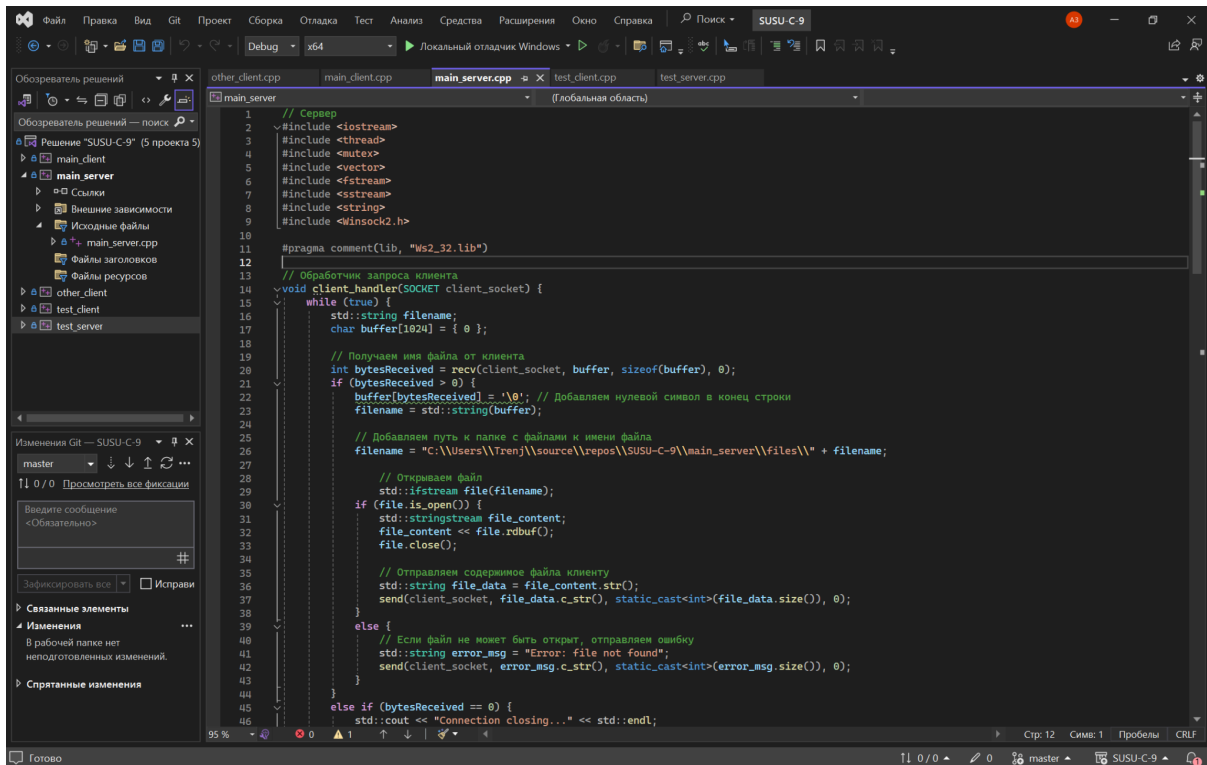
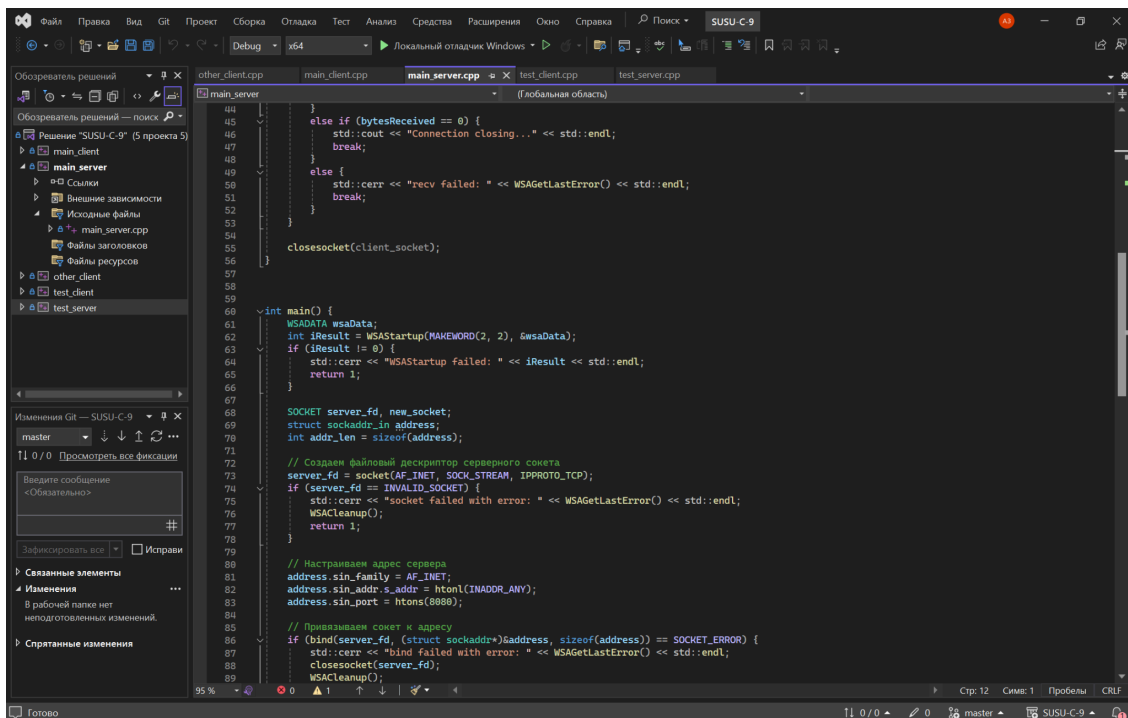


