**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

по дисциплине «Программирование сетевых приложений»

на тему: **«Программирование сетевого взаимодействия по**

**протоколам *HTTP*/*FTP*»**

Выполнил: студент гр. ИТП-41

Климанский П.В.

Принял: преподаватель

Гуменников Е.Д.

Гомель 2022

**Цель работы:** изучение сетевого взаимодействия по протоколам *HTTP*/*FTP*.

**Задание:**

Для задания представленном ниже для коммуникации использовать HTTP протокол. ( Вся работа должна быть выполнена с использованием стандартных сокетов, все HTTP заголовки формируются вручную) Разработать http-сервер (использовать пул потоков для параллельной обработки запросов клиентов). В качестве клиентского приложения используется веб-браузер. Все заголовки формируются вручную. Браузер должен правильно отображать получаемую информацию. Рекомендуется на GET запрос возвращать html страницу с формой ввода, и выполнять POSTзапрос, на который сервер присылает решение.

**Вариант 57.** Разработать конечный автомат для анализа корректности номера телефона в формате: [ +XXX (YY) XXX­XX­XX ]. Найти все номера телефонов в файле отправленном пользователем. Клиенту возвращается список всех найденных номеров телефонов в тексте

**Ход выполнения**

В ходе выполнения данной лабораторной работы было разработано сетевое приложение, в котором анализируется введенная в окно браузера строка и выводятся номера телефонов в заданном формате. На рисунке 1 представлена работа сервера. На рисунке 2 и 3 представлен вывод на html странице.

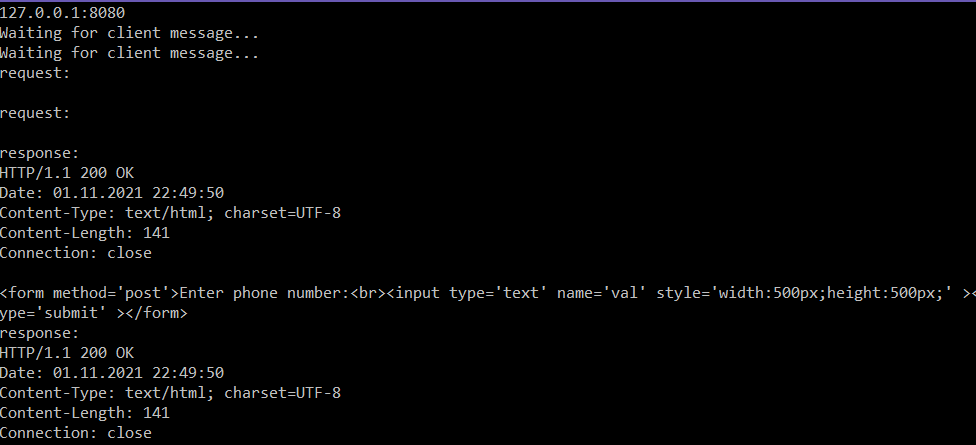


Рисунок 1 – Результат работы сервера

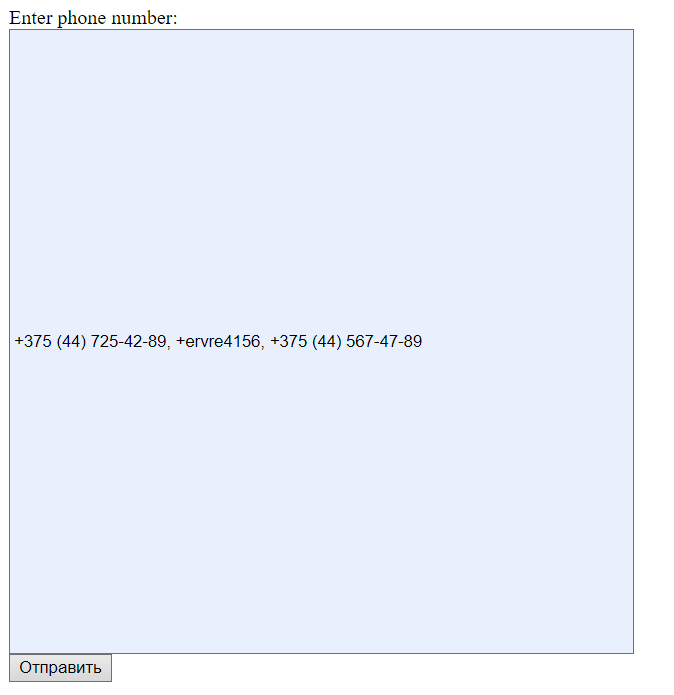


Рисунок 2 – Ввод значений

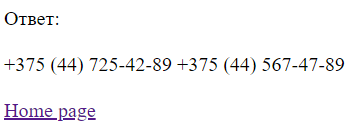


Рисунок 3 – Результат работы программы

**Вывод:** в ходе выполнения лабораторной работы разработано приложение с сетевым взаимодействием по протоколам *HTTP*/*FTP*.

