МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Программирование сетевых приложений»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

азработка клиент­серверных приложений на базе стека протоколов TCP/IP в .Net.

Выполнил: студент гр. ИТП-41

Болтникова Е.В.

Принял: ассистент

Гуменников Е.Д.

Гомель 2022

**Цель работы:** понять принципы разработки сетевых приложений, разработанных средствами .*NET*.

**Задание:**

Разработать сервер с использованием пула потоков для параллельной обработки запросов клиентов. В качестве задачи необходимо выполнить одномерную оптимизация функции методом Фибоначчи.

**Ход выполнения**

В процессе выполнения данной лабораторной работы было разработано два приложения: серверное и клиентское. Клиентское приложение осуществляет чтение файла и отправляет прочитанное на сервер. После чего сервер возвращает обработанный текст, который впоследствии отображается на экране клиенту. На рисунке 1 представлена работа клиентского приложения.

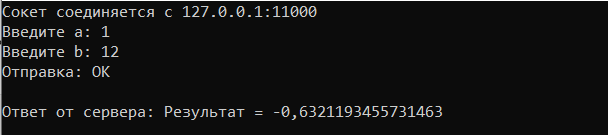


Рисунок 1 – Результат работы клиентского приложения

Серверное приложение в свою очередь осуществляет прослушивание подключений клиентов. Как только клиент подключается, сервер получает от него текст, который впоследствии обрабатывается регулярным выражением для получения из него email адресов. Полученные адреса отправляются обратно клиенту, который их впоследствии отображает. На рисунке 2 представлен результат работы серверного приложения.

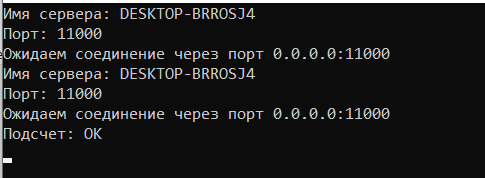


Рисунок 2 – Результат работы серверного приложения

Листинг программы приведён в приложении А.

**Вывод**: в процессе выполнения данной лабораторной работы были изучены основные принципы разработки клиент серверных приложений с использованием пула потоков для *.NET.* А также полученные знания были применены на практике для разработки распределенного приложения под систему *Windows*.

**Приложение А**

**Листинг серверного приложения:**

using System;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.Threading;

namespace Server

{

class Program

{

const double epsilon = 1e-3;

static int F(int n)

{

int f, f1 = 1, f2 = 1, m = 0;

while (m < n - 1)

{

f = f1 + f2;

f1 = f2;

f2 = f;

++m;

}

return f1;

}

static double Fun(double x)

{

return Math.Pow(Math.E, -1) - Math.Cos(x);

}

static double Fib(double a, double b)

{

double x1, x2, \_x, xf1, xf2;

int k = 0;

int N = 0;

double fn1 = 1, fn2 = 1, fn, f = (b - a) / epsilon;

while (fn1 < f)

{

fn = fn1 + fn2;

fn1 = fn2;

fn2 = fn;

++N;

}

bool bix;

int ix = N & 1;

if (ix == 1)

bix = true;

else

bix = false;

x1 = a + (double)F(N - 2) / F(N) \* (b - a) - (bix ? -1 : 1) \* epsilon / F(N);

x2 = a + (double)F(N - 1) / F(N) \* (b - a) + (bix ? -1 : 1) \* epsilon / F(N);

xf1 = Fun(x1);

xf2 = Fun(x2);

P:

++k;

if (xf1 >= xf2)

{

ix = (N - k) & 1;

if (ix == 1)

bix = true;

else

bix = false;

a = x1;

x1 = x2;

xf1 = xf2;

x2 = a + (double)F(N - k - 1) / F(N - k) \* (b - a) + (bix ? -1 : 1) \* epsilon / F(N - k);

xf2 = Fun(x2);

