Министерство образования Республики Беларусь

Учреждение образования «Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 5

За 6 семестр

По дисциплине: «Аппаратно-программное обеспечение ЭВМ и сетей»

Тема: «Изучение протокола UDP»

Выполнила:

студентка 3-го курса

группы АС-56

Карпенко М.В.

Проверил:

Булей Е. В.

Брест 2022

Цель работы: 1) изучить основы программирования сетевых приложений Windows на базе библиотеки WINSOCK2.H; 2) приобрести навыки по практическому использованию библиотеки для реализации сетевых приложений в среде C++ на базе протоколов TCP и UDP.

Вариант 4

Вариант задания для бригады студентов	**1**	**2**	**3**	**4**
Номер задания для реализации TCP (UDP) сервера (см. табл. 2).	1	2	3	4
Номер задания для реализации TCP (UDP) клиента (см. табл. 3).	8	7	6	5

5	После приема каждой цепочки символов, заканчивающейся символом «.», сервер отсылает предложение, в котором слова (цепочки символов, отделенные пробелом) расположены в обратном порядке. Если встречается последовательно два символа «..», то сервер отсылает сообщение об окончании сеанса и разрывает соединение.			
5	Home	1), 3)	+	1)

Код:

СЕРВЕР:

```
#define _CRT_SECURE_NO_WARNINGS
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#pragma comment(lib, "Ws2_32.lib")

#include <winsock2.h>
#include <stdio.h>
#include <time.h>
#include <string>
#include <vector>

#define BUFLEN 512	//Max length of buffer
#define PORT 8888	//The port on which to listen for incoming data

int main()
{
	SYSTEMTIME lt;
	SOCKET s;
	struct sockaddr_in server, si_other;
	int slen, recv_len;
	char buf[BUFLEN];
	WSADATA wsa;

	slen = sizeof(si_other);

	//Initialise winsock
	printf("\nInitialising Winsock...");
	if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
	{
		printf("Failed. Error Code : %d", WSAGetLastError());
		exit(EXIT_FAILURE);
	}
	printf("Initialised.\n");

	//Create a socket
	if ((s = socket(AF_INET, SOCK_DGRAM, 0)) == INVALID_SOCKET)
	{
		printf("Could not create socket : %d", WSAGetLastError());
	}
	printf("Socket created.\n");
```

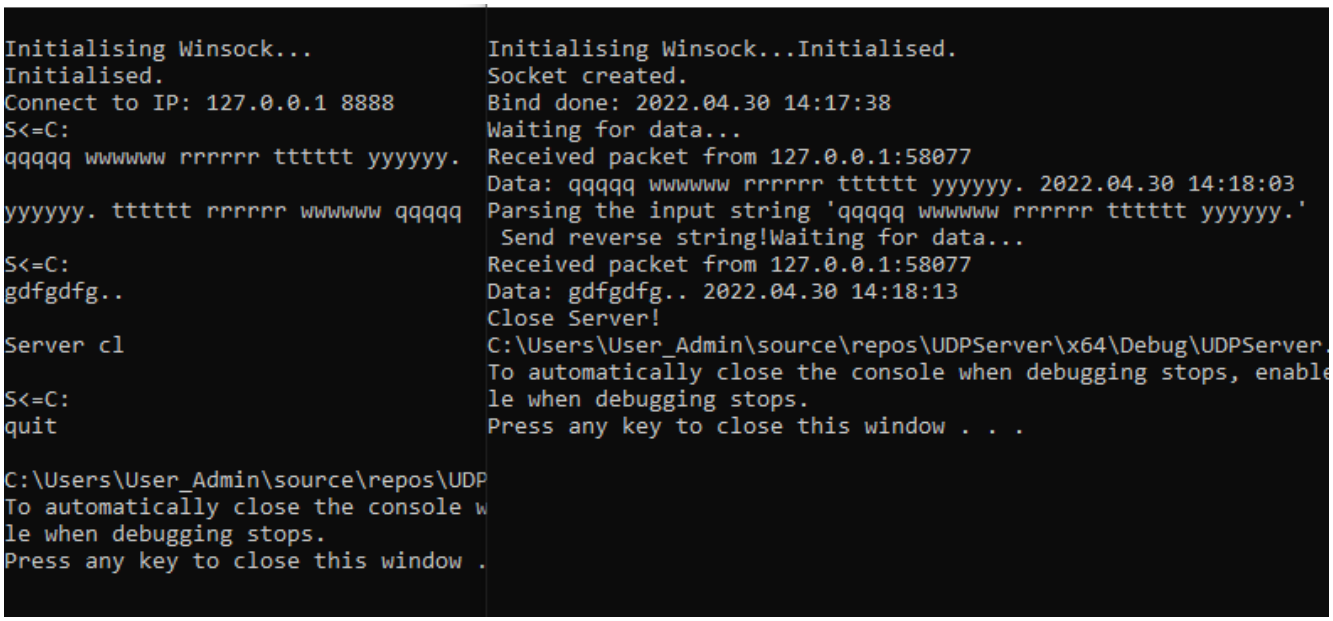
```
	//Prepare the sockaddr_in structure
	server.sin_family = AF_INET;
	server.sin_addr.s_addr = INADDR_ANY;
	server.sin_port = htons(PORT);

	//Time
	GetLocalTime(&lt);

	//Bind
	if (bind(s, (struct sockaddr*)&server, sizeof(server)) == SOCKET_ERROR)
	{
		printf("Bind failed with error code : %d", WSAGetLastError());
		exit(EXIT_FAILURE);
	}
	printf("Bind done: %02d.%02d.%02d %02d:%02d:%02d\n", lt.wYear, lt.wMonth, lt.wDay, lt.wHour, lt.wMinute,
lt.wSecond);

	//keep listening for data
	while (1)
	{
		printf("Waiting for data...\n");
		fflush(stdout);

		//clear the buffer by filling null, it might have previously received data
		memset(buf, '\0', BUFLEN);

		//try to receive some data, this is a blocking call
		if ((recv_len = recvfrom(s, buf, BUFLEN, 0, (struct sockaddr*)&si_other, &slen)) == SOCKET_ERROR)
		{
			printf("recvfrom() failed with error code : %d", WSAGetLastError());
			exit(EXIT_FAILURE);
		}


		GetLocalTime(&lt);
		//print details of the client/peer and the data received
		printf("Received packet from %s:%d\n", inet_ntoa(si_other.sin_addr), ntohs(si_other.sin_port));
		printf("Data: %s %02d.%02d.%02d %02d:%02d:%02d\n", buf, lt.wYear, lt.wMonth, lt.wDay, lt.wHour,
lt.wMinute, lt.wSecond);

		//now reply the client with the same data
		std::string outMessage;
		if (buf[strlen(buf) - 1] == '.')
		{
			if (buf[strlen(buf) - 2] == '.')
			{
				printf("Close Server!");
				if (sendto(s, "Server closed!", 15, 0, (struct sockaddr*)&si_other, slen) ==
SOCKET_ERROR)
				{
					printf("sendto() failed with error code : %d\n", WSAGetLastError());
					exit(EXIT_FAILURE);
				}
				break;
			}
			std::vector<std::string> vect1;
			rsize_t strmax = sizeof buf;
			const char* delim = " ";
			char* next_token;
			char* token;
			printf("Parsing the input string '%s'\n Send reverse string!", buf);
			token = strtok_s(buf, delim, &next_token);
			while (token) {
				vect1.push_back(std::string(token));
				token = strtok_s(NULL, delim, &next_token);
			}
```

```
                        for (auto it = vect1.crbegin(); it != vect1.crend(); ++it)
                        {
                                outMessage += *it;
                                outMessage += " ";
                        }
                }
                if (sendto(s, outMessage.c_str(), strlen(outMessage.c_str()), 0, (struct sockaddr*)&si_other, slen) ==
SOCKET_ERROR)
                {
                        printf("sendto() failed with error code : %d\n", WSAGetLastError());
                        exit(EXIT_FAILURE);
                }
        }

        closesocket(s);
        WSACleanup();

        return 0;
    }
```

