Санкт-Петербургский Национальный Исследовательский Университет

Информационных Технологий, Механики и Оптики

Факультет Инфокоммуникационных Технологий

Лабораторная работа №6.

Выполнил

Сидненко Дмитрий

Проверила

Марченко Е. В.

Санкт-Петербург, 2025

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ

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.

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.

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.

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

ХОД РАБОТЫ

Задание 1.

Для реализации клиентской и серверной части приложения были созданы файлы client.py и server.py. При подключении, клиент отсылает серверу сообщение «Hello, server», сообщение отражается на стороне сервера и сервер в ответ отсылает клиенту сообщение «Hello, client».

Код для запуска сервера представлен на рисунке 1, для запуска клиента – на рисунке 2.

```python
import socket


server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('localhost', 12345))
server_socket.listen(1)

print("Сервер запущен. Ожидание подключения...")


client_socket, client_address = server_socket.accept()
print(f"Подключен клиент: {client_address}")

# Получаем сообщение от клиента
data = client_socket.recv(1024).decode('utf-8')
print(f"Получено от клиента: {data}")


client_socket.send("Hello, client".encode('utf-8'))


client_socket.close()
server_socket.close()
```

Рисунок 1 – Код для запуска сервера

```python
import socket


client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 12345))  # Подключаемся к серверу


client_socket.send("Hello, server".encode('utf-8'))


response = client_socket.recv(1024).decode('utf-8')
print(f"Получено от сервера: {response}")

# Закрываем соединение
client_socket.close()
```

Рисунок 2 – Код для запуска клиента

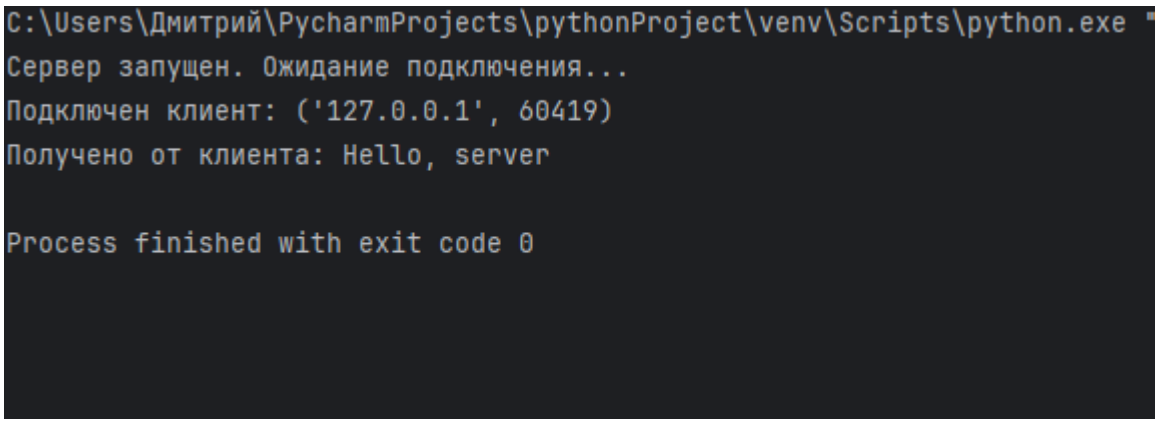

Рисунок 3 – Сообщение от клиента серверу

Рисунок 4 – Сообщение от сервера клиенту

Задание 2.

