Программирование на C++

Минцифры России

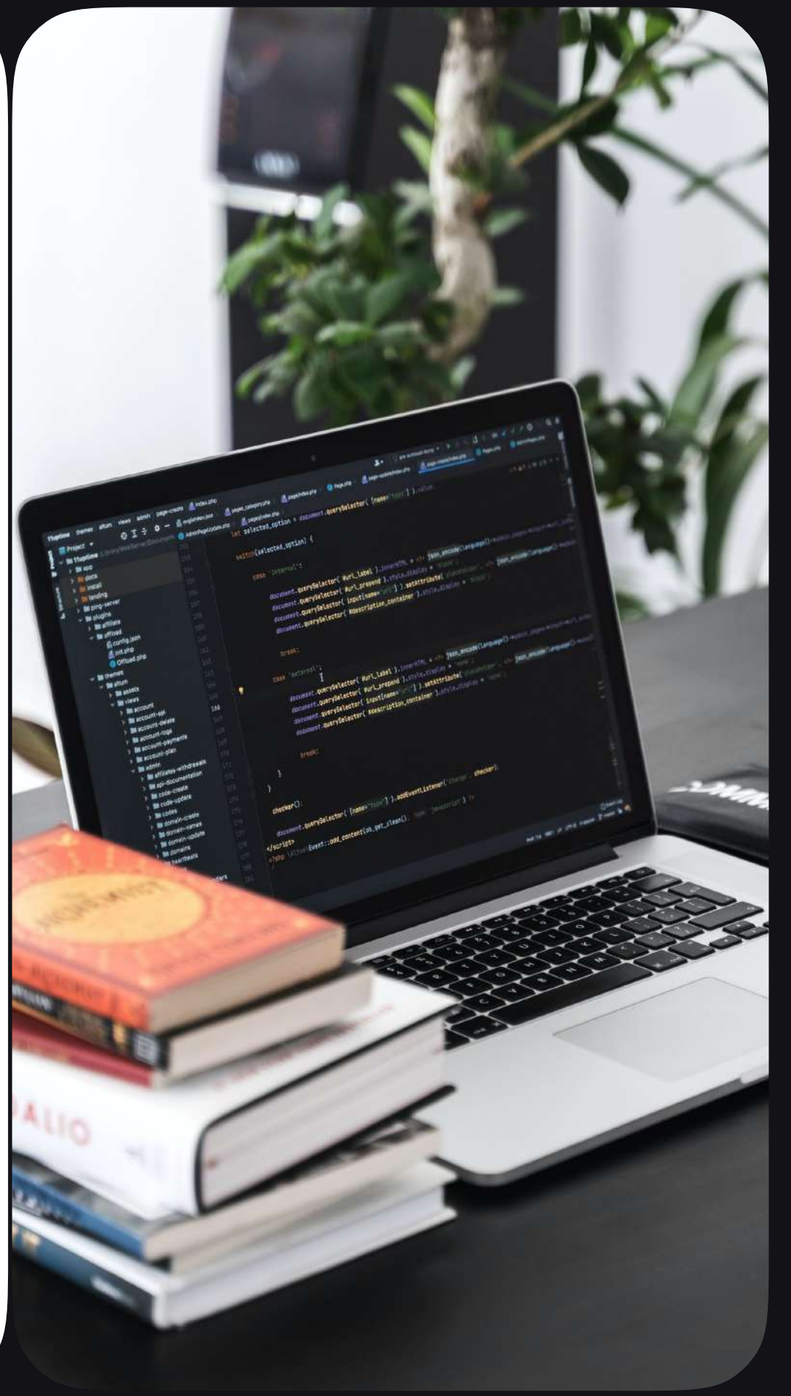
UCHi DOMA

20.35 УНИВЕРСИТЕТ

Урок 11 Модуль 4

Особенности проектирование клиент-серверных приложений (продолжение)

Полезные материалы



Цели урока

- Разработать клиент-серверное приложение «Чат».
- Отработать на практике написание клиент-серверных приложений на C++.

