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 №2**

**«Клиент-серверное приложение с общей средой – текстовым файлом»**

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:** разработать клиент-серверное приложение с общей средой – текстовым файлом.

**Задача:** приложение состоит из двух программ: client.exe и server.exe. При взаимодействии этих программ используются 2 файла. Файл REQUEST.txt заполняет клиент, он пишет в него свой запрос. Файл ANSWER.txt заполняет сервер, он пишет в него свой ответ на запрос клиента.

Взаимодействие клиента и сервера происходит по следующему сценарию:

Клиент дописывает в конец файла REQUEST.txt свой запрос. Сервер в бесконечном цикле проверяет, появились ли в файле REQUEST.txt новые запросы. Для этого он сравнивает предыдущий размер файла с текущим (в начале работы предыдущий и текущий размеры файлов равны 0).

Если эти размеры совпали – нет новых запросов. Иначе – появился новый запрос, в этом случае сервер считывает его из файла REQUEST.txt, обрабатывает и результат записывает в конец файла ANSWER.txt.

Клиент аналогично постоянно проверяет файл ANSWER.txt на наличие новых ответов от сервера.

**Практическая часть:**

Листинг клиента:

#include <iostream>

#include <fstream>

#include <windows.h> // для Sleep()

using namespace std;

int roots, sol1, sol2, a, b, c;

long size\_prev;

int main()

{

    // входной и выходной файловые потоки

    ofstream fR;

    ifstream fA;

    // указатели на строковые литералы путей

    const char\* nameR = "REQUEST.txt";

    const char\* nameA = "ANSWER.txt";

    while (true)

    {

        cout << "Enter coeffs: a b c" << endl;

        cin >> a >> b >> c;

        // открываем в режиме записи в конец

        fR.open(nameR, ios::app);

        /\* записываем данные в файл \*/

        fR << a << " " << b << " " << c << endl;

        fR.close();

        /\* открываем файл, где ожидаем увидеть

ответ от сервера \*/

        fA.open(nameA);

        // указатель в конец

        fA.seekg(0, ios::end);

        // запомнили размер файла

        size\_prev = fA.tellg();

        // ждем увеличения размера файла

        while (size\_prev >= fA.tellg())

        {

            Sleep(100);

            // указатель в конец файла

            fA.seekg(0, ios::end);

        }

        /\* ставим указатель на начало

данных ответа сервера \*/

        fA.seekg(size\_prev, ios::beg);

        // считываем число корней

        fA >> roots;

        // интерпретация ответа от сервера

        if (!roots)

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

<< endl << endl;

        else if (roots == 1)

        {

            fA >> sol1;

            cout << "1 root: " << sol1 << endl << endl;

        }

        else

        {

            fA >> sol1 >> sol2;

            cout << "2 roots: " << sol1 << " "

<< sol2 << endl << endl;

        }

        fA.close();

    }

    return 0;

}

Листинг сервера:

#include <iostream>

#include <fstream>

#include <windows.h> // для Sleep()

#include <cmath>

using namespace std;

int a, b, c;

long size\_prev;

int main()

{

    // входной и выходной файловые потоки

    ifstream fR;

    ofstream fA;

    // указатели на строковые литералы (путей)

    const char\* nameR = "REQUEST.txt";

    const char\* nameA = "ANSWER.txt";

    cout << "server is working\n";

    // открытие файла для чтения

    fR.open(nameR);

    // установка указателя в конец файла (0 - смещение)

    fR.seekg(0, ios::end);

    /\* предыдущий размер файла

(инициализируется текущим) \*/

    size\_prev = fR.tellg();

    fR.close();

    while (true)

    {

        // снова открываем и ставим указатель в конец

        fR.open(nameR);

        fR.seekg(0, ios::end);

        /\* пока в файле не появился запрос

        (размер файла такой же или уменьшился) \*/

        while (size\_prev >= fR.tellg())

        {

            Sleep(100);

            // указатель в конец файла

            fR.seekg(0, ios::end);

        } // вышли из цикла => появился запрос

/\* сместим указатель на место перед новой записью, т.е. после 'size\_prev' байт

от начала файла \*/

        fR.seekg(size\_prev, ios::beg);

        fR >> a >> b >> c;

        /\* '>>' добавляет перенос строки, притом

        не переводит указатель на нее, сделаем это сами

        ios::cur - относительно текущего места

        \*/

        fR.seekg(1, ios::cur);

        /\* обновим size\_prev -

        поставим на конец записанных данных \*/

        size\_prev = fR.tellg();

        fR.close();

        cout << a << " " << b << " " << c << endl;

        float D = b \* b - 4 \* a \* c;

        // открыли файл в режиме добавления в конец

        fA.open(nameA, ios::app);

        // просчет корней и запись ответа

        if (D > 0)

            fA << 2 << " " << (-b + sqrt(D)) / (2 \* a)

               << " " << (-b - sqrt(D)) / (2 \* a)

<< endl;

        if (D == 0)

            fA << 1 << " " << -b / (2 \* a) << endl;

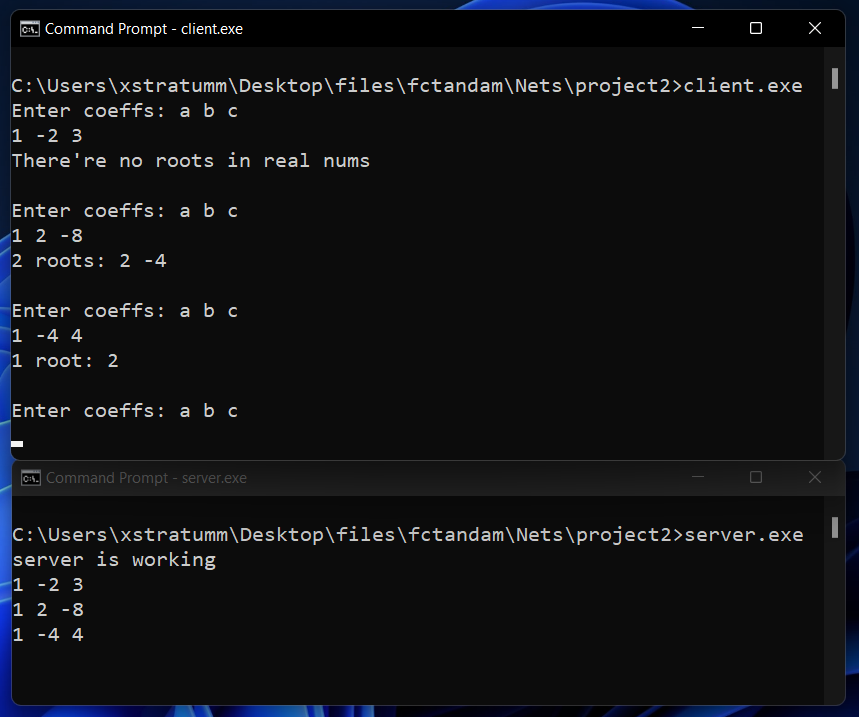
        else fA << 0 << endl;

        fA.close();

    }

}

Пример работы программ:



*Рисунок 1 – пример работы клиента с сервером*

**Вывод:**былоразработано клиент-серверное приложение с общей средой – текстовым файлом.