Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе 1

по дисциплине Веб-программирование

Автор: Александрин Антон

Факультет: ИКТ

Группа: К33422

Преподаватель: Говоров А.И.

Санкт-Петербург 2021

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.

```
# server.py
import socket
import time


conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # соединение
по протоколу TCP
conn.bind(('127.0.0.1', 5000))
conn.listen(5)

while True:
   try:
       clientsocket, address = conn.accept() # устанавливаем
соединение с клиентом
       data = clientsocket.recv(4096) # получаем закодированные
данные
       udata = data.decode('utf-8')
       clientsocket.send(b'Hello, client') # отправляем ответ
       print(time.ctime() + '\n' + udata)

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

```
# client.py
import socket

conn = socket.socket()
conn.connect(('127.0.0.1', 5000)) # подключаемся к серверу

conn.send(b'Hello, server') # отправляем закодированное сообщение
data = conn.recv(4096) # получаем и расшифровываем ответ
udata = data.decode('utf-8')
print(udata)

conn.close()
```

```
^Ch3ic@yoga:~/web/Lr1$ p task_1/server.py
Tue Sep 14 10:39:01 2021
Hello, server
Tue Sep 14 10:39:43 2021
Hello, server
```

```
h3ic@yoga:~/web/Lr1$ p task_1/client.py
Hello, client
h3ic@yoga:~/web/Lr1$ p task_1/client.py
Hello, client
h3ic@yoga:~/web/Lr1$
```

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 b: Решение квадратного уравнения.

```python
# server.py
import socket
import json
from math import sqrt

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.bind(('127.0.0.1', 5000))
conn.listen(5)

while True:
   try:
        clientsocket, address = conn.accept()
        data = clientsocket.recv(4096)
        udata = data.decode('utf-8')
        dict_data = json.loads(udata)
        a, b, c = dict_data

        d = b * b - 4 * a * c
        if d < 0:
            clientsocket.send(b'd < 0; invalid values')
        elif d > 0:
            x1 = (-b + sqrt(d)) / 2 * a
            x2 = (-b - sqrt(d)) / 2 * a
            clientsocket.send(bytes(f'possible x values are {x1},
{x2}', 'utf-8'))
        elif d == 0:
```

```
            x = (-b) / 2 * a
            clientsocket.send(bytes(f'x value is {x}', 'utf-8'))

    except KeyboardInterrupt:
        conn.close()
        break

    except Exception as e:
        print(e)
        conn.close()
        break
```

```
# client.py
import socket

conn = socket.socket()
conn.connect(('127.0.0.1', 5000))

a, b, c = input('enter the a, b, c values: ').split()
if not a:
    print('Invalid a value')
    conn.close()

values_dict = f'[{a}, {b}, {c}]'
conn.send(bytes(values_dict, 'utf-8')) # отправляем на сервер объект
с введенными пользователем данными

data = conn.recv(4096) # получаем, декодируем и выводим ответ
пользователю
udata = data.decode('utf-8')
print(udata)

conn.close()
```

```
^Ch3ic@yoga:~/web/Lr1$ p task_2/server.py
```

```
h3ic@yoga:~/web/Lr1$ p task_2/client.py
enter the a, b, c values: 6 2 1
d < 0; invalid values
h3ic@yoga:~/web/Lr1$ p task_2/client.py
enter the a, b, c values: 12 1 0
possible x values are 0.0, -12.0
h3ic@yoga:~/web/Lr1$ p task_2/client.py
enter the a, b, c values: 1 0 0
x value is 0.0
h3ic@yoga:~/web/Lr1$
```

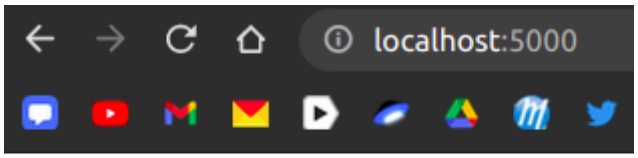
3. Реализовать серверную часть приложения. Клиент подключается к серверу. В ответ клиент получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

```python
# server.py
import socket

server = socket.socket()
server.bind(('127.0.0.1', 5000))
server.listen(5)

while True:
    client, (client_host, client_port) = server.accept()
    client.recv(4096)
    response_type = 'HTTP/1.1 200 OK\n'
    headers = 'Content-Type: text/html\n\n'

    with open('task_3/index.html', 'r') as f:
        body = f.read()
    response = response_type + headers + body # составляем читаемый браузером http ответ на запрос с основным содержанием в body
    client.send(response.encode('utf-8'))
    client.close()
```

```python
# client.py
import socket

conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
conn.connect(('127.0.0.1', 5000))

conn.send('request'.encode('utf-8'))
data = conn.recv(4096)
udata = data.decode('utf-8')
print(udata)
```

