**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

по дисциплине «Программирование сетевых приложений»

на тему: «**Разработка клиент­серверных приложений на базе**

**стека протоколов TCP/IP в .Net.**»

Выполнил: студент гр. ИТИ-41

Гончар А.С.

Принял: ассистент

Карабчикова Е.А.

Гомель 2020

**Цель работы**: изучение пуда потоков OC *Windows.*

Задание для 5 варианта: Разработать конечный автомат для анализа надежности и корректности пароля

Длина пароля не менее 6 символов. Разрешены только латинские буквы, цифры и знаки: пробел \_ ­ . В пароле обязательно должны присутствовать знаки в верхнем и нижнем регистре и цифры.

Пользователю возвращается информация о том, можно ли использовать указанный пароль

**Результат выполнения:**

Разработать сервер с использованием пула потоков для параллельной обработки запросов клиентов. К серверу подключается много клиентов, все они отправляют задачи (одномерная оптимизация функции), сервер обрабатывает каждый запрос в отдельном потоке (используя пул потоков) и возвращает ответ клиенту.

Выполнение.

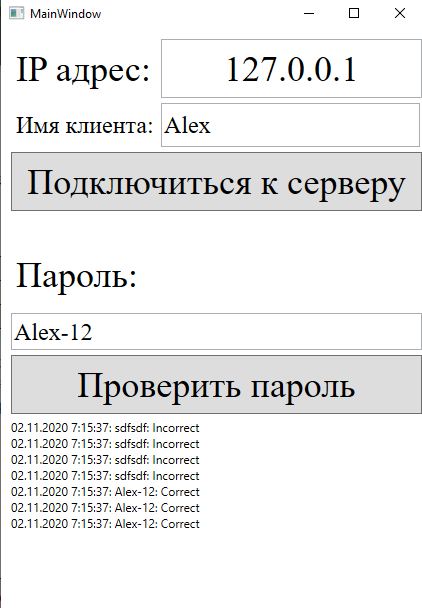


Рисунок 1 – Результат выполнение работы клиента

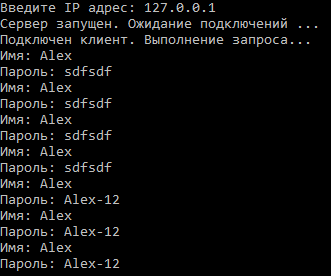


Рисунок 2 – Результат выполнение работы сервера

**Вывод**: в ходе выполнения лабораторной работы разработано серверное приложение нахождения площади фигуры с использованием сокетов *TCP*/*IP*.

**ПРИЛОЖЕНИЕ А**

Листинг программы

Файл ***ServerApp.cpp***:

// Пример простого TCP-эхо-сервера

#include <stdio.h>

#pragma comment(lib, "ws2\_32.lib")

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#include <winsock2.h> // Wincosk2.h должен быть раньше windows!

#include <windows.h>

#include <locale.h>

#include <iostream>

#include <regex>

using namespace std;

LPTHREAD\_START\_ROUTINE WINAPI NewClient(PVOID\* argt);

#define MY\_PORT 666 // Порт, который слушает сервер 666

const unsigned int MAX\_BUF\_LENGTH = 4096;

char buff[1024];

int main()

{

cout << "SERVER APPLICATION \n";

// Шаг 1 - Инициализация Библиотеки Сокетов

if (WSAStartup(0x0202, (WSADATA\*)&buff[0]))

{

// Ошибка!

printf("Error WSAStartup %d\n", WSAGetLastError());

return -1;

}

// Шаг 2 - создание сокета

SOCKET mysocket;

// AF\_INET - сокет Интернета

// SOCK\_STREAM - потоковый сокет (с установкой соединения)

// 0 - по умолчанию выбирается TCP протокол

if ((mysocket = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

// Ошибка!

printf("Error socket %d\n", WSAGetLastError());

WSACleanup(); // Деиницилизация библиотеки Winsock

return -1;

}

// Шаг 3 - связывание сокета с локальным адресом

sockaddr\_in local\_addr;

local\_addr.sin\_family = AF\_INET;

local\_addr.sin\_port = htons(MY\_PORT); // не забываем о сетевом порядке!!!

local\_addr.sin\_addr.s\_addr = 0; // сервер принимает подключения

// на все свои IP-адреса

// вызываем bind для связывания

if (bind(mysocket, (sockaddr\*)&local\_addr, sizeof(local\_addr)))

{

// Ошибка

printf("Error bind %d\n", WSAGetLastError());

closesocket(mysocket); // закрываем сокет!

