МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**Отчёт по лабораторной работе №4**

**«Сетевое клиент-серверное приложение на основе UDP-сокета»**

Работу выполнил

студент 22 группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.Е. Лучин

Направление подготовки 01.03.02 Прикладная математика и информатика

Проверил

канд. техн. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Полупанов

Краснодар 2022

**Цель работы:** реализовать сетевое клиент-серверное приложение на основе UDP-сокета, позволяющее передавать данные произвольных типов (структуру).

**Задача:** приложение состоит из двух программ: клиент и сервер, взаимодействие которых осуществляется с помощью UDP-сокета. Алгоритм взаимодействия следующий: клиент посылает данные через сокет, сервер в бесконечном цикле слушает свой входной порт и при появлении новых сообщений обрабатывает их, а затем посылает клиенту ответное сообщение с результатами обработки.

**Практическая часть:**

Листинг клиента:

#include <iostream>

// для работы с сокетами

#include <winsock2.h>

// для работы с Windows API

#include <windows.h>

// для доп. функций (напр-р, inet\_pton)

#include <WS2tcpip.h>

/\*

линкер скомпонует библиотеку в исполняемый файл

.exe =

объектные файлы (smth.cpp -> smth.o)

+

библиотеки

\*/

#pragma comment (lib, "Ws2\_32.lib")

using namespace std;

#define SRV\_PORT 1234

/\*

объявление структуры с коэффициентами

для квадратного трехчлена

\*/

struct Coefficients

{

    int a, b, c;

} coeffs;

/\*

объявление структуры с ответом

от сервера

\*/

struct Equation {

    int roots;

    double r1, r2;

} eq;

int main()

{

    /\*

    инициализация Winsock:

    0x0202 - шестнадцатеричный номер

версии интерфейса 2.2

    можно аналогично написать MAKEWORD(2, 2),

    что мы и сделали

    второй аргумент - структура WSADATA wsData

    в нее будут записаны сведения

    о конкретной реализации интерфейса Winsock

    \*/

    WSADATA wsData;

    // код результата. не ноль в случае ошибки

    int resCode = WSAStartup(MAKEWORD(2, 2), &wsData);

    if (resCode != 0)

    {

        // выведем ошибку

        cout << "Error WinSock initializaion #"

<< WSAGetLastError();

        return -1;

    }

    else

        cout << "WinSock initialization OK" << endl;

    /\*

    создание сокета для протокола IPv4 (AF\_INET)

    с режимом дейтаграммного соединения (SOCK\_DGRAM)

    IPPROTO\_UDP - выбор протокола UDP

для нашего соединения

    \*/

    SOCKET clntSock = socket(AF\_INET, SOCK\_DGRAM,

IPPROTO\_UDP);

    if (clntSock == INVALID\_SOCKET)

    {

        cout << "Error initialization socket #"

<< WSAGetLastError() << endl;

        closesocket(clntSock);

        // освободить выделенную память для Winsock

        WSACleanup();

        return -1;

    }

    else

        cout << "Client socket initialization OK"

<< endl << endl;

    // структура с информацией о сервере

    sockaddr\_in srvInfo;

    // опустошение памяти (на всякий случай)

    ZeroMemory(&srvInfo, sizeof(srvInfo));

    /\*

    преобразует локальный IP "127.0.0.1"

версии IPv4 (AF\_INET) в особый численный формат

    и записывает в поле srvInfo sin\_addr

    \*/

    resCode = inet\_pton(AF\_INET, "10.16.4.26",

&srvInfo.sin\_addr);

    if (resCode <= 0)

    {

        cout << "Error in IP translation" << endl;

        return -1;

    }

    // указываем версию IP IPv4

    srvInfo.sin\_family = AF\_INET;

    // htons() преобразует порт в сетевой формат

    srvInfo.sin\_port = htons(SRV\_PORT);

    // размер отправляемого и получаемого пакета

    short packet\_size = 0;

    while (true)

    {

        cout << "Input coeffs: a b c "

<< "(or enter all zeros to exit)"

<< endl;

        cin >> coeffs.a >> coeffs.b >> coeffs.c;

        /\*

прерывание работы в случае

нулевых коэффициентов

\*/

        if (!coeffs.a && !coeffs.b && !coeffs.c)

        {

            cout << "exit" << endl;

            closesocket(clntSock);

            // освободить выделенную память для Winsock

            WSACleanup();

            return 0;