Откройте проекты, созданные на прошлом уроке

Сервер

```cpp
#pragma comment(lib, "ws2_32.lib")
#include <winsock2.h>
#include <iostream>

#pragma warning(disable: 4996)

int main(int argc, char* argv[]) {
    //WSAStartup
    WSAData wsaData;
    WORD DLLVersion = MAKEWORD(2, 1);
    if (WSAStartup(DLLVersion, &wsaData) != 0) {
        std::cout << "Error" << std::endl;
        exit(1);
    }
    SOCKADDR_IN addr;
    int sizeofaddr = sizeof(addr);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    addr.sin_port = htons(1111);
    addr.sin_family = AF_INET;

    SOCKET sListen = socket(AF_INET, SOCK_STREAM, NULL);
    bind(sListen, (SOCKADDR*)&addr, sizeof(addr));
    listen(sListen, SOMAXCONN);

    SOCKET newConnection;
    newConnection = accept(sListen, (SOCKADDR*)&addr, &sizeofaddr);

    if (newConnection == 0) {
        std::cout << "Error #2\n";
    }
    else {
        std::cout << "Client Connected!\n";
    }
    system("pause");
    return 0;
}
```

Откройте проекты, созданные на прошлом уроке

Клиент

```cpp
#pragma comment(lib, "ws2_32.lib")
#include <winsock2.h>
#include <iostream>

#pragma warning(disable: 4996)

int main(int argc, char* argv[]) {
    //WSAStartup
    WSAData wsaData;
    WORD DLLVersion = MAKEWORD(2, 1);
    if (WSAStartup(DLLVersion, &wsaData) != 0) {
        std::cout << "Error" << std::endl;
        exit(1);
    }

    SOCKADDR_IN addr;
    int sizeofaddr = sizeof(addr);
    addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    addr.sin_port = htons(1111);
    addr.sin_family = AF_INET;

    SOCKET Connection = socket(AF_INET, SOCK_STREAM, NULL);
    if (connect(Connection, (SOCKADDR*)&addr, sizeof(addr)) != 0) {
        std::cout << "Error: failed connect to server.\n";
        return 1;
    }
    std::cout << "Connected!\n";

    system("pause");
    return 0;
}
```

Пусть сервер отправляет сообщение клиенту

Добавим функцию send() в код сервера

```
if (newConnection == 0) {
    std::cout << "Error #2\n";
 }
 else {
    std::cout << "Client Connected!\n";
    char msg[128] = "Hello!";
    send(newConnection,msg,sizeof(msg),NULL);
 }
```

Создадим текстовую переменную и запишем в нее «Hello».

Отправим сообщение

Клиент будет принимать сообщение

Добавим функцию recv() в код клиента

```
std::cout << "Connected!\n";
char msg[128];
recv(Connection, msg, sizeof(msg),NULL);
std::cout << msg << std::endl;
```

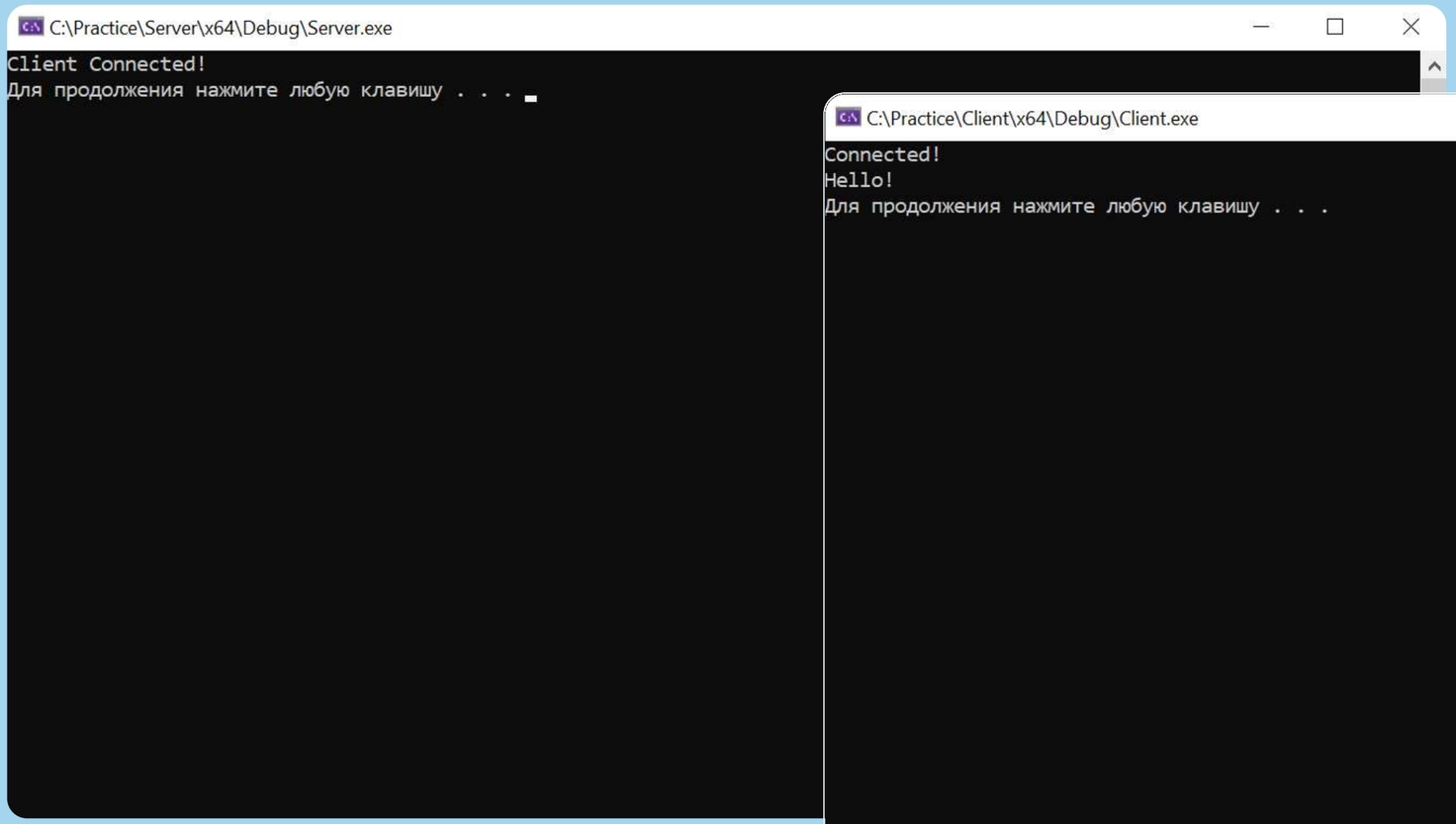
Создадим текстовую переменную msg.

