#include "stdafx.h"

#include "Winsock2.h" предоставляет функциональность для работы с сокетами

#pragma comment(lib, "WS2\_32.lib")

#include <string>

#include <iostream>

#include <ctime>

#include <chrono>

#define IP\_SERVER "192.168.134.8" IP-адрес сервера

using namespace std;

using namespace chrono;

string GetErrorMsgText(int code) возвращает текстовое описание ошибки по переданному коду ошибки

{

……

};

string SetErrorMsgText(string msgText, int code) формирование ошибки

{

………

};

char\* get\_message(int msg) возвращает текстовое представление выбранного типа сообщения по переданному числовому значению. Она используется для вывода типа выбранного сообщения пользователю

{

switch (msg)

{

case 1: return "Echo";

case 2: return "Time";

case 3: return "Random";

case 4: return "close";

default:

return "";

}

}

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

{

SOCKET cS; сокет сервера

WSADATA wsaData; настройки текущей версии

setlocale(0, "rus");

try

{

if (WSAStartup(MAKEWORD(2, 1), &wsaData) != 0)

throw SetErrorMsgText("Startup:", WSAGetLastError());

if ((cS = socket(AF\_INET, SOCK\_STREAM, NULL)) == INVALID\_SOCKET)

throw SetErrorMsgText("socket:", WSAGetLastError());

SOCKET cC; сокет клиента

if ((cC = socket(AF\_INET, SOCK\_STREAM, NULL)) == INVALID\_SOCKET)

throw SetErrorMsgText("socket:", WSAGetLastError());

SOCKADDR\_IN serv; настройки сервера

serv.sin\_family = AF\_INET;

serv.sin\_port = htons(2000);

serv.sin\_addr.s\_addr = inet\_addr("192.168.43.179");

if ((connect(cC, (sockaddr\*)&serv, sizeof(serv))) == SOCKET\_ERROR) установка соединения с сервером

throw SetErrorMsgText("connect:", WSAGetLastError());

while (true)

{

char message[50],

obuf[50];

int libuf = 0,

lobuf = 0;

puts("Choose");

int service;

puts("1 - Echo\n2 - Time\n3 - Random\n4 - close socket");

scanf("%d", &service);

char\* outMessage = new char[5];

strcpy(outMessage, get\_message(service));

if ((lobuf = send(cC, outMessage, strlen(outMessage) + 1, NULL)) == SOCKET\_ERROR)

throw SetErrorMsgText("send:", WSAGetLastError());

printf("sended: %s\n", outMessage);

if ((libuf = recv(cC, message, sizeof(message), NULL)) == SOCKET\_ERROR)

throw SetErrorMsgText("recv:", WSAGetLastError());

if (service == 4)

{

break;

}

if (!strcmp(message, "TimeOUT"))

{

puts("time out");

return -1;

}

if (service == 1)

{

for (int j = 30; j >= 0; --j)

{

Sleep(1000);

sprintf(outMessage, "%d", j);

if ((lobuf = send(cC, outMessage, strlen(outMessage) + 1, NULL)) == SOCKET\_ERROR)

throw SetErrorMsgText("send:", WSAGetLastError());

printf("send: %s\n", outMessage);

if ((libuf = recv(cC, message, sizeof(message), NULL)) == SOCKET\_ERROR)

throw SetErrorMsgText("recv:", WSAGetLastError());

printf("receive: %s\n", message);

}

}

else if (service == 2 || service == 3)

{

printf("receive: %s\n", message);

}

}

if (closesocket(cS) == SOCKET\_ERROR)

throw SetErrorMsgText("closesocket:", WSAGetLastError());

if (WSACleanup() == SOCKET\_ERROR)

throw SetErrorMsgText("Cleanup:", WSAGetLastError());

}

catch (string errorMsgText)

{

printf("\n%s", errorMsgText.c\_str());

}

system("pause");

return 0;

}