Для реализации клиентской и серверной часть приложения также были созданы два файла, однако в этом случае клиент запрашивает у сервера выполнение математической операции отправляя ему параметры, которые вводятся с клавиатуры, сервер обрабатывает полученные данные и возвращает результат клиенту. Согласно спискам группы, мой вариант: b. Решение квадратного уравнения.

```python
import socket
import math


server_socket = socket.socket()
server_socket.bind(('localhost', 12345))
server_socket.listen(1)
print("Сервер запущен. Ждём подключения...")


conn, addr = server_socket.accept()
print("Клиент подключен:", addr)


data = conn.recv(1024).decode()
a, b, c = map(float, data.split())

# Решаем уравнение
discriminant = b**2 - 4*a*c
if discriminant < 0:
    result = "Нет корней"
elif discriminant == 0:
    x = -b / (2*a)
    result = f"Один корень: {x}"
else:
    x1 = (-b + math.sqrt(discriminant)) / (2*a)
    x2 = (-b - math.sqrt(discriminant)) / (2*a)
    result = f"Два корня: {x1} и {x2}"


conn.send(result.encode())


conn.close()
server_socket.close()
```

Рисунок 5 – Код для запуска сервера

```python
import socket

# Создаём клиентский сокет
client_socket = socket.socket()
client_socket.connect(('localhost', 12345))

# Вводим коэффициенты
print("Введите коэффициенты a, b, c через пробел:")
a, b, c = input().split()


client_socket.send(f"{a} {b} {c}".encode())

# Получаем ответ
result = client_socket.recv(1024).decode()
print("Результат:", result)


client_socket.close()
```

Рисунок 6 – Код для запуска клиента

```
C:\Users\Дмитрий\PycharmProjects\pythonProject\venv\Scripts\pythor
Введите коэффициенты a, b, c через пробел:
3 4 5
Результат: Нет корней

Process finished with exit code 0
```

Рисунок 7 – Расчет корней квадратного уравнения

Задание 3.

В этом задании была реализована серверную часть приложения, в которой клиент подключается к серверу и в ответ клиент получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

```python
from http.server import BaseHTTPRequestHandler, HTTPServer

class MyServer(BaseHTTPRequestHandler):  1 usage
    def do_GET(self):
        self.send_response(200)
        self.send_header( keyword: 'Content-type',  value: 'text/html; charset=utf-8')  # Указываем кодировку
        self.end_headers()

        with open('index.html', 'rb') as file:  # Открываем файл
            self.wfile.write(file.read())


def run():  1 usage
    server_address = ('', 8000)
    httpd = HTTPServer(server_address, MyServer)
    print('Сервер запущен на http://localhost:8000')
    httpd.serve_forever()


if __name__ == '__main__':
    run()
```