# Самостоятельная работа



The screenshot shows the Visual Studio IDE with the file `main_server.cpp` open. The code implements a `client_handler` function that processes incoming client requests. It includes headers for `iostream`, `thread`, `mutex`, `vector`, `fstream`, `sstream`, `string`, and `Winsock2.h`. The function `client_handler` takes a `SOCKET` as an argument and enters a `while` loop. It receives data from the client into a `buffer`, constructs a `filename` by concatenating a base path with the received data, and attempts to open and read the file. The file content is then sent back to the client. Error handling is provided for file not found and connection closing scenarios.

```
1 // Сервер
2 #include <iostream>
3 #include <thread>
4 #include <mutex>
5 #include <vector>
6 #include <fstream>
7 #include <sstream>
8 #include <string>
9 #include <Winsock2.h>
10
11 #pragma comment(lib, "Ws2_32.Lib")
12
13 // Обработка запроса клиента
14 void client_handler(SOCKET client_socket) {
15     while (true) {
16         std::string filename;
17         char buffer[1024] = { 0 };
18
19         // Получаем имя файла от клиента
20         int bytesReceived = recv(client_socket, buffer, sizeof(buffer), 0);
21         if (bytesReceived > 0) {
22             buffer[bytesReceived] = '\\0'; // Добавляем нулевой символ в конец строки
23             filename = std::string(buffer);
24
25             // Добавляем путь к каталогу с файлами к имени файла
26             filename = "C:\\Users\\Trenj\\source\\repos\\SUSU-C-9\\main_server\\files\\" + filename;
27
28             // Открываем файл
29             std::ifstream file(filename);
30             if (file.is_open()) {
31                 std::stringstream file_content;
32                 file_content << file.rdbuf();
33                 file.close();
34
35                 // Отправляем содержимое файла клиенту
36                 std::string file_data = file_content.str();
37                 send(client_socket, file_data.c_str(), static_cast<int>(file_data.size()), 0);
38
39             } else {
40                 // Если файл не может быть открыт, отправляем ошибку
41                 std::string error_msg = "Error: file not found";
42                 send(client_socket, error_msg.c_str(), static_cast<int>(error_msg.size()), 0);
43             }
44         } else if (bytesReceived == 0) {
45             std::cout << "Connection closing..." << std::endl;
46         }
```



The screenshot shows the Visual Studio IDE with the file `main_server.cpp` open, displaying the `main` function. It initializes Winsock, sets up a server socket, and binds it to a specific address and port (8080). The server then enters a loop where it accepts incoming connections and spawns a new thread to handle each client request using the `client_handler` function.

```
44 }
45
46 else if (bytesReceived == 0) {
47     std::cout << "Connection closing..." << std::endl;
48     break;
49 }
50
51 else {
52     std::cerr << "recv failed: " << WSAGetLastError() << std::endl;
53     break;
54 }
55
56 closesocket(client_socket);
57
58
59 int main() {
60     WSADATA wsaData;
61     int iResult = WSASStartup(MAKEWORD(2, 2), &wsaData);
62     if (iResult != 0) {
63         std::cerr << "WSAStartup failed: " << iResult << std::endl;
64         return 1;
65     }
66
67     SOCKET server_fd, new_socket;
68     struct sockaddr_in address;
69     int addr_len = sizeof(address);
70
71     // Создаем файловый дескриптор серверного сокета
72     server_fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
73     if (server_fd == INVALID_SOCKET) {
74         std::cerr << "socket failed with error: " << WSAGetLastError() << std::endl;
75         WSACleanup();
76         return 1;
77     }
78
79     // Настраиваем адрес сервера
80     address.sin_family = AF_INET;
81     address.sin_addr.s_addr = htonl(INADDR_ANY);
82     address.sin_port = htons(8080);
83
84     // Привязываем сокет к адресу
85     if (bind(server_fd, (struct sockaddr*)&address, sizeof(address)) == SOCKET_ERROR) {
86         std::cerr << "bind failed with error: " << WSAGetLastError() << std::endl;
87         closesocket(server_fd);
88         WSACleanup();
89     }
```