```
h3ic@yoga:~/web/Lr1$ p task_3/server.py
```

```
h3ic@yoga:~/web/Lr1$ p task_3/client.py
HTTP/1.1 200 OK
Content-Type: text/html

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hi</title>
</head>
<body>
<h1>Hi</h1>
</body>
</html>
h3ic@yoga:~/web/Lr1$
```



Hi

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

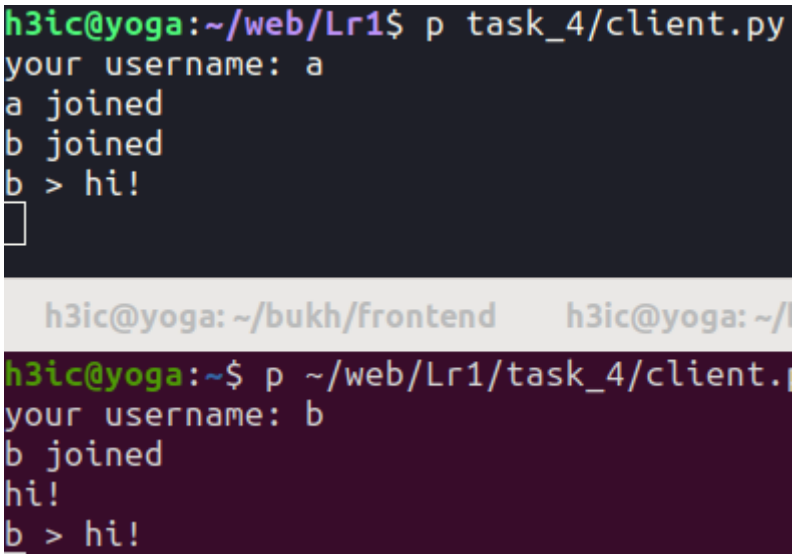
```python
# server.py
import socket
import threading

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('127.0.0.1', 5002))
server.listen()

clients = []
usernames = []


def broadcast(message):
    for client in clients:
        client.send(message)
```

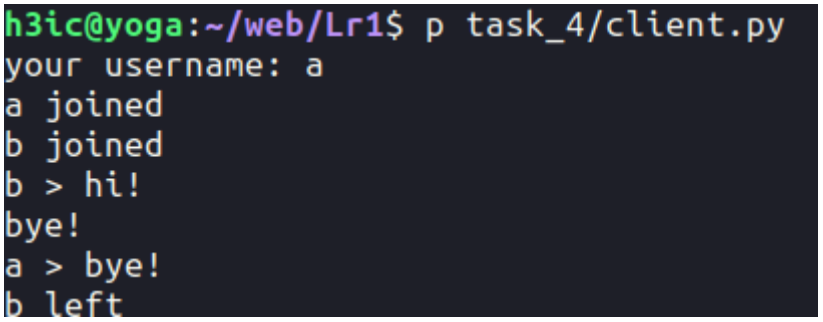
```
def handle(client):
    while True:
        try:
            message = client.recv(4096)
            broadcast(message)

        except Exception as e:
            print('Exception:', e)
            index = clients.index(client)
            clients.remove(client)
            client.close()

            username = usernames[index]
            broadcast(f'{username} left'.encode('utf-8'))
            usernames.remove(username)
            break


def receive(): # основная функция, в которой устанавливается
соединение и принимается имя пользователя, после чего запускается тред
для приема сообщений
    while True:
        try:
            client, client_address = server.accept()
            print(f'accepted connection from {client_address}')

            client.send('NICKNAME'.encode('utf-8'))
            username = client.recv(4096).decode('utf-8')
            clients.append(client)
            usernames.append(username)
            broadcast(f'{username} joined'.encode('utf-8'))

            handle_thread = threading.Thread(target=handle,
args=(client,))
            handle_thread.start()

        except Exception as e:
            print('Exception:', e)
            broadcast(f'')


receive()
```

client.py

```python
import socket
import threading

username = input("your username: ")

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('127.0.0.1', 5002))

# две функции для приема и отправки сообщений, не блокирующие друг друга

def receive(): # получение сообщений от сервера и отправка юзернейма при соответствующем запросе
    while True:
        try:
            message = client.recv(4096).decode('utf-8')
            if message == 'NICKNAME':
                client.send(username.encode('utf-8'))
            else:
                print(message)

        except Exception as e:
            print(e)
            client.close()
            break


def send(): # принимает на вход сообщение и отправляет на сервер
    while True:
        message = input()
        client.send(f'{username} > {message}'.encode('utf-8'))


send_thread = threading.Thread(target=send)
recv_thread = threading.Thread(target=receive)
send_thread.start()
recv_thread.start()
```

```
h3ic@yoga:~/web/Lr1$ p task_4/server.py
accepted connection from ('127.0.0.1', 34230)
accepted connection from ('127.0.0.1', 34276)
```





Вывод: в ходе проделанной работы были реализованы серверные и клиентские скрипты с помощью библиотек socket и threading.