderClient{get_client(sender)},receiverClient{get_client(receiver)};
bool authentic = crypto::verifySignature(senderClient->get_publickey(), trx, signature);

if(senderClient == nullptr || receiverClient == nullptr){
    return false;
}


return ((senderClient->get_wallet() >= value) && authentic);


}

size_t Server::mine(){

size_t nonce{};
std::string sender{},receiver{};
std::shared_ptr<Client> clie{},senderClient{},receiverClient{};
double value{};
std::string mempool{};
    for(const auto& trx : pending_trxs)
        mempool += trx;


```

```cpp
size_t Server::mine(){

size_t nonce{};
std::string sender{},receiver{};
std::shared_ptr<Client> clie{},senderClient{},receiverClient{};
double value{};
std::string mempool{};
    for(const auto& trx : pending_trxs)
        mempool += trx;


for(size_t i{};i<pending_trxs.size();){

    parse_trx(pending_trxs[i],sender,receiver,value);
    clie = get_client(sender);
    nonce = clie->generate_nonce();
    std::string hash = crypto::sha256(mempool + std::to_string(nonce));
    if(hash.substr(0, 10).find("000") != std::string::npos){
        clients[clie] += 6.25;
        std::cout<<"winner id: "<<clie->get_id()<<std::endl;
        break;
    }
    i++;
    if(i == pending_trxs.size()){
        i = 0;
    }

}

for(size_t i{}; i<pending_trxs.size(); i++){
parse_trx(pending_trxs[i], sender, receiver, value);
senderClient = get_client(sender);
receiverClient = get_client(receiver);
clients[senderClient] -= value;
clients[receiverClient] += value;

}

pending_trxs.clear();
return nonce;

}

```

توابع کلاس client را به صورت زیر تعریف می کنیم:

```cpp
#include "client.h"

Client::Client(std::string id, const Server& server):
id{id},
server{&server}
{
    crypto::generate_key(public_key, private_key);

}


std::string Client::get_id(){return id;}
std::string Client::get_publickey() const{return public_key;}
double Client::get_wallet(){

    Server svr{*server};

    return svr.get_wallet(id);
}

std::string Client::sign(std::string txt) const{

    return crypto::signMessage(private_key, txt);
}
bool Client::transfer_money(std::string receiver, double value){

Server svr{*server};
std::string trx{};
trx = id + "-" + receiver + "-" + std::to_string(value);
if(svr.add_pending_trx(trx,sign(trx))){

    pending_trxs.push_back(trx);
    return true;

}

return false;

}

size_t Client::generate_nonce(){

std::random_device rd;
std::mt19937 mt(rd());
std::uniform_int_distribution<> dist(0, 1000);

return dist(mt);

}
```

Server.h به صورت زیر می باشد:

```h
#ifndef SERVER_H
#define SERVER_H

#include <memory>
#include "client.h"
#include <map>
#include <random>
class Client;
class Server
{
public:
    Server();
    std::shared_ptr<Client> add_client(std::string id);
    std::shared_ptr<Client> get_client(std::string id) const;
    double get_wallet(std::string id);
    static bool parse_trx(std::string trx, std::string &sender, std::string &receiver, double &value);
    bool add_pending_trx(std::string trx, std::string signature);
    size_t mine();
private:
    std::map<std::shared_ptr<Client>, double> clients;
};

inline std::vector<std::string> pending_trxs;



#endif //SERVER_H
```

Client.h نیز به صورت زیر می باشد:

```h
#ifndef CLIENT_H
#define CLIENT_H

#include <string>
#include "server.h"
#include "crypto.h"
#include <map>

class Server;
class Client
{
public:
    Client(std::string id, const Server& server);
    std::string get_id();
    std::string get_publickey() const;
    double get_wallet();
    std::string sign(std::string txt) const;
    bool transfer_money(std::string receiver, double value);
    size_t generate_nonce();
private:
    Server const* const server;
    const std::string id;
    std::string public_key;
    std::string private_key;
};

static void  show_wallets(const  Server& server)
 {

    std::map<std::shared_ptr<Client>, double>* ptr = (std::map<std::shared_ptr<Client>, double>*)&server;
    std::cout << std::string(20, '*') << std::endl;
     for(auto member = ptr->begin();member != ptr->end();member++){
        std::cout << member->first->get_id() <<  " : "  << member->second << std::endl;
     }

    std::cout << std::string(20, '*') << std::endl;
 }

#endif //CLIENT_H
```

برای تولید عددهای رندوم، از همان روش گفته شده در تمرین یک استفاده کردیم.

مطابق تصویر، نیاز شد تا به بعضی از توابع، const، static و inline اضافه کنیم و همچنین برای تصحیح تابع show_wallets روشی به کمک پوینتر برای دسترسی خارج از کلاس به متغیر پرایوت داخل کلاس استفاده کردیم.(مطابق آخرین شکل)

برای پیاده سازی تابع mine ، از کدهای تست 15 نیز کمک گرفتیم.

در نهایت می بینیم که همه 15 تست پاس شدند.

آدرس تمرین در گیت : https://github.com/mahdimld/AP-HW02