Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Программирование сетевых приложений

Студент: Немкович А.В.

ФИТ 3 курс 1 группа

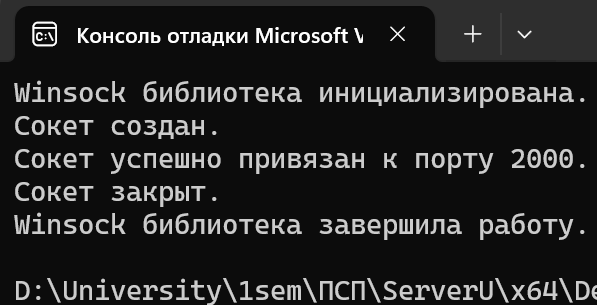
Преподаватель: Некрасова А.П.

Минск 2024

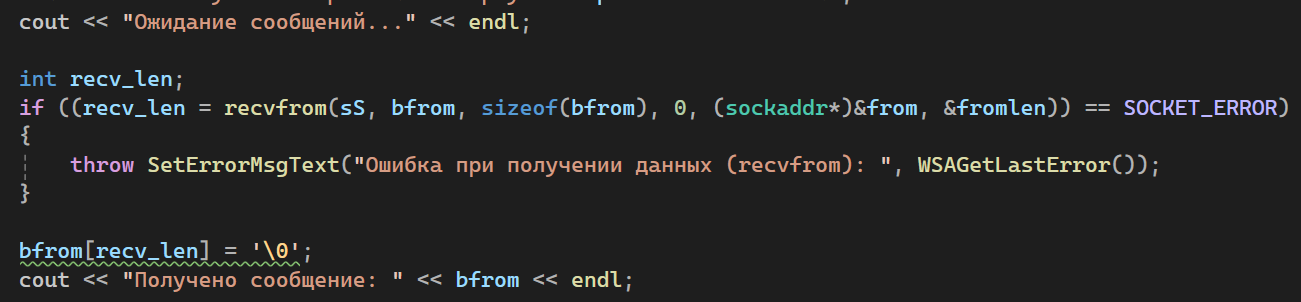
**Лабораторная работа №3**

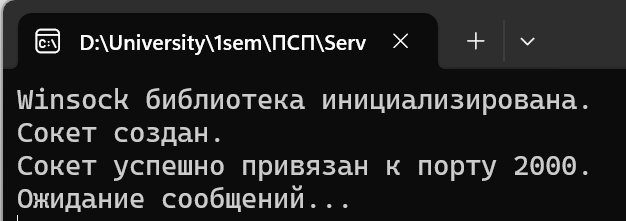
**Задание 1**. Разработайте программу **ServerU**, реализующую блоки 1, 2 и 5 схемы сервера, изображенной на рисунке 3.4.1. Подключите функции обработки ошибок, разработанные в практической работе № 2 (с применением команд структурной обработки ошибок **try-throw-catch**). Обратите внимание 1) на параметр **type** функции **socket**; 2) на отсутствие функций **listen** и **accept**, которые применялись в приложении с соединением. Убедитесь, что разработанная программа выполняет все функции Winsock2 без ошибок.

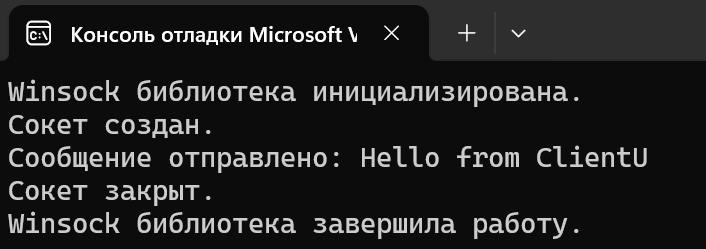


****

**Задание 2**. Реализуйте в программе **ServerU** блок 3 схемы сервера изображенной на рисунке 3.4.1. Используемая в блоке функция **recvfrom**  описана в разделе 3.12 пособия. Установите номер серверного сокета равным **2000**. Убедитесь, что при запуске программа **ServerU**  приостанавливает свое выполнение (переходит в состояние ожидания) сразу после вызова функции **recvfrom**.Завершите программу.