КЛИЕНТ:

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#pragma comment(lib, "Ws2_32.lib")

#include<conio.h>
#include<stdio.h>
#include<winsock2.h>
#include<iostream>
#include<string>

#define SERVER "127.0.0.1"          //ip address of udp server
#define BUFLEN 512          //Max length of buffer
#define PORT 8888           //The port on which to listen for incoming data
using namespace std;

int main(void)
{
        struct sockaddr_in si_other;
        int s, slen = sizeof(si_other);
        char buf[BUFLEN];
        char message[BUFLEN];
        string IP;

        WSADATA wsa;

        //Initialise winsock
        printf("\nInitialising Winsock...\n");
        if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
        {
                printf("Failed. Error Code : %d", WSAGetLastError());
                exit(EXIT_FAILURE);
        }
        printf("Initialised.\n");

        //create socket
        if ((s = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == SOCKET_ERROR)
        {
                printf("socket() failed with error code : %d", WSAGetLastError());
                exit(EXIT_FAILURE);
        }
        printf("Connect to IP: ");
        string Ip;
        int port;
        cin >> Ip >> port;
        //setup address structure
```

```
	memset((char*)&si_other, 0, sizeof(si_other));
	si_other.sin_family = AF_INET;
	si_other.sin_port = htons(port);
	si_other.sin_addr.S_un.S_addr = inet_addr(Ip.c_str());



	//start communication
	while (1)
	{
		printf("S<=C: \n");
		string messageStr;
		messageStr.clear();
		while (1)
		{
			int ch = _getch();
			if (ch == 224) break;
			if (ch == 13) messageStr +=static_cast<char>(10);
			messageStr += static_cast<char>(ch);
			if (ch == 13) printf("%c", 10);
			printf("%c",messageStr.back());
		}
		_getch();
		printf("\n");
		//fgets(&message[0], sizeof(message) - 1, stdin);
		//if (!strcmp(&message[0], "quit\n")) break;
		if (!strcmp(messageStr.c_str(), "quit")) break;


		//send the message
		if (sendto(s, messageStr.c_str(), strlen(messageStr.c_str()), 0, (struct sockaddr*)&si_other, slen) ==
SOCKET_ERROR)
		{
			printf("\nsendto() failed with error code : %d", WSAGetLastError());
			exit(EXIT_FAILURE);
		}

		//receive a reply and print it
		//clear the buffer by filling null, it might have previously received data
		memset(buf, '\0', BUFLEN);
		//try to receive some data, this is a blocking call
		if (recvfrom(s, buf, BUFLEN, 0, (struct sockaddr*)&si_other, &slen) == SOCKET_ERROR)
		{
			printf("recvfrom() failed with error code : %d", WSAGetLastError());
			exit(EXIT_FAILURE);
		}
		printf("\n");
		puts(buf);
		printf("\n");
	}

	closesocket(s);
	WSACleanup();

	return 0;
}
```



Вывод: 1) изучили основы программирования сетевых приложений Windows на базе библиотеки WINSOCK2.H; 2) приобрели навыки по практическому использованию библиотеки для реализации сетевых приложений в среде C++ на базе протоколов TCP и UDP.