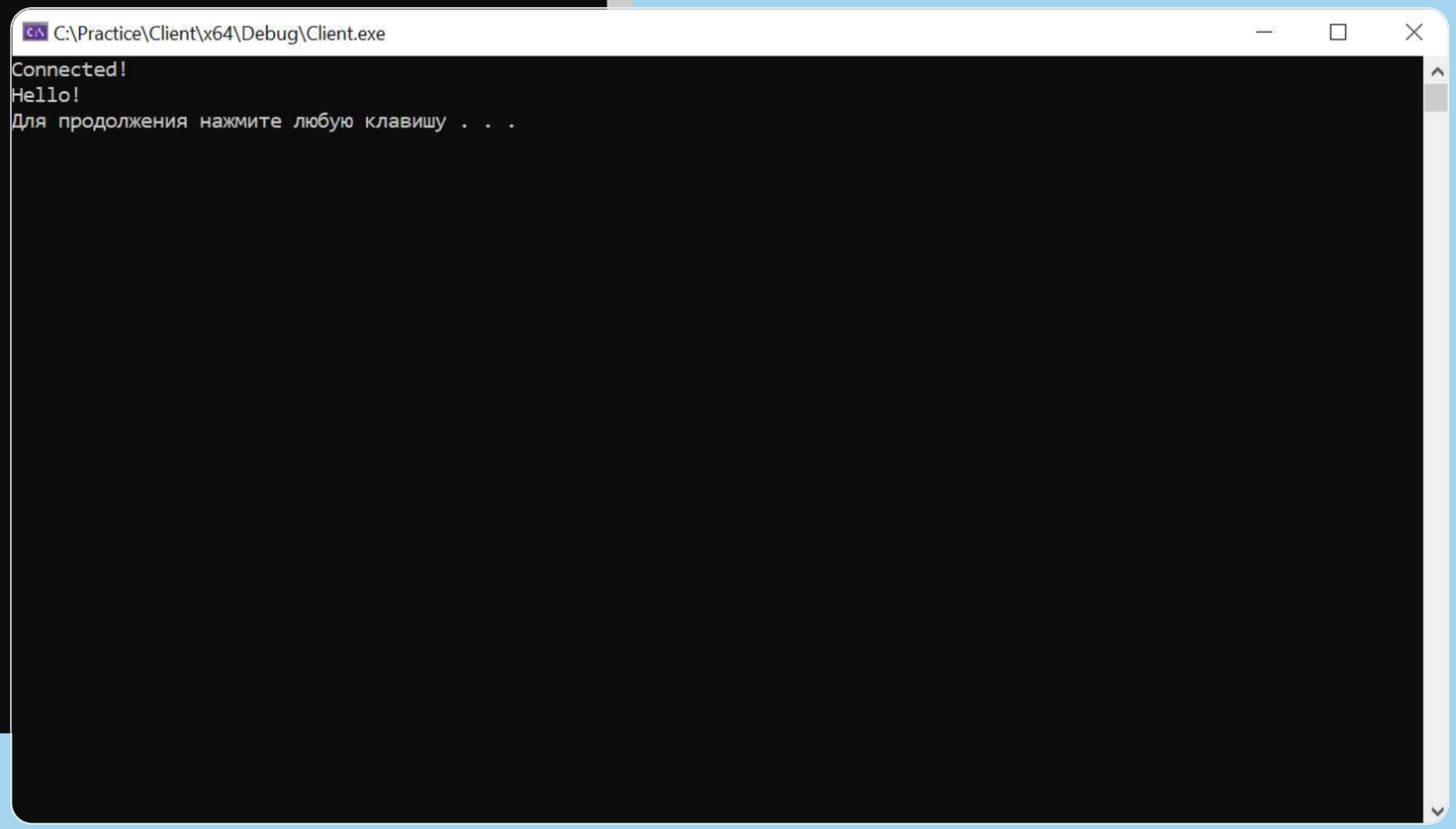
Получим сообщение в переменную msg.

Проверьте работу приложений.

Запустите сначала server затем client.

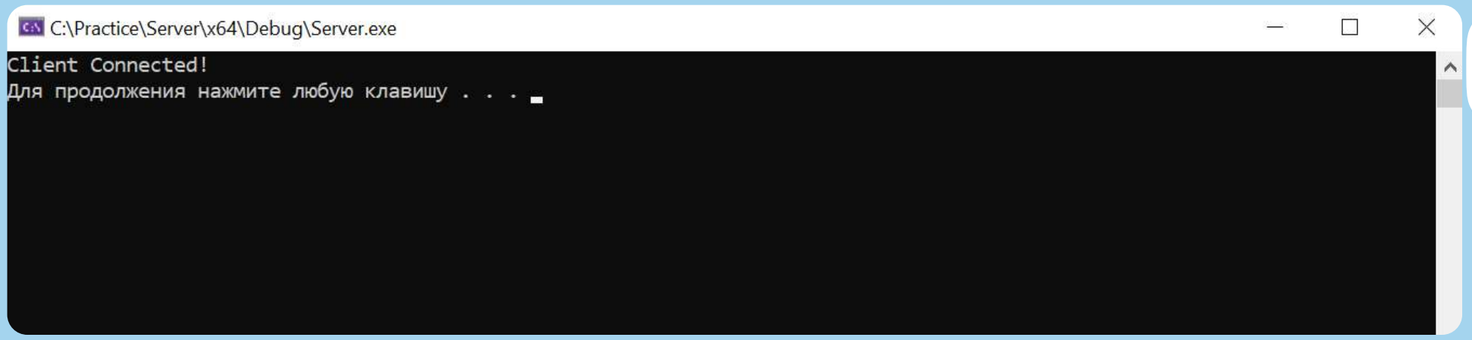
Клиент получил сообщение «Hello!».



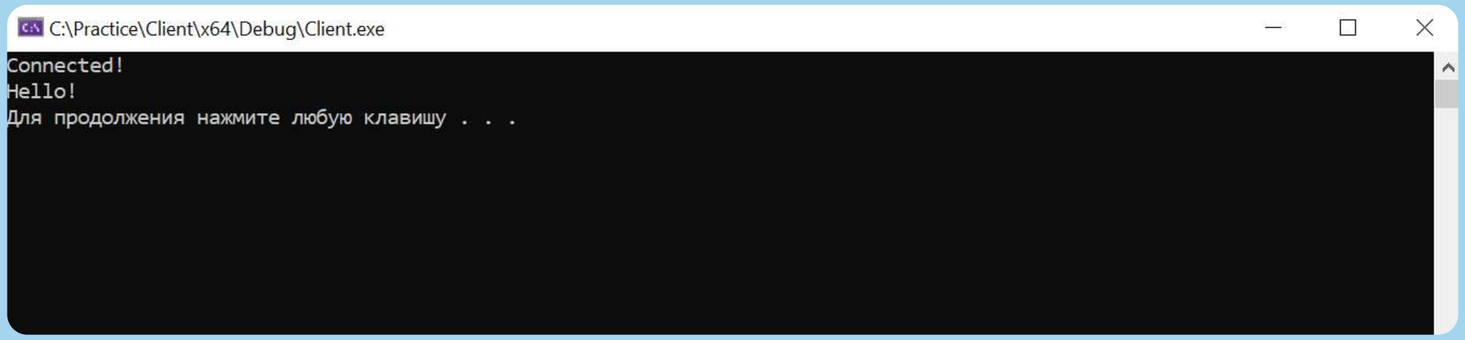


Запустите Server и несколько клиентов.

