МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САМАРСКИЙ УНИВЕРСИТЕТ

Отчёт по лабораторной работе №2

«Исследование web-сайта»

по курсу «Сети ЭВМ и телекоммуникации»

Выполнил:
Власов Г.В.
гр.6303

Проверил:
Пигусов А.С.

Самара 2019

Цель работы:

Освоение принципов работы протокола прикладного уровня HTTP, приобретение навыков формирования и обработки запросов по протоколу HTTP.

Задание:

Определить и вывести количество и суммарный размер всех файлов сервера, не являющихся HTML-документами и рисунками. Вывести их список.

Листинг программы:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
using HtmlAgilityPack;
using System.Net.Sockets;
using HtmlDocument = HtmlAgilityPack.HtmlDocument;
using System.Linq;
namespace Laba2_HTTP
{
    public partial class Form1 : Form
    {
        public Form1() {
            InitializeComponent();
            label1.Visible = false;
            label2.Visible = false;
            label6.Visible = false;
            label7.Visible = false;
        }
        private void Form1_Load(object sender, EventArgs e) { }
        private string[] exp = { ".doc",".txt",".docx",".pdf"};
        private static Socket GetSocket(string host, int port)
        {
            Socket s = null;
            var tempSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
            try
            {
                tempSocket.Connect(host, port);
            }
            catch (SocketException ex) { return null; }
            if (tempSocket.Connected) s = tempSocket;
            return s;
        }
        private static string GetPage(string host, int port, string pages)
        {
                string request = pages != "" ? "GET " + pages + " HTTP/1.1\r\n" + "Host:
" + host + "\r\nConnection: Close\r\n\r\n" : "GET / HTTP/1.1\r\nHost: " + host +
"\r\nConnection: Close\r\n\r\n";
                Byte[] bytesSent = Encoding.Default.GetBytes(request);
                Byte[] bytesReceived = new Byte[256];
                Socket s = GetSocket(host, port);
```

```
                if (s == null) return "Connection failed";
                s.Send(bytesSent, bytesSent.Length, 0);
                int bytes;
                string page = "";
                do
                {
                    bytes = s.Receive(bytesReceived, 0, bytesReceived.Length, 0);
                    page = page + Encoding.Default.GetString(bytesReceived, 0, bytes);
                }
                while (bytes > 0);
                return page;
        }
        private static List<Uri> FindLinks(string responseData, Uri linkForResponse)
        {
            HtmlDocument doc = new HtmlDocument();
            doc.LoadHtml(responseData);
            List<Uri> urlList = new List<Uri>();
            if (doc.DocumentNode?.SelectNodes(@"//a[@href]") == null) return urlList;
            foreach (HtmlNode link in doc.DocumentNode.SelectNodes(@"//a[@href]"))
            {
                try
                {
                    HtmlAttribute attribute = link.Attributes["href"];
                    if (attribute == null) continue;
                    string href = attribute.Value;
                    if (href.StartsWith("javascript",
StringComparison.InvariantCultureIgnoreCase)) continue;
                    Uri urlNext = new Uri(href, UriKind.RelativeOrAbsolute);
                    urlNext = new Uri(linkForResponse, urlNext);
                    if (!urlList.Contains(urlNext))
                    {
                        urlList.Add(urlNext);
                    }
                }
                catch (Exception e){}
            }
            return urlList;
        }
        private void button1_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
            listBox2.Items.Clear();
            List<String> allLinks1 = new List<string>();
            List<String> rightLinks1 = new List<string>();
            List<Uri> allLinks2 = new List<Uri>();
            List<Uri> rightLinks2 = new List<Uri>();
            int size = 0;
            int Deep = int.Parse(TextBox_Deep.Text);
            int k = 0;
            int l = -1;
            List<Uri> urlsList = new List<Uri>();
            bool result = Uri.TryCreate(textBox_URI.Text, UriKind.Absolute, out var
uriResult) && uriResult.Scheme == Uri.UriSchemeHttp;
            urlsList.Add(uriResult);
            string buf = "";
            for (int i = 0; i < Deep; i++)
            {
                List<Uri> allLinks = new List<Uri>();
                List<Uri> rightLinks = new List<Uri>();
                k = 0;
                foreach (var uri in urlsList)
                {
                    string page = GetPage(uri.Host, 80, uri.LocalPath);
                    List<Uri> LinksOnPage = FindLinks(page, uri);
```

```
                    foreach (var findedUri in LinksOnPage)
                    {
                        if (findedUri.Host == uriResult.Host &&
!allLinks.Contains(findedUri))
                        {
                            allLinks.Add(findedUri);
                            allLinks2.Add(findedUri);
                            string str = findedUri.ToString();
                            int j = str.Length - 1;
                            while (!str[j].Equals('.'))
                            {
                                j--;
                            }
                            buf = str.Substring(j, str.Length - j);
                            for (int q = 0; q < exp.Length; q++)
                            {
                                if (buf.Equals(exp[q]))
                                {
                                    rightLinks2.Add(findedUri);
                                    rightLinks.Add(findedUri);
                                    size += page.Length;
                                    k++;
                                }
                            }
                        }
                    }
                }
                urlsList = allLinks;
            }
            for (int j = 0; j < allLinks2.Count; j++)
            {
                allLinks1.Add(allLinks2[j].ToString());
            }
            for (int j = 0; j < rightLinks2.Count; j++)
            {
                rightLinks1.Add(rightLinks2[j].ToString());
            }
            allLinks1 = allLinks1.Distinct().ToList();
            rightLinks1 = rightLinks1.Distinct().ToList();
            for (int j = 0; j < allLinks1.Count; j++)
            {
                listBox1.Items.Add((j + 1) + ") " + allLinks1[j]);
            }
            for (int j = 0; j < rightLinks1.Count; j++)
            {
                listBox2.Items.Add((j + 1) + ") " + rightLinks1[j]);
            }
            label1.Visible = true;
            label2.Visible = true;
            label6.Visible = true;
            label7.Visible = true;
            label1.Text = "" + k + " Штук";
            label2.Text = "" + size + " Байт";
        }
        private void Form1_Load_1(object sender, EventArgs e)
        {

        }
    }
}
```

Результат выполнения программы: