Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ

Звіт

з лабораторної роботи № 5

з дисципліни «Системне програмне забезпечення»

на тему: «Робота з мережевим вводом-виводом, сокети»

Виконав: ст. гр. КІ-302

Радевич-Винницький Я.А.

Перевірила: викладач

Ногаль М.В.

Мета роботи: Навчитися розробляти програми керування мережними засобами операційних систем Навчитися розробляти програми керування мережними засобами операційних систем.

Варіант: 18.

Завдання:

Розробити дві програми, яка демонструють роботу використання механізму сокетів. Одна програма виконує роль сервера, інша – роль клієнта. Індивідуальні завдання такі ж, як до лабораторної роботи №1. Клієнт запитує у користувача вхідні дані і передає їх серверу. Сервер виконує основні обчислення і результат передає клієнту. Клієнт виводить результат на екран. Розробити дві програми, яка демонструють роботу використання механізму сокетів. Одна програма виконує роль сервера, інша – роль клієнта. Індивідуальні завдання такі ж, як до лабораторної роботи №1. Клієнт запитує у користувача вхідні дані і передає їх серверу. Сервер виконує основні обчислення і результат передає клієнту. Клієнт виводить результат на екран.

Виконання завдання:

1. Створено програмний проект сервера.

Код файлу main.cpp:

Лістинг 1

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

#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>
#include <windows.h>

int calculateSumOfOffNumbersInRange(int L, int U, int array[]) {
    int sum = 0;
    for (int i = L; i <= U; i++) {
        if (array[i] % 2 != 0) {
            sum += array[i];
        }
    }
    return sum;
};

int main(void)
{
    printf("TCP Server\n");
    printf("__________\n\n");

    WSADATA wsaData;
    SOCKET ListeningSocket;
    SOCKET NewConnection;
    struct sockaddr_in ServerAddr;
    struct sockaddr_in ClientAddr;
```

```
    int ClientAddrLen;
    u_short Port = 5150;
    int Ret;
    int L, U, size;
    int sum = 0;
    int* array;

    if ((Ret = WSAStartup(MAKEWORD(2, 2), &wsaData)) != 0)
    {
        printf("WSAStartup error, number of error: %d\n", Ret);
        return -1;
    }

    if ((ListeningSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) ==
        INVALID_SOCKET)
    {
        printf("Socket error, number of error: %d\n", WSAGetLastError());
        WSACleanup();
        return -2;
    }

    ServerAddr.sin_family = AF_INET;
    ServerAddr.sin_port = htons(Port);
    ServerAddr.sin_addr.s_addr = htonl(INADDR_ANY);

    if (bind(ListeningSocket, (struct sockaddr*)&ServerAddr, sizeof(ServerAddr)) ==
        SOCKET_ERROR)
    {
        printf("Bind error, number of error: %d\n", WSAGetLastError());
        closesocket(ListeningSocket);
        WSACleanup();
        return -3;
    }

    if (listen(ListeningSocket, 5) == SOCKET_ERROR)
    {
        printf("Listen error, number of error: %d\n", WSAGetLastError());
        closesocket(ListeningSocket);
        WSACleanup();
        return -4;
    }

    printf("Waiting for connection on port: %d.\n", Port);
    ClientAddrLen = sizeof(ClientAddr);

    if ((NewConnection = accept(ListeningSocket, (struct sockaddr*)&ClientAddr,
        &ClientAddrLen)) == INVALID_SOCKET)
    {
        printf("Accept error, number of error: %d\n", WSAGetLastError());
        closesocket(ListeningSocket);
        WSACleanup();
        return -5;
    }

    printf("Connection success with %s:%d.\n", inet_ntoa(ClientAddr.sin_addr),
        ntohs(ClientAddr.sin_port));
    closesocket(ListeningSocket);

    if ((Ret = recv(NewConnection, (char*)&L, sizeof(int), 0)) == SOCKET_ERROR)
    {
        printf("Recv error, number of error: %d\n", WSAGetLastError());
        closesocket(NewConnection);
        WSACleanup();
        return -6;
    }
```