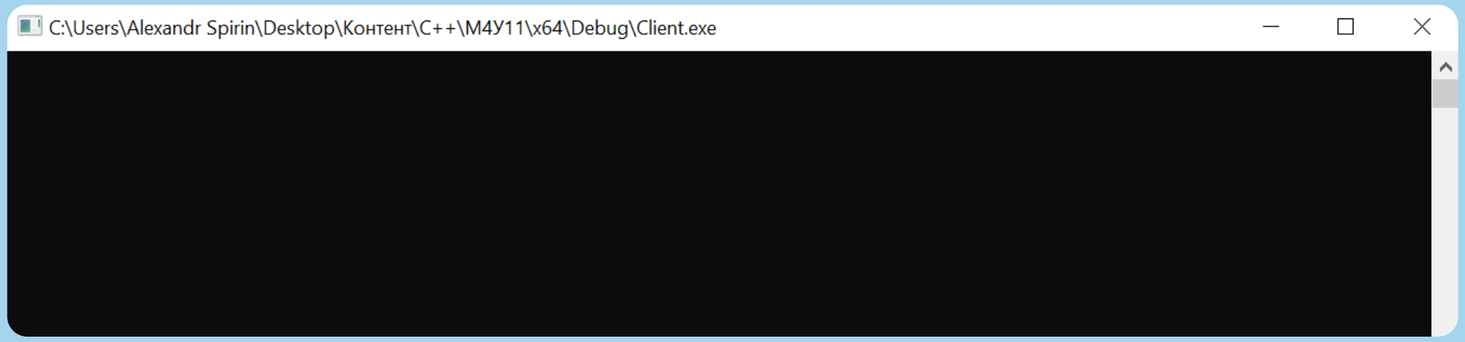
Сколько клиентов смогли подключить?



Сервер



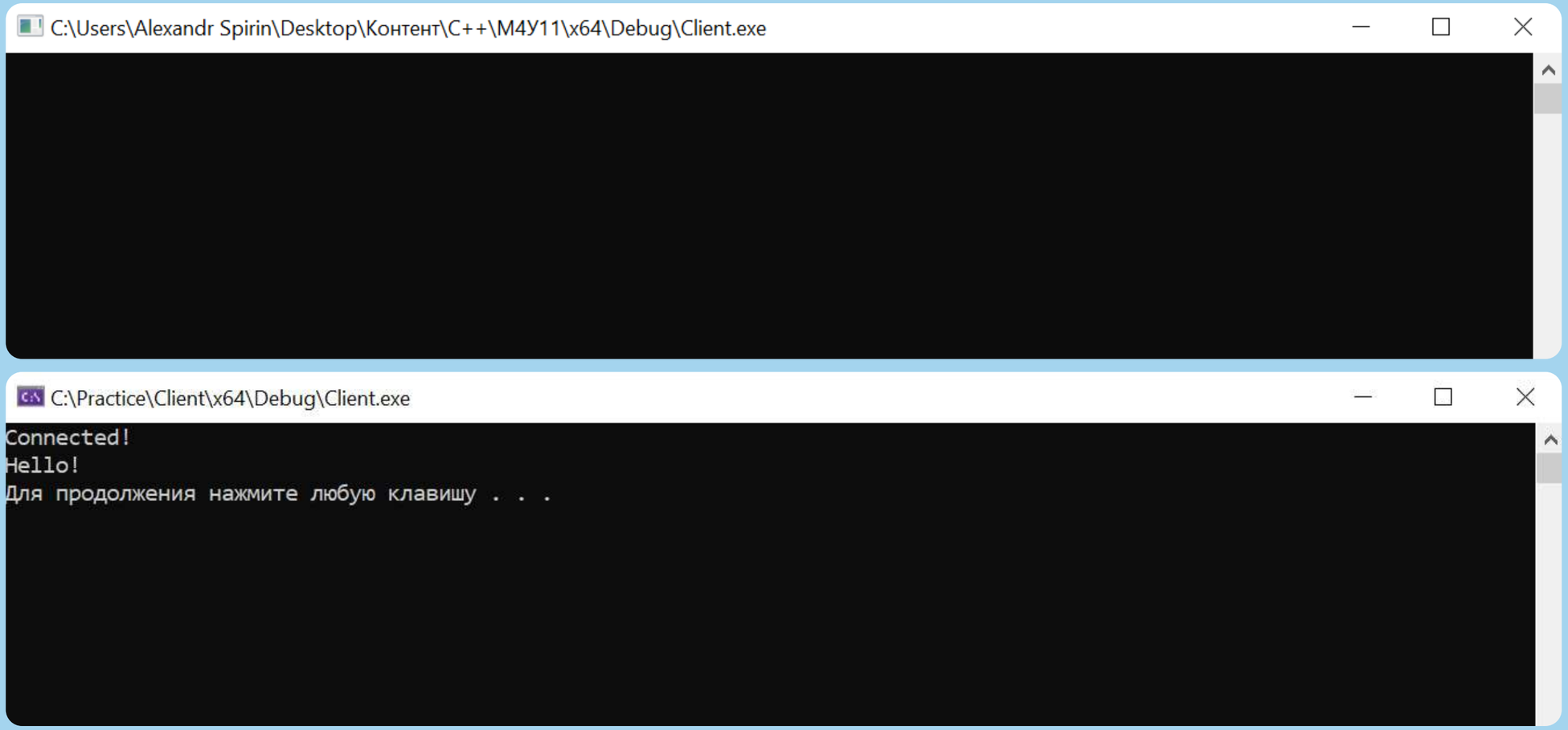
Клиент



Клиент

Запустите Server и несколько клиентов.

Сколько клиентов смогли подключить?



Подключился только один клиент!

Только один! Исправим это.

Сделаем так, чтобы могли подключаться одновременно несколько клиентов.

Сейчас мы устанавливает только одно соединение (создаем один сокет), после чего устанавливаем связь с клиентом, передаёссм сообщение и завершаем программу.

```
SOCKET newConnection;
newConnection = accept(sListen, (SOCKADDR*)&addr, &sizeofaddr);

if (newConnection == 0) {
    std::cout << "Error #2\n";
}
else {
    std::cout << "Client Connected!\n";
    char msg[128] = "Hello!";
    send(newConnection,msg,sizeof(msg),NULL);
}
system("pause");
return 0;
```

создали сокет

установили связь с клиентом

передали сообщение

завершили программу

Создадим массив в котором будем хранить до 100 сокетов.

```
#pragma comment(lib, "ws2_32.lib")
#include <winsock2.h>
#include <iostream>

#pragma warning(disable: 4996)
SOCKET Connections[100];
```

Также создадим переменную, в которой будем хранить индекс соединения.

```
#pragma comment(lib, "ws2_32.lib")
#include <winsock2.h>
#include <iostream>

#pragma warning(disable: 4996)
SOCKET Connections[100];
int count = 0;
```

Будем создавать соединения в цикле.

```
for (int i = 0; i < 100; i++)
{
    SOCKET newConnection;
    newConnection = accept(sListen, (SOCKADDR*)&addr, &sizeofaddr);

    if (newConnection == 0) {
        std::cout << "Error #2\n";
    }
    else {
        std::cout << "Client Connected!\n";
        char msg[128] = "Hello!";
        send(newConnection, msg, sizeof(msg), NULL);
        Connections[count] = newConnection;
        count++;
    }
}
system("pause");
return 0;
```

