МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»

ФАКУЛЬТЕТ ИНФОРМАТИКИ

КАФЕДРА ПРОГРАММНЫХ СИСТЕМ

Отчет к лабораторной работе № 2

«Исследование web-сайта»

по курсу «Сети ЭВМ и телекоммуникации»

Выполнил:

Медов Денис Андреевич

гр. 6303-090301D

Проверил:

Пигусов А.С.

Самара 2021

Введение

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

import requests

from bs4 import BeautifulSoup

DOCTYPE = ['.pdf', '.docx', '.xlsx', '.doc', '.xls']

BAD\_DOCTYPE = ['.xml', '.jpg']

user\_agent = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.54 Safari/537.36'}

def chek\_reference(href):

    return 'https://' in href or 'http://' in href

def bad\_refrence(href, site):

    if not site in href: return True

    for type in BAD\_DOCTYPE:

        if type in href: return True

    if href.replace(site, '') == '': return True

    if href.replace(site, '') == ' ': return True

    if href.replace(site, '') == '#': return True

    return False

def get\_links(html, site):

    soup = BeautifulSoup(html, features="html.parser")

    links = []

    for link in soup.find\_all('a', href=True):

        href = link['href']

        if href == '' or href == ' ' or href == '/' or href == '/ ': continue

        elif chek\_reference(href):

            if bad\_refrence(href, site): continue

        else:

            if href[0] == '/':

                href = site + href

            else:

                href = site + '/' + href

            if bad\_refrence(href, site): continue

        links.append(href)

    return list(set(links))

def check\_file(link):

    for type in DOCTYPE:

        if link.find(type) != -1:

            return True

    return False

def parser(session, respons, site, deep):

    links = get\_links(respons.text, site)

    file, subsite = [], []

    for link in links:

        try:

            print(link)

            if check\_file(link):

                respons = session.head(link, timeout=2)

                if respons.ok:

                    file.append({

                        'file' : link,

                        'size' : respons.headers['Content-Length']})

            elif deep > 0:

                respons = session.get(link, timeout=2)

                if respons.ok:

                    subsite.append(link)

                    sub\_file , sub\_subsite = parser(session, respons, site, deep - 1)

                    for item\_file in sub\_file:

                        if not item\_file in file:

                            file.append(item\_file)

                    for item\_subsite in sub\_subsite:

                        if not item\_subsite in subsite:

                            subsite.append(item\_subsite)

        except:

            pass

    return file, subsite

def parse(site, deep):

    session = requests.Session()

    session.headers.update(user\_agent)

    respons = session.get(site)

    file, subsite = [], []

    if respons.ok:

        file , subsite = parser(session, respons, site, deep)

    return {

        "site" : site,

        "file" : file,

        "subsite" : subsite

    }

Результат выполнения программы

Результат выполнения программы представлен на рисунке 1.

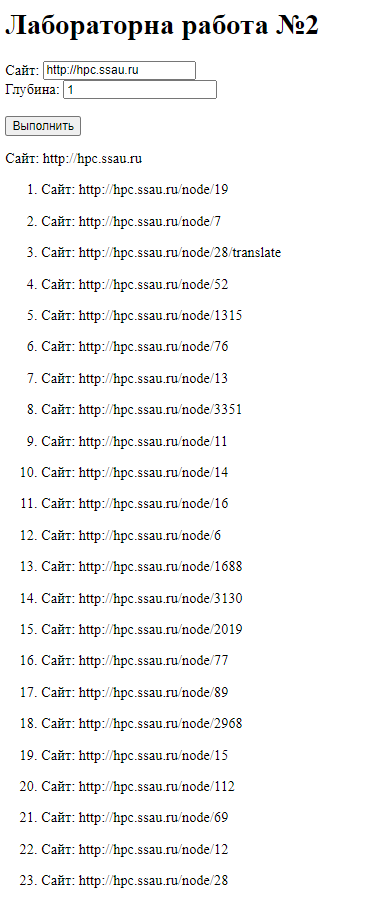
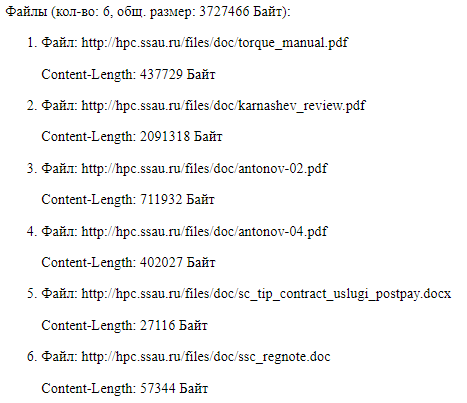
 

Рисунок 1 – Результат выполнения программы