WSACleanup();

return -1;

}

// Шаг 4 - ожидание подключений

// размер очереди - 0x100

if (listen(mysocket, 0x100))

{

// Ошибка

printf("Error listen %d\n", WSAGetLastError());

closesocket(mysocket);

WSACleanup();

return -1;

}

cout << "SERVER IS READY!\n";

cout << "Waiting for connections...\n";

// Шаг 5 - извлекаем сообщение из очереди

SOCKET client\_socket; // сокет для клиента

sockaddr\_in client\_addr; // адрес клиента (заполняется системой)

// функции accept необходимо передать размер структуры

int client\_addr\_size = sizeof(client\_addr);

// цикл извлечения запросов на подключение из очереди

while ((client\_socket = accept(mysocket, (sockaddr\*)&client\_addr, \

& client\_addr\_size)))

{

// пытаемся получить имя хоста

HOSTENT\* hst;

hst = gethostbyaddr((char\*)&client\_addr.sin\_addr.s\_addr, 4, AF\_INET);

// вывод сведений о клиенте

printf("+%s [%s] new connect!\n",

(hst) ? hst->h\_name : "", inet\_ntoa(client\_addr.sin\_addr));

QueueUserWorkItem((LPTHREAD\_START\_ROUTINE)NewClient, &client\_socket, WT\_EXECUTEDEFAULT);

}

return 0;

}

LPTHREAD\_START\_ROUTINE WINAPI NewClient(PVOID\* argt)

{

SOCKET client\_socket = ((SOCKET\*)argt)[0];

char message[MAX\_BUF\_LENGTH];

regex regextext("^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?[0-9])(?=.\*?[-\_]).{6,}$");

regex regexkir("[а-яА-Я]");

regex regexzn("[/\*+=!@#$%^,.<>;:()&|\`~{}]");

while (strcmp(message, "quit") != 0) {

bool correct = false;

int bytes\_recv;

bytes\_recv = recv(client\_socket, &message[0], MAX\_BUF\_LENGTH, 0);

cout << message << endl;

if (regex\_match(message, regextext)) {

if (!regex\_match(message, regexkir) && !regex\_match(message, regexzn)) {

correct = true;

}

}

if (correct) {

strcpy\_s(message, "Valid");

}

else {

strcpy\_s(message, "Invalid");

}

send(client\_socket, &message[0], MAX\_BUF\_LENGTH, 0);

}

printf("-disconnect\n");

// закрываем сокет

closesocket(client\_socket);

return 0;

}

Файл ***ClientApp.cpp***:

// Пример простого TCP-клиента

#include <stdio.h>

#include <string.h>

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

#pragma comment(lib, "ws2\_32.lib")

#include <winsock2.h>

#include <windows.h>

#include <iostream>

#include <string>

using namespace std;

#define PORT 666

const unsigned int MAX\_BUF\_LENGTH = 4096;

char SERVERADDR[11];

int main(int argc, char\* argv[])

{

char buff[1024];

printf("CLIENT APPLICATION\n");

int choose;

cout << "Choose:\n1.Local IP\n2.Enter IP\n";

cin >> choose;

switch (choose)

{

case 1:

strcpy\_s(SERVERADDR, "127.0.0.1");

break;

case 2:

cout << "Enter server's ip address:";

cin >> SERVERADDR;

break;

}

// Шаг 1 - инициализация библиотеки Winsock

if (WSAStartup(0x202, (WSADATA\*)&buff[0]))

{

printf("WSAStart error %d\n", WSAGetLastError());

return -1;

}

// Шаг 2 - создание сокета

SOCKET my\_sock;

my\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

if (my\_sock < 0)

{

printf("Socket() error %d\n", WSAGetLastError());

return -1;

}

// Шаг 3 - установка соединения

// заполнение структуры sockaddr\_in - указание адреса и порта сервера

sockaddr\_in dest\_addr;

dest\_addr.sin\_family = AF\_INET;

dest\_addr.sin\_port = htons(PORT);

HOSTENT\* hst;

// преобразование IP адреса из символьного в сетевой формат

if (inet\_addr(SERVERADDR) != INADDR\_NONE)

dest\_addr.sin\_addr.s\_addr = inet\_addr(SERVERADDR);

else

{

// попытка получить IP адрес по доменному имени сервера

if (hst = gethostbyname(SERVERADDR))

// hst->h\_addr\_list содержит не массив адресов,

// а массив указателей на адреса

((unsigned long\*)&dest\_addr.sin\_addr)[0] =

((unsigned long\*\*)hst->h\_addr\_list)[0][0];

else

{

printf("Invalid address %s\n", SERVERADDR);

closesocket(my\_sock);

WSACleanup();

return -1;

}

}

// адрес сервера получен - пытаемся установить соединение

if (connect(my\_sock, (sockaddr\*)&dest\_addr, sizeof(dest\_addr)))

{

printf("Connect error %d\n", WSAGetLastError());

return -1;

}

printf("Connection to %s successfully\n \

Type quit for quit\n\n", SERVERADDR);

char message[MAX\_BUF\_LENGTH];

while (strcmp(message, "quit") != 0) {

cout << "Enter the message.\n";

cin >> message;

// Шаг 4 - чтение и передача сообщений

int nsize;

// передаем строку клиента серверу

send(my\_sock, &message[0], MAX\_BUF\_LENGTH, 0);

recv(my\_sock, &message[0], MAX\_BUF\_LENGTH, 0);

cout << message << endl;

}

closesocket(my\_sock);

WSACleanup();

int i;

cin >> i;

return 0;

}