Лабораторная работа #1, 2 «Программирование сетевых серверов и клиентов» Вариант #1	Выполнил	Ноздренков С.В.
	Группа	ЭВМ-1.Н
	Проверил	Жариков Д. Н.
	Подпись	

Цель работы

Изучение транспортных и прикладных протоколов семейства TCP/IP, структуры сетевых приложений, основных приемов программирования Internet-приложений на основе этих протоколов с использованием программных интерфейсов сокетов BSD UNIX и Windows Sockets 2.

Задание

Разработать две программы: клиент и сервер, моделирующие работу отделения банка. Сервер должен предоставлять клиенту возможность открытия и закрытия счета, перечисления и снятия денег со счета, перевода некоторых сумм на другие счета и выполнения других подобных операций.

Engine.hpp

```
#ifndef ENGINE_HPP
#define ENGINE_HPP
#pragma comment(lib, "WS2_32.lib")
#pragma comment(linker, "/STACK:36777216")

#include <iostream>
#include <string>
#include <cstring>
#include <WinSock2.h>
using namespace std;

#define die(s) { echo(s); return; }
#define dief(s) { echo(s); return false;}

/**
@brief Universal class for working with sockets
*/
class engine_t
{
    string type;

    WSADATA wsaData;
    SOCKET mysock, remsock;
    sockaddr_in sai;
    char buf[2000000];
public:
    /**
    @brief Shows message
    @detailed We can overload this function for another way of log-messaging
    @param s - Message
    */
    void echo(const string &s) { cout << s << endl; }
```

```
/**
@brief Initialisation
@param mtype - Application type. It can be: "client" or "server"
@param ip - ip-address
@param port - port
*/
engine_t(const string &mtype, const string &ip, int port)
{
    type = mtype;

    // Windows sockets initialisation
    if (WSAStartup(MAKEWORD(2, 0), &wsaData))
        die("Can't startup Windows Sockets");
        echo("Windows Sockets started");

    // Creates a socket that is bound to a specific transport service provider
    if ((mysock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == INVALID_SOCKET)
        die("Can't create socket");
        echo("Socket Created");

    memset(&sai, 0, sizeof(sockaddr_in));
    sai.sin_family = AF_INET;
    sai.sin_port = htons(port);
    sai.sin_addr.s_addr = type == "server" ? INADDR_ANY : inet_addr(ip.c_str());

    if (type == "server")
    {
        // Associates a local address with a socket
        if (bind(mysock, (sockaddr*)(&sai), sizeof(sai)) == SOCKET_ERROR)
            die("Bind error");
            echo("Bind OK!");

        // Places a socket in a state in which it is listening for an incoming connection
        if (listen(mysock, 1) == SOCKET_ERROR)
            die("Listen error");
            echo("Listen OK!");
    }
}

/**
@brief Connects to client/server for chatting
*/
bool connect()
{
    if (type == "client")
    {
        echo("Connecting...");
        if (::connect(mysock, (sockaddr*)(&sai), sizeof(sai)) == SOCKET_ERROR)
            dief("Connect error!");
            echo("Connection complete!");
    }
    else
    {
        echo("Accepting...");
        if ((remsock = accept(mysock, NULL, NULL)) == INVALID_SOCKET)
            dief("Accept error!");
            echo("Accepted!");
    }
    return true;
}
```

```
    /**
    @brief Sends message
    @param s - message
    */
    bool write(const string &s)
    {
        int len = s.size();
        SOCKET to = type == "server" ? remsock : mysock;
        int f1 = send(to, (char*)(&len), sizeof(len), NULL);
        strcpy(buf, s.c_str());
        int f2 = send(to, buf, len + 1, NULL);
        return f1 == sizeof(int) && f2 == len + 1;
    }

    /**
    @brief Gets message
    @param s - message
    */
    bool read(string &s)
    {
        int len = 0;
        SOCKET from = type == "server" ? remsock : mysock;
        int f1 = recv(from, (char*)(&len), sizeof(len), NULL);
        int f2 = recv(from, buf, len + 1, NULL);
        s = string(buf);
        return f1 == sizeof(int) && f2 == len + 1;
    }

    /**
    @brief Destructor
    @detailed Closes sockets
    */
    ~engine_t()
    {
        closesocket(mysock);
        WSACleanup();
    }
};

#endif
```

nsv_client.cpp

```cpp
#include <iostream>
#include "../common/engine.hpp"
using namespace std;

void hint()
{
    puts("\n=== OPERATIONS ==========================================================");
    puts("info id            -- gets information about account id");
    puts("open               -- opens new account and gets new id for user");
    puts("close id           -- tries to close account with id");
    puts("add id amount      -- deposits money to account id");
    puts("get id amount      -- tries to take money from account id");
    puts("mov src dst amount -- tries to move money from account src to account dst");
    puts("=========================================================================\n");
}

int main()
{
    puts("CLIENT-BANK!");
    engine_t engine("client", "127.0.0.1", 5001);
    engine.connect();

    hint();

    while (true)
    {
        printf("> ");

        string query, ans;
        getline(cin, query);

        engine.write(query);
        engine.read(ans);

        puts("\n======= RESULT ====================================");
        puts(ans.c_str());
        puts("===================================================\n");
    }

    cout << "GOOD BYE!" << endl;
    return 0;
}
```