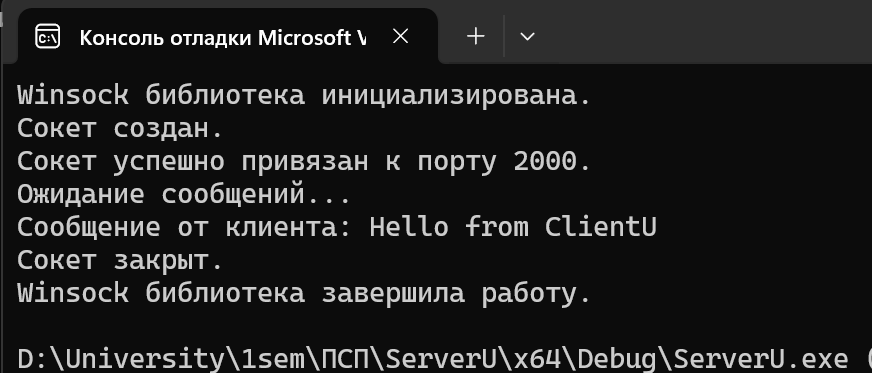




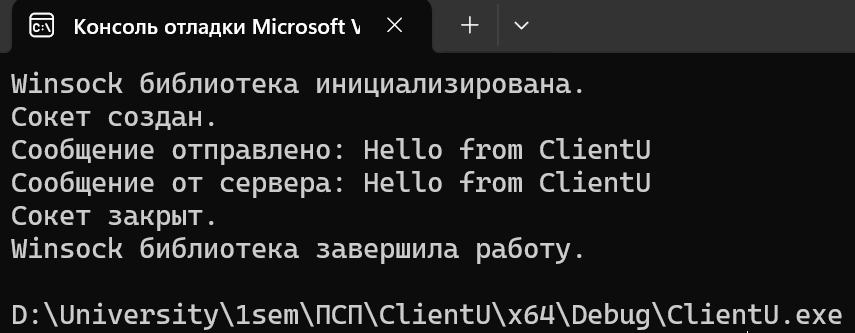
**Задание 3**. Создайте новое С++ -приложение с именем **ClientU**. Реализуйте блоки 1, 2, 3 и 5 схемы клиента, изображенной на рисунке 3.4.1. Подключите функции обработки ошибок, разработанные в практической работе № 2. В параметре **to** команды **sendto** (раздел 3.12), установите адрес структуры **SOCKADDR\_IN**, содержащей IP-адрес равный **127.0.0.1** и номер порта равный **2000**. Обеспечьте пересылку сообщения ***Hello from ClientU***. Запустите на выполнение программу **ClientU** при отсутствующем сервере. Проанализируйте полученный код возврата. 

UDP-протоколе клиент может отправлять данные, даже если сервер не запущен, потому что протокол UDP не требует установки соединения (в отличие от TCP). UDP просто отправляет пакет данных в сеть, не проверяя, есть ли активный получатель. Поэтому в данном случае функция sendto завершилась успешно. Если бы это была программа на основе протокола TCP, возникла бы ошибка, так как TCP требует установления соединения перед отправкой данных.

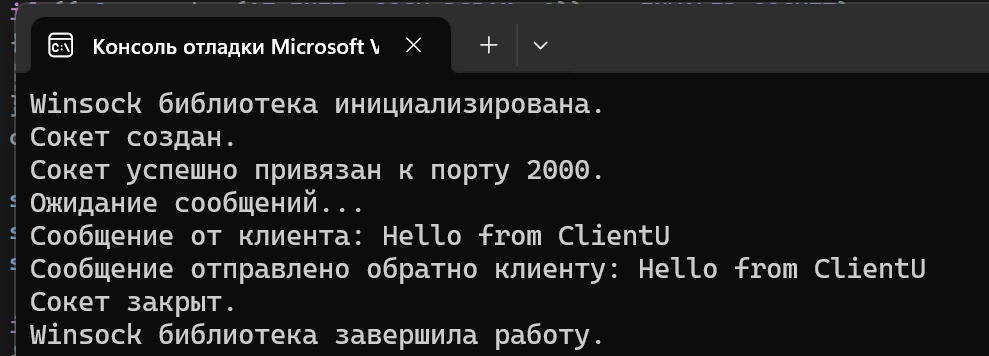
**Задание 4**. Запустите на выполнение программу **ServerU** и убедитесь, что она приостановила свое выполнение. Запустите на этом же компьютере программу **ClientU** и убедитесь, что программы сервера получила сообщение и завершилась нормально.



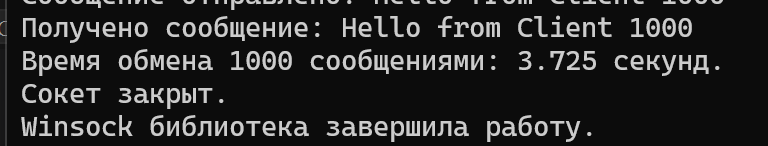
**Задание 5**. Реализуйте блоки 4 в обеих программах. Перешлите полученное сервером сообщение обратно в адрес клиента и убедитесь, что сообщение получено.