**Листинги программы:**

**Листинг Program.cs:**

using System;

using System.IO;

using System.Linq;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading;

namespace Lab5

{

class Program

{

//const string path = @"e:\University\Course4\PSP\Lab5\Lab5\answer.txt";

//public static void CreateFile(string result)

//{

// string fileName = "answer.txt";

// using (FileStream fstream = new FileStream(path, FileMode.OpenOrCreate))

// {

// // преобразуем строку в байты

// byte[] array = Encoding.Default.GetBytes(result);

// // запись массива байтов в файл

// fstream.Write(array, 0, array.Length);

// }

//}

public static string SuccessHeaders(int contentLength)

{

StringBuilder builder = new StringBuilder();

builder.Append("HTTP/1.1 200 OK").Append("\r\n");

builder.Append("Date: ").Append(DateTime.Now).Append("\r\n");

builder.Append("Content-Type: text/html; charset=UTF-8").Append("\r\n");

builder.Append("Content-Length: ").Append(contentLength).Append("\r\n");

builder.Append("Connection: close").Append("\r\n");

builder.Append("\r\n");

return builder.ToString();

}

private static string AnswerPage(string val)

{

StringBuilder bodyBuilder = new StringBuilder();

bodyBuilder.Append("Ответ:");

bodyBuilder.Append("<br><br>");

bodyBuilder.Append(val);

bodyBuilder.Append("<br>");

bodyBuilder.Append("<br>");

bodyBuilder.Append("<a href='/'> Home page </a> ");

string body = bodyBuilder.ToString();

//warning, if body contains non-ascii symbols next record can be incorect

return string.Concat(SuccessHeaders(body.Length), body);

}

private static string BadAnswerPage()

{

StringBuilder bodyBuilder = new StringBuilder();

bodyBuilder.Append("Something was bad");

bodyBuilder.Append("<br>");

bodyBuilder.Append("<a href='/'> Save result </a> ");

string body = bodyBuilder.ToString();

//warning, if body contains non-ascii symbols next record can be incorect

return string.Concat(SuccessHeaders(body.Length), body);

}

private static string IndexPage()

{

StringBuilder bodyBuilder = new StringBuilder();

bodyBuilder.Append("<form method='post'>");

bodyBuilder.Append("Enter phone number:");

bodyBuilder.Append("<br>");

bodyBuilder.Append("<input type='text' name='val' style='width:500px;height:500px;' >");

bodyBuilder.Append("<br>");

bodyBuilder.Append("<input type='submit' >");

bodyBuilder.Append("</form>");

string body = bodyBuilder.ToString();

//warning, if body contains non-ascii symbols next record can be incorect

return string.Concat(SuccessHeaders(body.Length), body);

}

static void ProcessClient(object obj)

{

TcpClient client = (TcpClient)obj;

NetworkStream clientStream = client.GetStream();

try

{

clientStream.ReadTimeout = 200;

clientStream.WriteTimeout = 200;

Console.WriteLine("Waiting for client message...");

string messageData = ReadMessage(clientStream);

Console.WriteLine("request: ");

Console.WriteLine(messageData);

string page = "";

if (messageData.StartsWith("POST"))

{

Regex r = new Regex("\r\n\r\n");

string[] request = r.Split(messageData, 2);

string phoneNumber = "\\+[0-9]{3} \\([0-9]{2}\\) [0-9]{3}-[0-9]{2}-[0-9]{2}";

//Regex r = new Regex("\r\n\r\n");

string result = "";

if (request.Length < 1)

{

page = BadAnswerPage();

}

else

{

string val = request[1].Split('=')[1];

string result1 = "";

int step = 0;

for(int i = 0; i < val.Length; i+= step)

{

if (val[i] == '%')

{

if(val[i + 1] == '2')

{

if(val[i + 2] == 'B')

{

result1 += "+";

step = 3;

}

if (val[i + 2] == '8')

{

result1 += "(";

step = 3;

}

if (val[i + 2] == '9')

{

result1 += ")";

step = 3;

}

if (val[i + 2] == 'C')

{

result1 += ",";

step = 3;

}

}

}

else if(val[i] == '+')

{

result1 += " ";

step = 1;

}

else

{

result1 += val[i];

step = 1;

}

}

string[] arr;

try

{

arr = result1.Split(',').ToArray();

for (int i = 0; i < arr.Length; i++)

{

if (Regex.IsMatch(arr[i], phoneNumber, RegexOptions.IgnoreCase))

{

result += arr[i] + " ";

}

}

}

catch

{

val = "Validation error";

}

page = AnswerPage(result);

}

}

else

{

page = IndexPage();

}

Console.WriteLine("response:");

Console.WriteLine(page);

byte[] message = Encoding.UTF8.GetBytes(page);

clientStream.Write(message, 0, message.Length);

clientStream.Flush();

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

clientStream.Close();

client.Close();

}

}

static void Main(string[] args)

{

TcpListener listener = new TcpListener(IPAddress.Parse("127.0.0.1"), 8080);

listener.Start();

Console.WriteLine(listener.LocalEndpoint);

while (true)

{

TcpClient client = listener.AcceptTcpClient();

ThreadPool.QueueUserWorkItem(ProcessClient, client);

}

}

static string ReadMessage(NetworkStream clientStream)

{

StringBuilder messageData = new StringBuilder();

try

{

byte[] buffer = new byte[2048];

int bytes = -1;

do

{

//Read the client's test message.

bytes = clientStream.Read(buffer, 0, buffer.Length);

//Use Decoder class to convert from bytes to UTF8 in case a character spans two buffers.

Decoder decoder = Encoding.UTF8.GetDecoder();

char[] chars = new char[decoder.GetCharCount(buffer, 0, bytes)];

decoder.GetChars(buffer, 0, bytes, chars, 0);

messageData.Append(chars);

}

while (bytes != 0);

}

catch (Exception)

{

}

return messageData.ToString();

}

}

}