создаём сокет

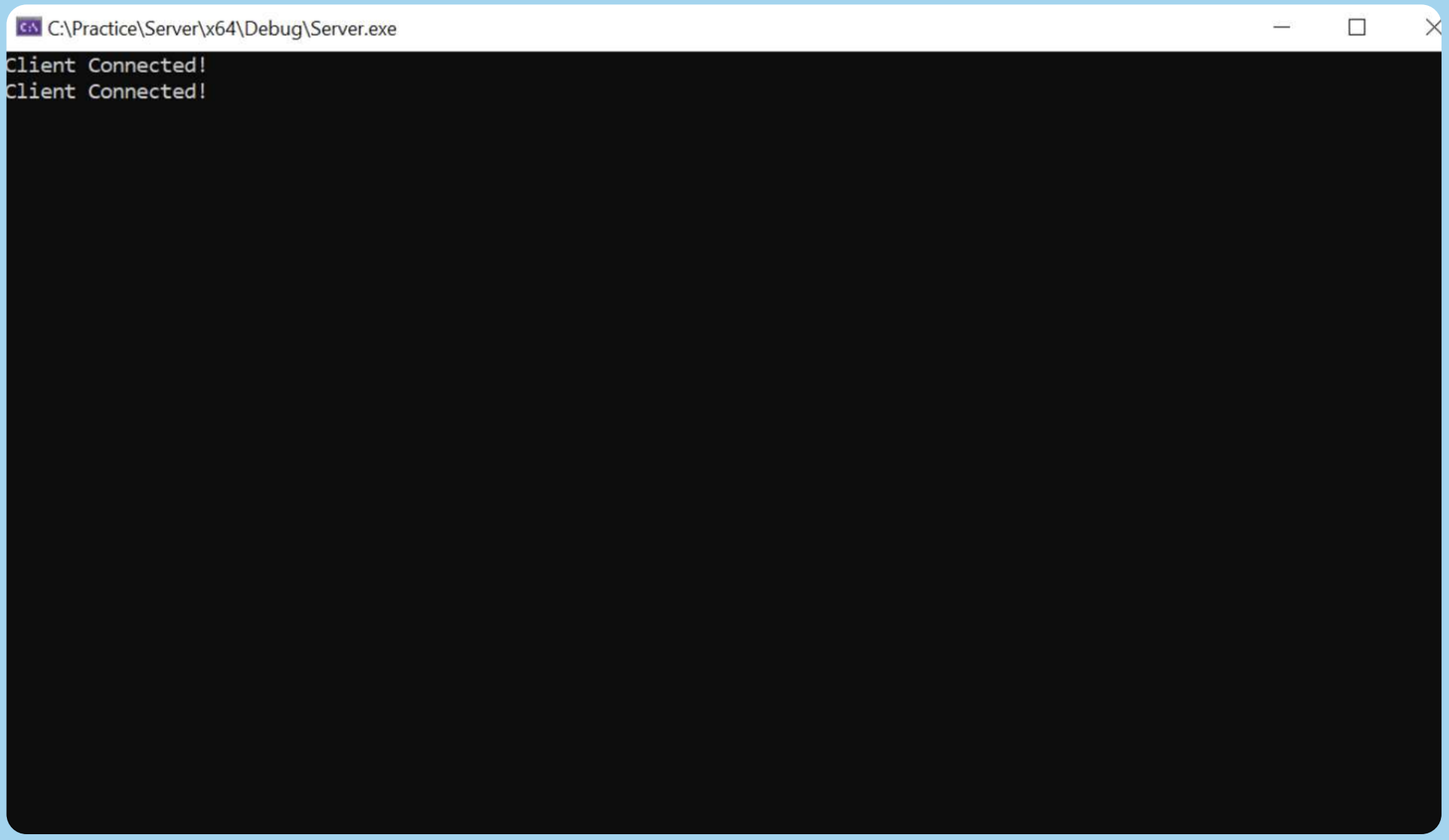
добавляем сокет в массив

увеличиваем счётчик сокетов на 1

Проверьте работу программ.

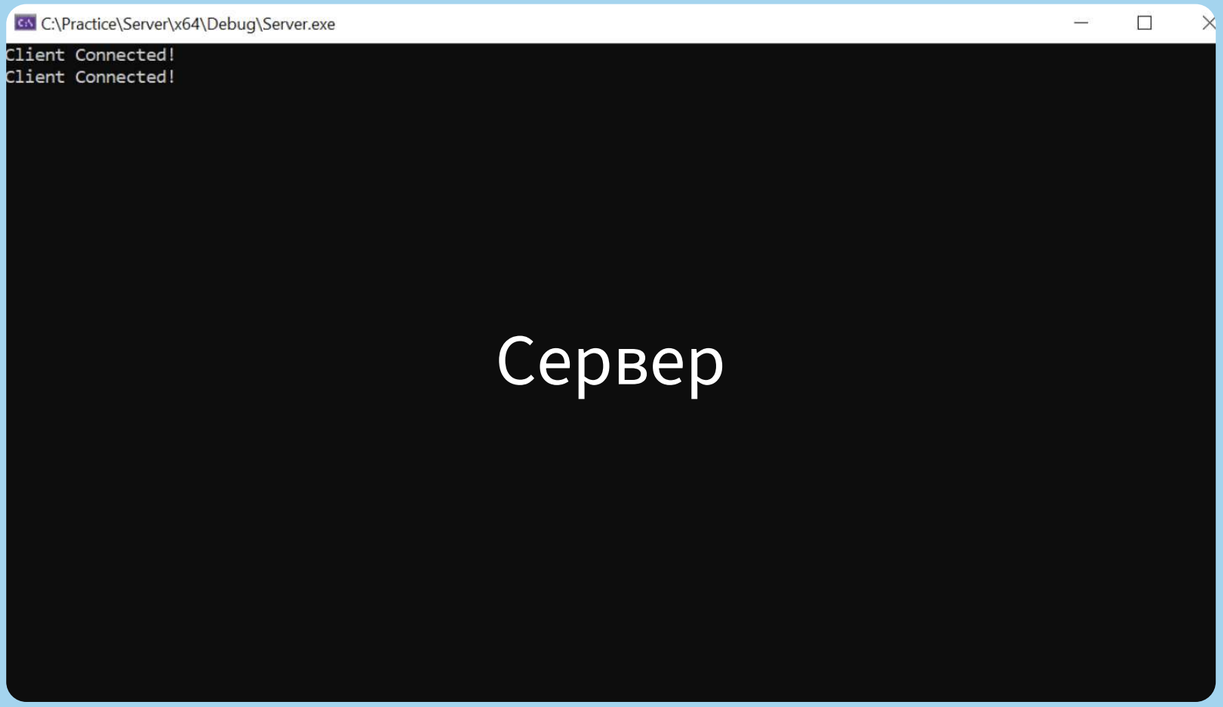
Запустите сервер и несколько клиентов.

