

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 .....	3
ХОД РАБОТЫ.....	4
Задание 1.....	4
Задание 2.....	5
Задание 3.....	7
Задание 4.....	8
ЗАКЛЮЧЕНИЕ .....	11

## ЦЕЛЬ РАБОТЫ

1. Реализовать клиентскую и серверную часть приложения. Клиент отправляет серверу сообщение «Hello, server». Сообщение должно отразиться на стороне сервера. Сервер в ответ отправляет 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 чата 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 отправляет серверу сообщение «Hello, server», сообщение отражается на стороне сервера и сервер в ответ отправляет клиенту сообщение «Hello, client».

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

```
1 import socket
2
3
4 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 server_socket.bind(('localhost', 12345))
6 server_socket.listen(1)
7
8 print("Сервер запущен. Ожидание подключения...")
9
10
11 client_socket, client_address = server_socket.accept()
12 print(f"Подключен клиент: {client_address}")
13
14 # Получаем сообщение от клиента
15 data = client_socket.recv(1024).decode('utf-8')
16 print(f"Получено от клиента: {data}")
17
18 client_socket.send("Hello, client".encode('utf-8'))
19
20
21
22 client_socket.close()
23 server_socket.close()
```

Рисунок 1 – Код для запуска сервера

```
1 import socket
2
3
4 client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
5 client_socket.connect(('localhost', 12345)) # Подключаемся к серверу
6
7
8 client_socket.send("Hello, server".encode('utf-8'))
9
10
11 response = client_socket.recv(1024).decode('utf-8')
12 print(f"Получено от сервера: {response}")
13
14 # Закрываем соединение
15 client_socket.close()
```

Рисунок 2 – Код для запуска клиента

```
C:\Users\Дмитрий\PycharmProjects\pythonProject\venv\Scripts\python.exe "
Сервер запущен. Ожидание подключения...
Подключен клиент: ('127.0.0.1', 60419)
Получено от клиента: Hello, server

Process finished with exit code 0
```

Рисунок 3 – Сообщение от клиента серверу

```
C:\Users\Дмитрий\PycharmProjects\pythonProject\venv\Script
Получено от сервера: Hello, client

Process finished with exit code 0
```

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.

```

1  import socket
2  import math
3
4
5  server_socket = socket.socket()
6  server_socket.bind(('localhost', 12345))
7  server_socket.listen(1)
8  print("Сервер запущен. Ждём подключения...")
9
10
11 conn, addr = server_socket.accept()
12 print("Клиент подключен:", addr)
13
14
15 data = conn.recv(1024).decode()
16 a, b, c = map(float, data.split())
17
18 # Решаем уравнение
19 discriminant = b**2 - 4*a*c
20 if discriminant < 0:
21     result = "Нет корней"
22 elif discriminant == 0:
23     x = -b / (2*a)
24     result = f"Один корень: {x}"
25 else:
26     x1 = (-b + math.sqrt(discriminant)) / (2*a)
27     x2 = (-b - math.sqrt(discriminant)) / (2*a)
28     result = f"Два корня: {x1} и {x2}"
29
30
31 conn.send(result.encode())
32
33
34 conn.close()
35 server_socket.close()

```

Рисунок 5 – Код для запуска сервера

```

1  import socket
2
3  # Создаём клиентский сокет
4  client_socket = socket.socket()
5  client_socket.connect(('localhost', 12345))
6
7  # Вводим коэффициенты
8  print("Введите коэффициенты a, b, c через пробел:")
9  a, b, c = input().split()
10
11
12 client_socket.send(f"{a} {b} {c}".encode())
13
14 # Получаем ответ
15 result = client_socket.recv(1024).decode()
16 print("Результат:", result)
17
18
19 client_socket.close()

```

Рисунок 6 – Код для запуска клиента

```
C:\Users\Дмитрий\PycharmProjects\pythonProject\venv\Scripts\python
Введите коэффициенты а, b, с 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.

```
from http.server import BaseHTTPRequestHandler, HTTPServer

class MyServer(BaseHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.send_header('Content-type', '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():
    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

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Моя страница</title>
5  </head>
6  <body>
7      ⚡ <h1>Привет от сервера!</h1>
8      <p>Это HTML-страница.</p>
9  </body>
10 </html>

```

Рисунок 9 – Код файла index.html

---

**Привет от сервера!**

Это HTML-страница.

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.



```

7 def handle_client(client_socket, address): 1 usage
9     clients.append(client_socket)
10
11     try:
12         while True:
13             message = client_socket.recv(1024).decode('utf-8')
14             if not message:
15                 break
16             print(f'{address}: {message}')
17             broadcast(message, client_socket)
18         except:
19             print(f'Клиент {address} отключился')
20         finally:
21             clients.remove(client_socket)
22             client_socket.close()
23
24
25 def broadcast(message, sender_socket): 1 usage
26     for client in clients:
27         if client != sender_socket:
28             try:
29                 client.send(message.encode('utf-8'))
30             except:
31                 clients.remove(client)
32
33
34 def start_server(): 1 usage
35     server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
36     server.bind(('0.0.0.0', 5555))
37     server.listen(5)
38     print("Сервер чата запущен. Ожидание подключений...")
39
40     while True:
41         client_socket, address = server.accept()
42         thread = threading.Thread(target=handle_client, args=(client_socket, address))
43         thread.start()
44
45
46 if __name__ == "__main__":
47     start_server()

```

Рисунок 11 – Код для запуска сервера

```

1 > import ...
3
4 def receive_messages(client_socket): 1 usage
5     while True:
6         try:
7             message = client_socket.recv(1024).decode('utf-8')
8             print(message)
9         except:
10            print("Соединение разорвано!")
11            client_socket.close()
12            break
13
14 def start_client(): 1 usage
15     nickname = input("Введите ваш никнейм: ")
16     client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17     client_socket.connect(('localhost', 5555))
18
19     receive_thread = threading.Thread(target=receive_messages, args=(client_socket,))
20     receive_thread.start()
21
22     while True:
23         message = input()
24         if message.lower() == 'exit':
25             break
26         full_message = f"{nickname}: {message}"
27         client_socket.send(full_message.encode('utf-8'))
28
29     client_socket.close()
30
31 if __name__ == "__main__":
32     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 отправляет серверу сообщение, которое отражается на стороне сервера, а сервер в ответ отправляет 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.