bool GetRequestFromClient(char\* name, short port, sockaddr\* from, int\* flen)

{

char nameServer[50];

объявление массива символов для хранения имени сервера.

memset(from, 0, sizeof(flen));

заполнение области памяти, на которую указывает from, нулевыми байтами, размер которых равен значению, на которое указывает flen.

if ((serverSocket = socket(AF\_INET, SOCK\_DGRAM, NULL)) == INVALID\_SOCKET)

{

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

}

создание датаграммного сокета UDP с помощью функции socket(). Функция возвращает новый сокет, который связывается с доменом AF\_INET (IPv4) и типом сокета SOCK\_DGRAM (датаграммный сокет). Если функция возвращает INVALID\_SOCKET, то вызывается функция SetErrorMsgText(), которая генерирует сообщение об ошибке и завершает программу

SOCKADDR\_IN serv;

объявление структуры SOCKADDR\_IN для хранения параметров сервера.

serv.sin\_family = AF\_INET;

задание семейства адресов для структуры serv. Здесь указывается AF\_INET, что означает использование протокола IPv4

serv.sin\_port = htons(port);

задание порта для структуры serv. Передаваемый порт приводится к сетевому порядку байтов с помощью функции htons().

serv.sin\_addr.s\_addr = INADDR\_ANY;

задание адреса для структуры serv. Здесь указывается INADDR\_ANY, что означает принятие входящих пакетов на любой доступный сетевой интерфейс.

//свзять пар-ры сокета

if (bind(serverSocket, (LPSOCKADDR)&serv, sizeof(serv)) == SOCKET\_ERROR)

{

throw SetErrorMsgText("Походу такой сервер уже есть:", WSAGetLastError());

}

связывание созданного сокета с портом и адресом, указанными в структуре serv. Если произошла ошибка, вызывается функция SetErrorMsgText(), которая генерирует сообщение об ошибке и завершает программу.

////////////////////////////////////////////////////////////

char\* message = new char[strlen(name)];

выделение динамической памяти под массив символов message длиной strlen(name)

int buf = 0;

объявление переменной для хранения количества принятых байт

SOCKADDR\_IN client;

объявление структуры SOCKADDR\_IN для хранения параметров клиента.

int size = sizeof(client);

определение размера структуры client.

while (strcmp(name, message))

{

цикл, который продолжается до тех пор, пока name не будет равно message

memset(&client, 0, size);

заполнение области памяти, на которую указывает &client, нулевыми байтами, размер которых равен size

buf = recvfrom(serverSocket, message, strlen(message), NULL, (sockaddr\*)&client, &size);

ожидает прихода UDP-пакета от клиента на заданный сокет serverSocket, считывает его в буфер message, запоминает адрес клиента в структуре client и возвращает количество считанных байтов.

char\* addr = (char\*)&client.sin\_addr;

получаем адрес клиента из структуры client.

hostent\* cl = gethostbyaddr(addr, 4, AF\_INET);

получаем информацию об адресе клиента.

cout << "\nClient name : " << cl->h\_name << endl;

выводим имя клиента.

switch (buf)

проверяем количество считанных байтов:

{

case SOCKET\_ERROR:

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

break;

case WSAETIMEDOUT:

return false;

default:

break;

}

message[buf] = 0x00;

добавляем завершающий нулевой символ к буферу

}

\*from = \*((sockaddr\*)&client);

сохраняем адрес клиента в from

\*flen = sizeof(client);

сохраняем размер структуры client в flen.

return true;

возвращаем true, если получили пакет от клиента успешно.

}