        }

        /\*

        передаем сокет,

        преобразовываем указатель на структуру

        в указатель на char (первая ячейка памяти,

занимаемая структурой)

        передаем размер памяти структуры

        0 - стандартный флаг

        передаем указатель на структуру

        srvInfo (инф-я о том, куда мы отправляем)

        и ее размер

        \*/

        packet\_size = sendto(clntSock, (char\*) &coeffs,

sizeof(coeffs), 0,

(sockaddr\*) &srvInfo,

sizeof(srvInfo));

        if (packet\_size == SOCKET\_ERROR)

        {

            cout << "Can't send message to server. "

<< "Error #" << WSAGetLastError()

<< endl;

            closesocket(clntSock);

            // освободить выделенную память для Winsock

            WSACleanup();

            return -1;

        }

        /\*

        информация о том, от кого получаем ответ

        в данной задаче совпадает с тем, кому

        отправляем

        \*/

        sockaddr newInfo;

        int newInfoSize = sizeof(newInfo);

        // аналогично sendto, но получаем

        packet\_size = recvfrom(clntSock, (char\*) &eq,

sizeof(eq), 0,

(sockaddr\*) &newInfo, &newInfoSize);

        if (packet\_size == SOCKET\_ERROR)

        {

            cout << "Can't recieve message from Server."

<< "Error #" << WSAGetLastError()

<< endl;

            closesocket(clntSock);

            // освободить выделенную память для Winsock

            WSACleanup();

            return -1;

        }

        // интерпретация ответа от сервера

        if (eq.roots == 0)

            cout << "There're no roots in real nums"

<< endl;

        else if (eq.roots == 1)

        {

            cout << "1 root: " << eq.r1 << endl;

        }

        else

        {

            cout << "2 roots: " << eq.r1 << " " << eq.r2

<< endl;

        }

        cout << endl;

    }

    return 0;

}

Листинг сервера:

#include <iostream>

#include <winsock2.h>

#include <windows.h>

#include <WS2tcpip.h>

#include <cmath>

#pragma comment (lib, "Ws2\_32.lib")

using namespace std;

#define SRV\_PORT 1234

struct Coefficients

{

    int a, b, c;

} coeffs;

struct Equation {

    int roots;

    double r1, r2;

} eq;

int main() {

    WSADATA wsData;

    int resCode = WSAStartup(MAKEWORD(2, 2), &wsData);

    if (resCode != 0)

    {

        cout << "Error WinSock initializaion #"

<< WSAGetLastError();

        return -1;

    }

    else

        cout << "WinSock initialization OK" << endl;

    SOCKET servSock = socket(AF\_INET, SOCK\_DGRAM, 0);

    if (servSock == INVALID\_SOCKET)

    {

        cout << "Error initialization socket #"

<< WSAGetLastError() << endl;

        closesocket(servSock);

        WSACleanup();

        return -1;

    }

    else

        cout << "Server socket initialization OK"

<< endl;

    // информация о сервере

    sockaddr\_in servInfo;

    ZeroMemory(&servInfo, sizeof(servInfo));

    servInfo.sin\_family = AF\_INET;

    /\*

готовы принимать сообщения от

любого адреса (INADDR\_ANY)

\*/

    servInfo.sin\_addr.s\_addr = INADDR\_ANY;

    servInfo.sin\_port = htons(SRV\_PORT);

    // привязываем заполненную структуру к сокету

    resCode = bind(servSock, (sockaddr\*) &servInfo,

sizeof(servInfo));

    if (resCode != 0)

    {

        cout << "Error Socket binding to server. "

<< "Error #" << WSAGetLastError() << endl;

        closesocket(servSock);

        WSACleanup();

        return -1;

    }

    else

        cout << "Binding socket to Server info OK"

<< endl << endl;

    short packet\_size = 0;

    while (true)

    {

        // информация о клиенте

        sockaddr\_in clientInfo;

        int clientInfo\_size = sizeof(clientInfo);

        /\*

получаем информацию о клиенте (clientInfo)

и его сообщение

\*/

        packet\_size = recvfrom(servSock,

(char\*) &coeffs,

sizeof(coeffs), 0,

(sockaddr \*) &clientInfo,

&clientInfo\_size);

        if (packet\_size == SOCKET\_ERROR)

        {

            cout << "Can't send message. Error #"

<< WSAGetLastError() << endl;

            closesocket(servSock);

            WSACleanup();

            return -1;

        }

        /\*

        INET\_ADDSTRLEN = длина занимаемого адреса IPv4

        запишем в него строковое

        обозначение

(PSTR - вариация строкового литерала)

