**Лабораторная работа 3 "Продвинутый сервер" (сервер может одновременно обслуживать несколько клиентов).**

**Схема изменений** сервера из Лабораторной работы 2 (клиент без изменений)

1. При реализации сервера код обработки клиента вынести в отдельную функцию.

Сначала это будет функция типа void ClientFunc(SOCKET Conn);

В самой функции идет обмен сообщениями клиента и сервера до получения кодового слова и закрытие сокета. Т.е. пока просто часть кода вынесли в функцию. Проверили работоспособность сервера.

while(1)

{

Conn = accept(SrvSock, (struct sockaddr \*) &ConnAddr, &AddrLen);

ClientFunc(Conn);

}

2. Потом мы вызовем эту функцию ClientFunc(Conn) в отдельном потоке для каждого подключившегося клиента.

Для этого надо изменить описание функции ClientFunc

#include "stdafx.h"

#include <windows.h>

#include <stdio.h>

DWORD WINAPI ClientFunc ( LPVOID lpParam ) // функция, которую потом будем запускать в отдельном потоке

{

SOCKET Conn = \*(SOCKET\*)lpParam; // привели параметр к нужному типу

// цикл обмена сообщениями

}

// Изменения в main

int main()

{

DWORD ThreadId;

INT ThreadParameter = 19;

HANDLE hThread;

// пропущены все шаги создания сокета.

// изменения только в одном цикле

while(1)

{

Conn = accept(SrvSock, (struct sockaddr \*) &ConnAddr, &AddrLen);

// создаем поток с функцией вместо прямого вызова функции

hThread = CreateThread(

NULL, // атрибуты безопасности по умолчанию

0, // размер стека по умолчанию

ClientFunc , // указатель на функцию создаваемого потока

&Conn, // аргумент, передаваемый функции потока

0, // флаги создания по умолчанию

&ThreadId); // возвращаемый идентификатор потока

if (hThread == NULL) printf("CreateThread failed." );

getchar();

CloseHandle( hThread );

return 0;

}

Предоставьте  файлы исходного кода  с описанием работы сервера и клиента.

Предоставьте скриншот работы приложения, использующего сокеты.

Должно быть видно окна клиентов и сервера на одном рисунке.