ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет «Инфокоммуникационных технологий»
Направление подготовки «45.03.04 Интеллектуальные системы в гуманитарной сфере»

О Т Ч Е Т

Лабораторная работа №1, Web-программирование

Тема задания: Работа с сокетами

Выполнил:
Студент **Чанова С. Ю.** **К33422**
Фамилия И.О. номер группы

Проверил:
Преподаватель **Говоров А. И.**
Фамилия И.О

Санкт-Петербург
2020

Ход работы

Часть 1

Реализовать клиентскую и серверную часть приложения. Клиент отсылает серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отсылает клиенту сообщение «Hello, client». Сообщение должно отобразиться у клиента.

Серверная часть - task_1_server.py

```python
import socket
import time

# сокет сервера
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('localhost', 14000))
server_socket.listen(10)
server_socket.setblocking(False)

while True:
    try:
        # подключение к сокету клиента
        client_socket, client_address = server_socket.accept()
        print(f'connected to: {client_address}')
        server_socket.setblocking(True)

        # получение сообщения от клиента
        data = client_socket.recv(16384)
        data = data.decode('utf-8')
        print('received data: ' + data)

        # отправка сообщения клиенту
        server_msg = 'hello, client!'
        client_socket.send(bytes(server_msg, 'utf-8'))

        server_socket.close()
        break

    except socket.error:
        time.sleep(5)

    except KeyboardInterrupt:
        server_socket.close()
        break
```

Клиентская часть - task_1_client.py

```python
import socket

# сокет клиента
client_socket = socket.socket(socket.AF_INET,
                              socket.SOCK_STREAM)

client_socket.connect(('localhost', 14000))

# отправка сообщения серверу
client_msg = 'hello, server!'
client_socket.send(bytes(client_msg, 'utf-8'))

# получение сообщения от сервера
data = client_socket.recv(16384)
data = data.decode('utf-8')
print('received data: ' + data)

client_socket.close()
```

Исполнение программ:

```
(base) spiritofsofya@cardamom:~/web_2020/lab_1$ python3 task_1_server.py
connected to: ('127.0.0.1', 46450)
received data: hello, server!
```

```
(base) spiritofsofya@cardamom:~/web_2020/lab_1$ python3 task_1_client.py
received data: hello, client!
```

Часть 2

Реализовать клиентскую и серверную часть приложения. Клиент запрашивает у сервера выполнение математической операции, параметры, которые вводятся с клавиатуры. Сервер обрабатывает полученные данные и возвращает результат клиенту.

Вариант 3: поиск площади трапеции.

Серверная часть - task_2_server.py

```python
import socket
import time
import struct

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = 'localhost'
port = 14042
server_socket.bind((host, port))
server_socket.listen(10)
server_socket.setblocking(False)

while True:
    try:
        client_socket, client_address = server_socket.accept()
        print(f'connected to: {client_address}')
        server_socket.setblocking(True)

        # получение данных об основаниях и высоте трапеции
        data = client_socket.recv(16384)
        data = struct.unpack('ddd', data)
        a = data[0]
        b = data[1]
        h = data[2]

        print('received first base: ' + str(a))
        print('received second base: ' + str(b))
        print('received height: ' + str(h))

        # расчет площади
        area = (a + b) / 2 * h
        area = struct.pack('d', area)

        client_socket.send(area)

        server_socket.close()
        break

    except socket.error:
        print('waiting for clients')
        time.sleep(3)

    except KeyboardInterrupt:
        server_socket.close()
        break
```

Клиентская часть - task_2_client.py

```python
import socket
import struct


def get_positive_float():  # получение данных в корректной форме
    while True:
        try:
            value = float(input())
        except ValueError:
            print("incorrect input. please enter a positive number")
            continue

        if value <= 0:
            print("incorrect input. please enter a positive number")
            continue
        else:
            break
    return value


client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 14003))

# отправка данных о трапеции
print('send values to server to calculate area of trapezium: ')

print('first base: ')
a = get_positive_float()

print('second base: ')
b = get_positive_float()

print('height: ')
h = get_positive_float()

data = struct.pack('ddd', a, b, h)
client_socket.send(data)

# получение расчитанной площади
area = client_socket.recv(16384)
area = struct.unpack('d', area)
area = str(area[0])
print('trapezium area: ' + area)

client_socket.close()
```