****

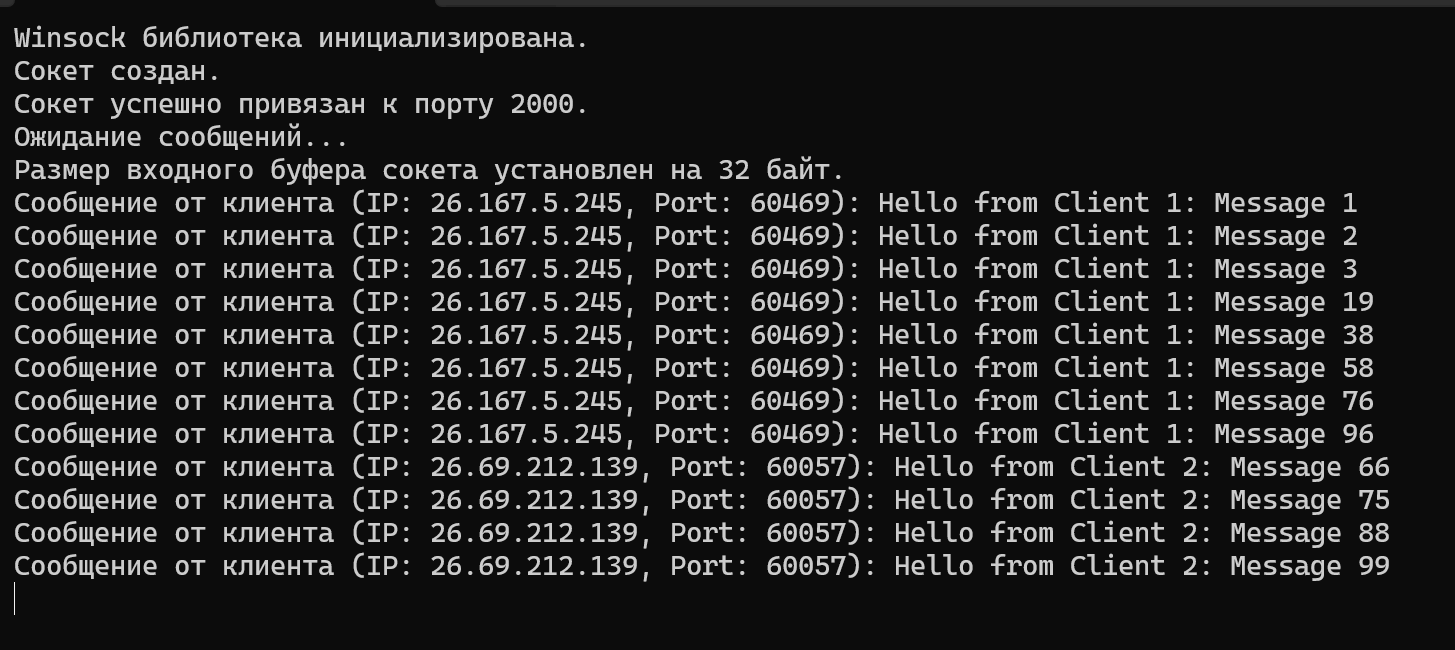
**Задание 6**. Внесите необходимые изменения в программу **ClientU** для того, чтобы программы можно было бы расположить на разных компьютерах локальной сети. Убедитесь в работоспособности приложения.

****

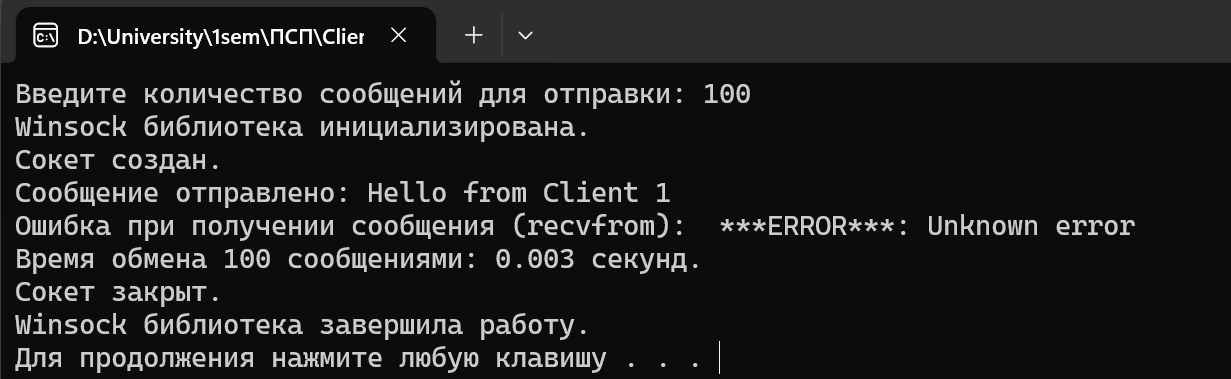
**Задание 7**. Реализуйте последовательную пересылку данных от клиента к серверу и обратно по тому же принципу как это было сделано в заданиях 13, 14 практической работы № 2. Проведите измерения аналогичные оценки скорости передачи, сравните результаты.



**Задание 8.** Запустите сервер **ServerU** на одном из компьютеров и одновременно два клиента на двух других компьютерах локальной сети. Оцените количество сообщений, которые успел передать и получить каждый из клиентов.

****

**Задание 9.** Запустите сервер **ServerT** (разработанный в практической работе № 2) и программу клиента **ClientU**. Объясните полученный результат.

****

**Задание 10.** Запустите сервер **ServerU** и клиент **ClientТ**(разработанный в практической работе № 2). Объясните полученный результат.