Теперь могут подключаться несколько клиентов.



Создадим чат

Клиент



Клиент

Клиент

Доработаем сервер

Создадим функцию, которая будет принимать сообщение и рассылать его всем, кроме автора.

```
void ClientHandler(int index)
{
    char msg[128];
    while (true)
    {
        recv(Connections[index], msg, sizeof(msg), NULL);
        for (int i = 0; i < count; i++)
        {
            if (i == index) continue;
            send(Connections[i], msg, sizeof(msg), NULL);
        }

    }
}
```

- функция будет получать индекс отправителя сообщения
- создадим текстовую переменную
- бесконечный цикл для приёма сообщений
- получение сообщений из сокета с номером index
- перебор всех подключений
- если номер подключения равен номеру отправителя, то пропустить
- отправить полученное сообщение

Потоки

- Когда приложение начинает свою работу, для него создаётся процесс (process). Обычно, каждой программе соответствует один процесс.
- При создании процесса для него выделяется память — виртуальное адресное пространство (virtual address space).
- Каждый процесс имеет как минимум один поток (thread).
- Потоки могут выполняться параллельно, что обеспечивает многопоточность.

Запустим функцию в новом потоке

```
for (int i = 0; i < 100; i++)
{
    SOCKET newConnection;
    newConnection = accept(sListen, (SOCKADDR*)&addr, &sizeofaddr);

    if (newConnection == 0) {
        std::cout << "Error #2\n";
    }
    else {
        std::cout << "Client Connected!\n";
        char msg[128] = "Hello!";
        send(newConnection, msg, sizeof(msg), NULL);
        Connections[count] = newConnection;
        count++;
        CreateThread(NULL, NULL, (LPTHREAD_START_ROUTINE)ClientHandler, (LPVOID)(i), NULL, NULL);
    }
}
system("pause");
return 0;
```

Запустим функцию в новом потоке

```
CreateThread(NULL, NULL, (LPTHREAD_START_ROUTINE)ClientHandler, (LPVOID)(i), NULL, NULL);
```

Запускаемая функция

Параметр функции

Доработаем клиент

Сделаем переменную Connection глобальной.

```
#pragma comment(lib, "ws2_32.lib")
#include <winsock2.h>
#include <iostream>

#pragma warning(disable: 4996)
SOCKET Connection;
```

Создадим функцию для приёма сообщений

Сделаем переменную Connection глобальной.

```
void ClientHandler()
{
    char msg[128];
    while (true)
    {
        recv(Connection, msg, sizeof(msg), NULL);
        std::cout << msg << std::endl;

    }
}
```

Создадим текстовую переменную msg

Бесконечный цикл

Принимаем сообщения в переменную msg

Выводим полученное сообщение в консоль

Запустим функцию в новом потоке

```
CreateThread(NULL, NULL, (LPTHREAD_START_ROUTINE)ClientHandler, NULL, NULL, NULL);
```

Запускаемая функция

Нет передаваемого параметра функции

Ранее реализованный приём сообщений больше не нужен, так как теперь принимаем сообщения в функции.

```
SOCKADDR_IN addr;
int sizeofaddr = sizeof(addr);
addr.sin_addr.s_addr = inet_addr("127.0.0.1");
addr.sin_port = htons(1111);
addr.sin_family = AF_INET;

Connection = socket(AF_INET, SOCK_STREAM, NULL);

if (connect(Connection, (SOCKADDR*)&addr, sizeof(addr)) != 0) {
    std::cout << "Error: failed connect to server.\n";
    return 1;
}
std::cout << "Connected!\n";

char msg[128];
recv(Connection, msg, sizeof(msg),NULL);
std::cout << msg << std::endl;

CreateThread(NULL, NULL, (LPTHREAD_START_ROUTINE)ClientHandler, NULL, NULL, NULL);
char msg1[128];
while (true)
{
    std::cin.getline(msg1, sizeof(msg1));
    send(Connection, msg1, sizeof(msg1), NULL);
    Sleep(10);

}
```

Запустите приложения и проверьте работу.