}

else

{

ix = (N - k) & 1;

if (ix == 1)

bix = true;

else

bix = false;

b = x2;

x2 = x1;

xf2 = xf1;

x1 = a + (double)F(N - k - 2) / F(N - k) \* (b - a) - (bix ? -1 : 1) \* epsilon / F(N - k);

xf1 = Fun(x1);

}

if (Math.Abs(b - a) <= epsilon)

{

\_x = (a + b) / 2;

double result = Fun(\_x);

return result;

}

else

goto P;

}

static void ThreadProc(Object obj)

{

Socket handler = (Socket)obj;

string data = null;

// Мы дождались клиента, пытающегося с нами соединиться

byte[] bytes = new byte[sizeof(double) \* 2];

double[] numbers = new double[2];

int bytesRec = handler.Receive(bytes);

data += Encoding.UTF8.GetString(bytes, 0, bytesRec);

Buffer.BlockCopy(bytes, 0, numbers, 0, bytes.Length);

double a = Convert.ToDouble(numbers[0]);

double b = Convert.ToDouble(numbers[1]);

Console.Write("Подсчет: ");

double result = Fib(a, b);

Console.WriteLine("ОК");

// Отправляем ответ клиенту

string reply = "Результат = " + result;

byte[] msg = Encoding.UTF8.GetBytes(reply);

handler.Send(msg);

handler.Shutdown(SocketShutdown.Both);

handler.Close();

}

static void Main(string[] args)

{

// Устанавливаем для сокета локальную конечную точку

string host = Dns.GetHostName();

int port = 11000;

IPEndPoint ipEndPoint = new IPEndPoint(IPAddress.Any, port);

// Создаем сокет Tcp/Ip

Socket sListener = new Socket(ipEndPoint.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

// Назначаем сокет локальной конечной точке и слушаем входящие сокеты

try

{

sListener.Bind(ipEndPoint);

sListener.Listen(10);

// Начинаем слушать соединения

while (true)

{

Console.WriteLine("Имя сервера: {0}", host);

Console.WriteLine("Порт: {0}", port);

Console.WriteLine("Ожидаем соединение через порт {0}", ipEndPoint);

// Программа приостанавливается, ожидая входящее соединение

Socket handler = sListener.Accept();

ThreadPool.QueueUserWorkItem(new WaitCallback(ThreadProc), handler);

}

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

finally

{

Console.ReadLine();

}

}

}

}

**Листинг клиентского приложения:**

using System;

using System.Text;

using System.Net;

using System.Net.Sockets;

namespace Client

{

class Program

{

static void Main(string[] args)

{

try

{

string host = "127.0.0.1";

// Буфер для входящих данных

byte[] bytes = new byte[1024];

// Устанавливаем удаленную точку для сокета

IPEndPoint ipEndPoint = new IPEndPoint(IPAddress.Parse(host), 11000);

// Создание сокета отправителя, который привязывается к удаленной токче с протоколом TCP

Socket client\_socket = new Socket(ipEndPoint.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

// Соединяем сокет с удаленной точкой

client\_socket.Connect(ipEndPoint);

Console.WriteLine("Сокет соединяется с {0} ", client\_socket.RemoteEndPoint.ToString());

Console.Write("Введите a: ");

double a = Convert.ToDouble(Console.ReadLine());

Console.Write("Введите b: ");

double b = Convert.ToDouble(Console.ReadLine());

double[] numbers = new double[2];

numbers[0] = a;

numbers[1] = b;

byte[] msg;

msg = new byte[numbers.Length \* sizeof(double)];

Buffer.BlockCopy(numbers, 0, msg, 0, msg.Length);

// Отправляем данные через сокет

client\_socket.Send(msg);

Console.WriteLine("Отправка: ОК");

// Получаем ответ от сервера

int bytesReceive = client\_socket.Receive(bytes);

Console.WriteLine("\nОтвет от сервера: {0}\n\n", Encoding.UTF8.GetString(bytes, 0, bytesReceive));

// Освобождаем сокет

client\_socket.Shutdown(SocketShutdown.Both);

client\_socket.Close();

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

finally

{

Console.ReadLine();

}

}

}

}