Исполнение программ:

```
(base) spiritofsofya@cardamom:~/web_2020/lab_1$ python3 task_2_server.py
connected to: ('127.0.0.1', 35490)
received first base: 10.0
received second base: 6.0
received height: 4.0
```

```
(base) spiritofsofya@cardamom:~/web_2020/lab_1$ python3 task_2_client.py
send values to server to calculate area of trapezium:
first base:
10
second base:
6
height:
4
trapezium area: 32.0
```

Часть 3

Реализовать серверную часть приложения. Клиент подключается к серверу. В ответ клиент получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

Серверная часть - task_3_server.py

```python
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = 'localhost'
port = 14042
server_socket.bind((host, port))


print('starting server on: ', host, port, ' press ctrl+c to exit')

server_socket.listen(1)
client_socket, client_address = server_socket.accept()

while True:
    with open('index.html', 'r') as page:
        response_type = 'HTTP/1.0 200 OK\n'
        headers = 'Connect-Type: text/html\n\n'
        body = ''.join(page)
        response = response_type + headers + body
        client_socket.send(response.encode('utf-8'))
    data = client_socket.recv(1024)
    if not data:
        break

client_socket.close()
```

Клиентская часть - task_3_client.py

```python
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = 'localhost'
port = 14042
client_socket.connect((host, port))

while True:
    data = client_socket.recv(1024)
    if not data:
        break
    print(data.decode('utf-8'))

client_socket.close()
```

```
(base) spiritofsofya@cardamom:~/web_2020/lab_1$ python3 task_3_server.py
starting server on:  localhost 14045  press ctrl+c to exit
```

```
(base) spiritofsofya@cardamom:~/web_2020/lab_1$ python3 task_3_client.py
HTTP/1.0 200 OK
Connect-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>:)</title>
</head>
<body>
<h1>hello, client</h1>
</body>
</html>
```

Часть 4

Реализовать двухпользовательский или многопользовательский чат. Реализация многопользовательского часа позволяет получить максимальное количество баллов.

Реализован многопользовательский чат с потоками.

Серверная часть - task_4_server.py

```python
import socket
import threading

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

host = 'localhost'
port = 14042

server_socket.bind((host, port))
server_socket.listen(10)

clients = []
nicknames = []
print('starting chat server on: ', host, port, ' press ctrl+c to exit')


def broadcast(message):  # рассылка сообщения всем клиентам в чате
    for client in clients:
        client.send(message)


def handle_client(client):  # получить сообщение или выход из чата
    while True:
        try:
            message = client.recv(1024)
            broadcast(message)
        except:
            client_index = clients.index(client)
            clients.remove(client)
            client.close()
            nickname = nicknames[client_index]
            nicknames.remove(nickname)
            broadcast(f'{nickname} has left the chat'.encode('utf-8'))
            break


def receive():  # потоковое получение сообщений
    while True:
        client_socket, client_address = server_socket.accept()
        print(f'connected to: {client_address}')

        client_socket.send('NICKNAME'.encode('utf-8'))
        nickname = client_socket.recv(1024).decode('utf-8')
        nicknames.append(nickname)
        clients.append(client_socket)

        print(f'nickname of the client is {nickname}')
        broadcast(f'{nickname} has entered the chat'.encode('utf-8'))
        client_socket.send('successful connection to the chat'.encode('utf-8'))

        thread = threading.Thread(target=handle_client, args=(client_socket,))
        thread.start()


receive()
```

Клиентская часть - task_4_client.py

```python
import socket
import threading

nickname = input('please enter your nickname: ')

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = 'localhost'
port = 14042
client_socket.connect((host, port))


def receive_messages():  # получение сообщений или выбрать ник
    while True:
        try:
            message = client_socket.recv(1024).decode('utf-8')
            if message == 'NICKNAME':
                client_socket.send(nickname.encode('utf-8'))
            else:
                print(message)
        except:
            print('error')
            client_socket.close()
            break


def send_message():  # отправить сообщение
    while True:
        message = f'{nickname}: {input("")}'
        client_socket.send(message.encode('utf-8'))


# поток получения сообщений
receive_thread = threading.Thread(target=receive_messages)
receive_thread.start()

# поток отправки сообщений
send_thread = threading.Thread(target=send_message)
send_thread.start()
```

Исполнение программы:

Сервер следит за подключениями клиентов - их адреса и ники.

Три клиента, в интерфейсе каждого показываются все сообщения, отправленные другими пользователями

Вывод:

В процессе выполнения лабораторной работы были получены практические навыки и умения реализации web-серверов и использования сокетов. Помимо этого, были изучены и применены потоки для реализации многопользовательского чата.