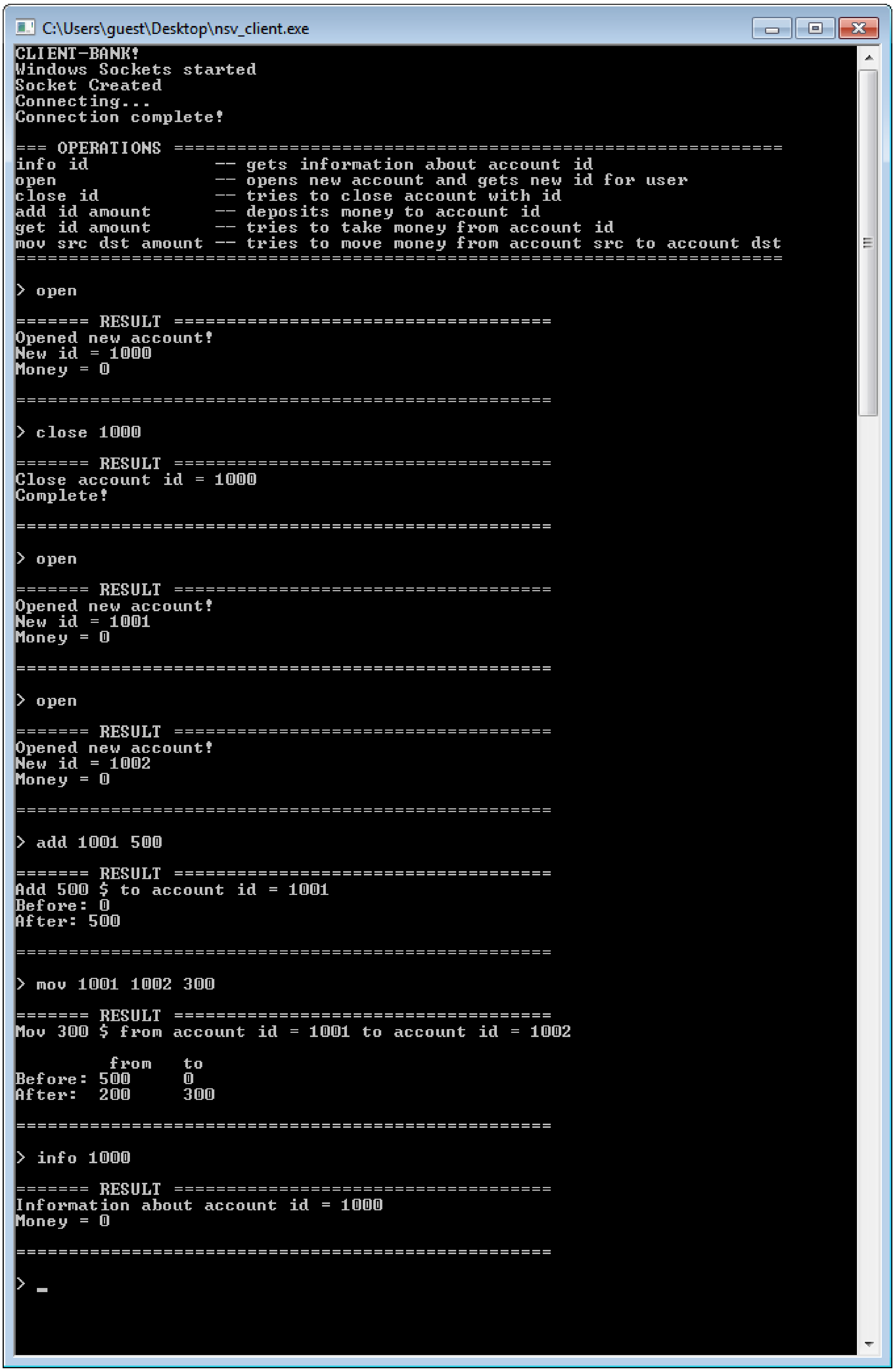
nsv_server.cpp

```cpp
#include "../common/engine.hpp"
#include <unordered_map>
#include <sstream>
#include <vector>
using namespace std;

/**
@brief Bank emulating class
*/
class bank_t
{
    // data <account, money>
    unordered_map<int, int> data;

    int new_id;

    engine_t *engine;

public:
    /**
    @brief Creates new bank
    */
    bank_t()
    {
        engine = new engine_t("server", "127.0.0.1", 5001);
        engine->connect();
        new_id = 1000;
    }

    /**
    @brief Starts main process
    */
    void start()
    {
        while (true)
        {
            string s;
            if (engine->read(s))
                process(s);
            else
            {
                cout << "Connection closed!" << endl;
                engine->connect();
            }
        }
    }

    /**
    @brief Switch how to process request req
    @param req - clients request
    */
    void process(const string &req)
    {
        istringstream is(req);
        string type; is >> type;

        if (type == "open")
        {
            open();
        }
        else if (type == "info")
        {
            int id;
            is >> id;
            info(id);
        }
        else if (type == "close")
        {
            int id;
            is >> id;
            close(id);
        }
        else if (type == "add")
        {
            int id, cnt;
            is >> id >> cnt;
            add(id, cnt);
        }
        else if (type == "get")
        {
            int id, cnt;
            is >> id >> cnt;
            get(id, cnt);
        }
        else if (type == "mov")
        {
            int from, to, cnt;
            is >> from >> to >> cnt;
            mov(from, to, cnt);
        }
        else
            engine->write("Invalid request!");
    }

    /**
    @brief Opens new account and gets new id for user
    */
    void open()
    {
        int id = new_id++;
        data[id] = 0;
        ostringstream os;
        os << "Opened new account!" << endl;
        os << "New id = " << id << endl;
        os << "Money = " << 0 << endl;
        engine->write(os.str());
    }

    /**
    @brief Gets main information about account id
    */
    void info(int id)
    {
        ostringstream os;
        os << "Information about account id = " << id << endl;
        if (data.count(id))
            os << "Money = " << data[id] << endl;
        else
            os << "Account is not opened!" << endl;

        engine->write(os.str());
    }
```

```
/**
@brief Tries to close account with id
@param id - account id
*/
void close(int id)
{
    ostringstream os;
    os << "Close account id = " << id << endl;
    if (data.count(id))
        os << "Complete!" << endl;
    else
        os << "Account is not exist!" << endl;

    engine->write(os.str());
}

/**
@brief Deposits money to account id
@param id - account id
@param cnt - amount of money
*/
void add(int id, int cnt)
{
    ostringstream os;
    os << "Add " << cnt << " $ to account id = " << id << endl;
    if (data.count(id))
    {
        os << "Before: " << data[id] << endl;
        data[id] += cnt;
        os << "After: "  << data[id] << endl;
    }
    else
        os << "Account is not exist!" << endl;

    engine->write(os.str());
}

/**
@brief Tries to take money from account id
@param id - account id
@param cnt - amount of money
*/
void get(int id, int cnt)
{
    ostringstream os;
    os << "Get " << cnt << " $ from account id = " << id << endl;