Рисунок 8 – Код для запуска сервера

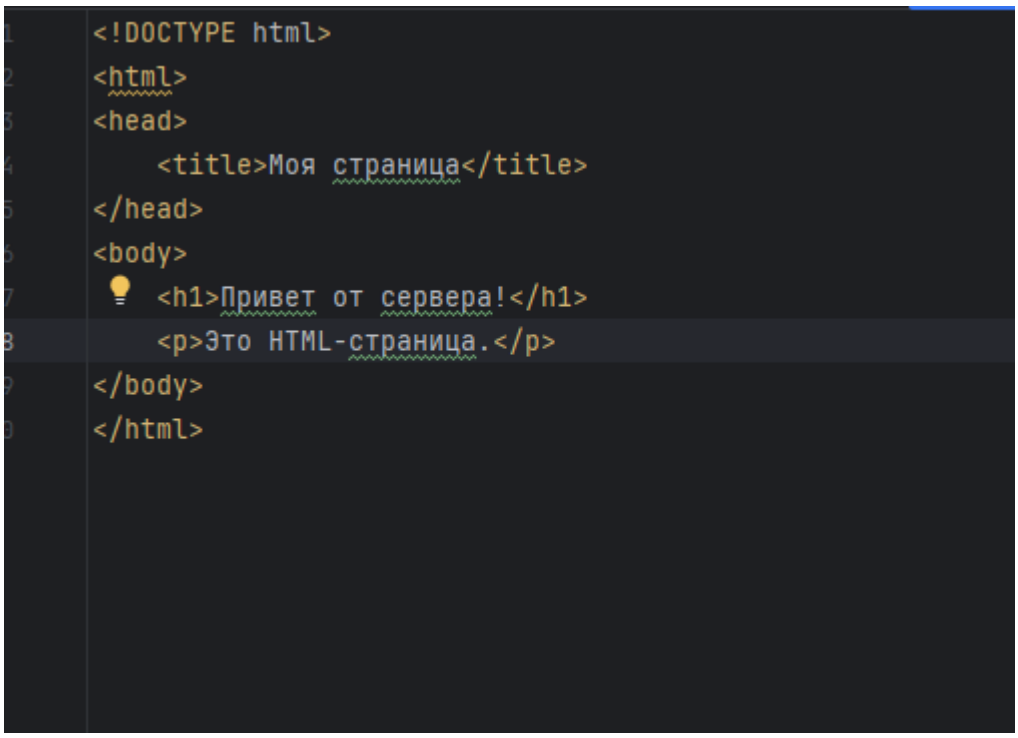

Рисунок 9 – Код файла index.html

Рисунок 10 – Полученное http сообщение

Задание 4.

В этом задании был реализован многопользовательский чат. Для реализации было создано два файла – клиент и сервер. Для проверки чата необходимо запустить файл клиента несколько раз в разных терминалах.

```python
def handle_client(client_socket, address):  1 usage
    clients.append(client_socket)

    try:
        while True:
            message = client_socket.recv(1024).decode('utf-8')
            if not message:
                break
            print(f"{address}: {message}")
            broadcast(message, client_socket)
    except:
        print(f"Клиент {address} отключился")
    finally:
        clients.remove(client_socket)
        client_socket.close()


def broadcast(message, sender_socket):  1 usage
    for client in clients:
        if client != sender_socket:
            try:
                client.send(message.encode('utf-8'))
            except:
                clients.remove(client)


def start_server():  1 usage
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(('0.0.0.0', 5555))
    server.listen(5)
    print("Сервер чата запущен. Ожидание подключений...")

    while True:
        client_socket, address = server.accept()
        thread = threading.Thread(target=handle_client, args=(client_socket, address))
        thread.start()


if __name__ == "__main__":
    start_server()
```

Рисунок 11 – Код для запуска сервера

```python
import ...

def receive_messages(client_socket):  1 usage
    while True:
        try:
            message = client_socket.recv(1024).decode('utf-8')
            print(message)
        except:
            print("Соединение разорвано!")
            client_socket.close()
            break

def start_client():  1 usage
    nickname = input("Введите ваш никнейм: ")
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 5555))

    receive_thread = threading.Thread(target=receive_messages, args=(client_socket,))
    receive_thread.start()

    while True:
        message = input()
        if message.lower() == 'exit':
            break
        full_message = f"{nickname}: {message}"
        client_socket.send(full_message.encode('utf-8'))

    client_socket.close()

if __name__ == "__main__":
    start_client()
```

Рисунок 12 – Код для запуска клиентов

```
Сервер чата запущен. Ожидание подключений...
Новое подключение: ('127.0.0.1', 60588)
('127.0.0.1', 60588): test1: Hello
Новое подключение: ('127.0.0.1', 60601)
('127.0.0.1', 60601): test2: Hi
Новое подключение: ('127.0.0.1', 60604)
('127.0.0.1', 60604): test3: Welcome
Новое подключение: ('127.0.0.1', 60607)
('127.0.0.1', 60607): test4: gg
```

Рисунок 13 – Тест многопользовательского чата

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были реализованы клиентская и серверная часть приложения с разными задачами.

1. Клиент отсылает серверу сообщение, которое отражается на стороне сервера, а сервер в ответ отсылает сообщение клиенту.

2. Клиент запрашивает у сервера выполнение математической операции, сервер обрабатывает полученные данные и возвращает результат клиенту. Вариант: b. Решение квадратного уравнения.

3. Клиент подключается к серверу и получает http-сообщение, содержащее html-страницу, которую сервер подгружает из файла index.html.

Также был реализован многопользовательский чат.