```
    if ((Ret = recv(NewConnection, (char*)&U, sizeof(int), 0)) == SOCKET_ERROR)
    {
        printf("Recv error, number of error: %d\n", WSAGetLastError());
        closesocket(NewConnection);
        WSACleanup();
        return -7;
    }

    if ((Ret = recv(NewConnection, (char*)&size, sizeof(int), 0)) == SOCKET_ERROR)
    {
        printf("Recv error, number of error: %d\n", WSAGetLastError());
        closesocket(NewConnection);
        WSACleanup();
        return -8;
    }

    array = (int*)malloc(size * sizeof(int));

    if ((Ret = recv(NewConnection, (char*)array, sizeof(int) * size, 0)) ==
SOCKET_ERROR)
    {
        printf("Recv error, number of error: %d\n", WSAGetLastError());
        closesocket(NewConnection);
        WSACleanup();
        return -9;
    }
    printf("Array: \n");
    for (int i = 0; i < size; i++) {
        printf("%d\n", array[i]);
    }

    printf("\n");

    sum = calculateSumOfOffNumbersInRange(L, U, array);

    printf("Sum of odd numbers in the range [%d, %d] is: %d\n", L, U, sum);

    if ((Ret = send(NewConnection, (char*)&sum, sizeof(int), 0)) == SOCKET_ERROR)
    {
        printf("Send error, number of error: %d\n", WSAGetLastError());
        closesocket(NewConnection);
        WSACleanup();
        return -10;
    }

    printf("Result sent to client.\n");
    printf("Closing connection with client.\n");
    closesocket(NewConnection);
    WSACleanup();
    printf("Press Enter to finish.\n");
    getchar();
    return 0;
}
```

2. Створено програмний проект клієнта.

Код файлу main.cpp:

Лістинг 2

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

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <winsock2.h>
#include <windows.h>

int main()
{
    printf("TCP Client\n");
    printf("__________\n\n");

    WSADATA wsaData;
    SOCKET s;
    struct sockaddr_in ServerAddr;
    u_short Port = 5150;
    int Ret;
    int L, U;
    int* array;
    int size;
    int sum = 0;

    if ((Ret = WSAStartup(MAKEWORD(2, 2), &wsaData)) != 0)
    {
        printf("WSAStartup error, number of error: %d\n", Ret);
        return 1;
    }

    s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (s == INVALID_SOCKET)
    {
        printf("Socket error, number of error: %d\n", WSAGetLastError());
        WSACleanup();
        return 2;
    }

    char ipAddress[32];
    printf("Enter server IP-address : ");
    gets_s(ipAddress);

    printf("Enter the lower bound (L) of the range: ");
    scanf_s("%d", &L);
    printf("Enter the upper bound (U) of the range: ");
    scanf_s("%d", &U);

    printf("Enter array size: ");
    scanf_s("%d", &size);

    array = (int*)malloc(size * sizeof(int));

    srand(time(NULL));
    printf("Generated array: \n");
    for (int i = 0; i < size; i++)
    {
        array[i] = rand() % 50;
        printf("%d\n", array[i]);
    }
```

```
    printf("\n");

    ServerAddr.sin_family = AF_INET;
    ServerAddr.sin_port = htons(Port);
    ServerAddr.sin_addr.s_addr = inet_addr(ipAddress);

    printf("Connection attempt with %s:%d ...\n", inet_ntoa(ServerAddr.sin_addr),
        ntohs(ServerAddr.sin_port));

    if (connect(s, (struct sockaddr*)&ServerAddr, sizeof(ServerAddr)) ==
        SOCKET_ERROR)
    {
        printf("Connect error, number of error: %d\n", WSAGetLastError());
        closesocket(s);
        WSACleanup();
        return 3;
    }

    printf("Connection success.\n");

    if ((Ret = send(s, (char*)&L, sizeof(int), 0)) == SOCKET_ERROR)
    {
        printf("Send error, number of error: %d\n", WSAGetLastError());
        closesocket(s);
        WSACleanup();
        return 4;
    }

    if ((Ret = send(s, (char*)&U, sizeof(int), 0)) == SOCKET_ERROR)
    {
        printf("Send error, number of error: %d\n", WSAGetLastError());
        closesocket(s);
        WSACleanup();
        return 5;
    }


    if ((Ret = send(s, (char*)&size, sizeof(int), 0)) == SOCKET_ERROR)
    {
        printf("Send error, number of error: %d\n", WSAGetLastError());
        closesocket(s);
        WSACleanup();
        return 6;
    }

    if ((Ret = send(s, (char*)array, sizeof(int) * size, 0)) == SOCKET_ERROR)
    {
        printf("Send error, number of error: %d\n", WSAGetLastError());
        closesocket(s);
        WSACleanup();
        return 7;
    }



    printf("Data sent to server.\n");

    printf("Data received from server:\n");

    if ((Ret = recv(s, (char*)&sum, sizeof(int), 0)) == SOCKET_ERROR)
    {
        printf("Recv error, number of error: %d\n", WSAGetLastError());
        closesocket(s);
        WSACleanup();
        return -8;
```

```
    }

    printf("Sum of odd numbers in the range [%d, %d] is: %d\n", L, U, sum);

    closesocket(s);
    WSACleanup();
    printf("Press ENTER to finish.\n");
    getchar();
    return 0;
}
```