    if (data.count(id))
    {
        if (data[id] >= cnt)
        {
            os << "Before: " << data[id] << endl;
            data[id] -= cnt;
            os << "After: "  << data[id] << endl;
        }
        else
            os << "Insufficient funds!" << endl;
    }
    else
        os << "Account is not exist!" << endl;

    engine->write(os.str());
}
```

```
    /**
    @brief Tries to move money from account src to account dst
    @param from - source account
    @param to - destination account
    @param cnt - amount of money
    */
    void mov(int from, int to, int cnt)
    {
        ostringstream os;
        os << "Mov " << cnt << " $ from account id = " << from
           << " to account id = " << to << endl;
        if (!data.count(from))
            os << "Account id = " << from << " is not exist!" << endl;
        else if (!data.count(to))
            os << "Account id = " << to   << " is not exist!" << endl;
        else
        {
            if (data[from] >= cnt)
            {
                os << endl;
                os << "     \t from\tto" << endl;
                os << "Before:\t" << data[from] << "\t" << data[to] << endl;
                data[from] -= cnt;
                data[to]   += cnt;
                os << "After:\t"  << data[from] << "\t" << data[to]  << endl;
            }
            else
                os << "Insufficient funds!" << endl;
        }

        engine->write(os.str());
    }
};

int main()
{
    cout << "SERVER-BANK-TERMINAL" << endl;

    bank_t bank;
    bank.start();

    cerr << "Good bye!" << endl;
    return 0;
}
```

Результат работы программы


C:\Users\guest\Desktop\nsv_client.exe

```
CLIENT-BANK!
Windows Sockets started
Socket Created
Connecting...
Connection complete!

=== OPERATIONS =========================================================
info id             -- gets information about account id
open                -- opens new account and gets new id for user
close id            -- tries to close account with id
add id amount       -- deposits money to account id
get id amount       -- tries to take money from account id
mov src dst amount -- tries to move money from account src to account dst
========================================================================

> open

======= RESULT ====================================
Opened new account!
New id = 1000
Money = 0

===================================================

> close 1000

======= RESULT ====================================
Close account id = 1000
Complete!

===================================================

> open

======= RESULT ====================================
Opened new account!
New id = 1001
Money = 0

===================================================

> open

======= RESULT ====================================
Opened new account!
New id = 1002
Money = 0

===================================================

> add 1001 500

======= RESULT ====================================
Add 500 $ to account id = 1001
Before: 0
After: 500

===================================================

> mov 1001 1002 300

======= RESULT ====================================
Mov 300 $ from account id = 1001 to account id = 1002

         from    to
Before: 500      0
After:  200      300

===================================================

> info 1000

======= RESULT ====================================
Information about account id = 1000
Money = 0

===================================================

> _
```