        адреса клиента (clientInfo.sin\_addr)

        \*/

        char ipstr[INET\_ADDRSTRLEN];

        inet\_ntop(AF\_INET, &clientInfo.sin\_addr,

(PSTR) ipstr, sizeof(ipstr));

        /\*

преобразовываем порт клиента

(он в сетевом формате) с помощью ntohs()

\*/

        cout << "Received from " << ipstr << ":"

<< ntohs(clientInfo.sin\_port) << endl;

        cout << "Coeffs: " << coeffs.a << ", "

<< coeffs.b << ", " << coeffs.c << endl;

        // считаем дискриминант

        double D = coeffs.b \* coeffs.b –

4 \* coeffs.a \* coeffs.c;

        // просчет вещ-ых корней (если они есть)

        if (D > 0)

        {

            eq.roots = 2;

            eq.r1 = (-coeffs.b + sqrt(D))

/ (2 \* coeffs.a);

            eq.r2 = (-coeffs.b - sqrt(D))

/ (2 \* coeffs.a);

            cout << "Roots: " << eq.r1 << ", "

<< eq.r2 << endl << endl;

        }

        else if (D == 0)

        {

            eq.roots = 1;

            eq.r1 = -coeffs.b / (2 \* coeffs.a);

            cout << "Root: " << eq.r1 << endl << endl;

        }

        else

        {

            eq.roots = 0;

            cout << "No roots" << endl << endl;

        }

        /\*

отправляем тому же клиенту,

от которого получили

\*/

        packet\_size = sendto(servSock, (char\*) &eq,

sizeof(eq), 0,

(sockaddr\*) &clientInfo,

sizeof(clientInfo));

        if (packet\_size == SOCKET\_ERROR)

        {

            cout << "Can't send message. Error #"

<< WSAGetLastError() << endl;

            closesocket(servSock);

            WSACleanup();

            return -1;

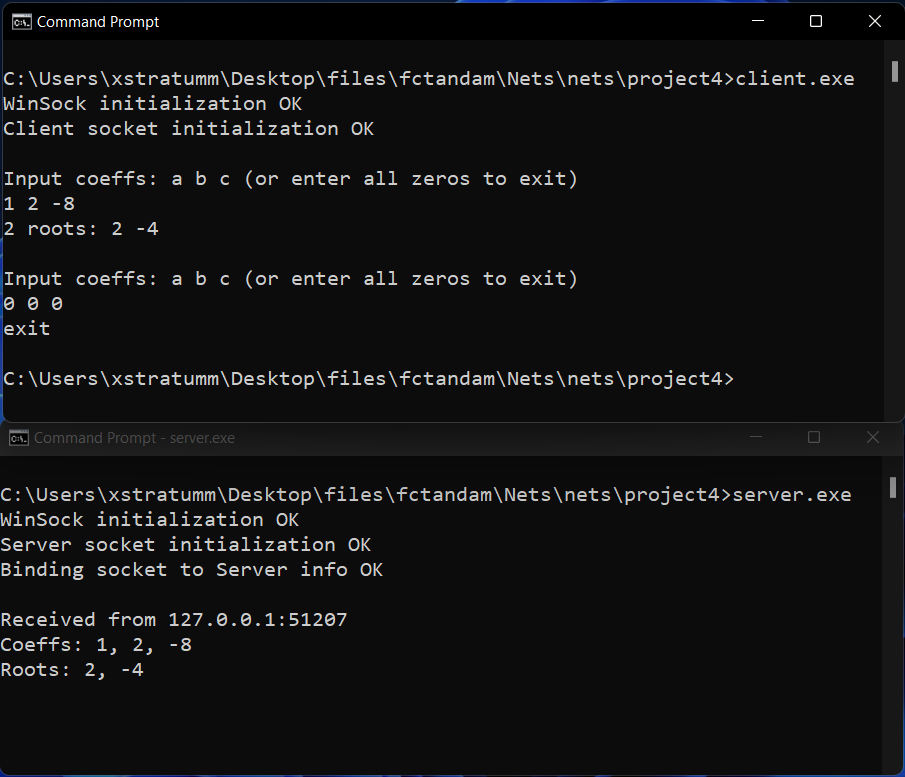
        }

    }

    return 0;

}

Пример работы программ:



*Рисунок 1 – пример работы клиента с сервером*

**Вывод:**былоразработано клиент-серверное приложение на основе UDP-сокетов.