Клієнтська програма запитує користувача розмір масиву, генерує його заповнюючи випадковими числами та ініціює з'єднання з сервером, надсилає йому проініціалізований масив та отримує результат обчислення. Серверна програма слухає з'єднання, отримує дані та проводить обчислення задане варіантом завдання.

Результат роботи програм:

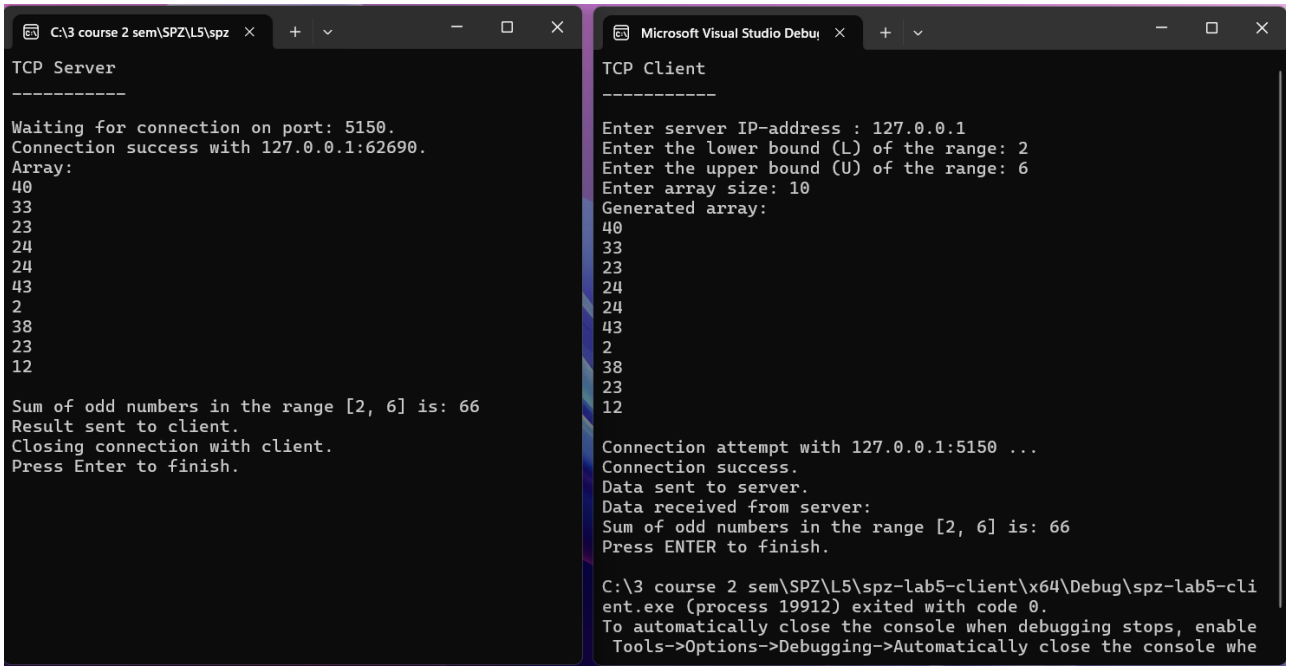

Рис. 1 – результат роботи програм

Висновок: під час виконання лабораторної роботи було набуто навичок розробки програм керування мережними засобами операційних систем. Було створено клієнтську програму яка ініціює з’єднання з сервером та надсилає йому вхідні дані, а також серверну програму, яка отримує вхідні дані, проводить обчислення та надсилає результат на клієнтську програму.