Московский государственный технический университет им. Н.Э.Баумана

Утверждаю:

Галкин В.А.
"__"_____2022 г.

Курсовая работа по дисциплине «Сетевые технологии» «Локальная безадаптерная сеть»

Описание программы
(вид документа)

писчая бумага
(вид носителя)

27
(количество листов)

ИСПОЛНИТЕЛИ:

студенты группы ИУ5Ц-82Б
ИУ5Ц-81Б
Пылаев Б.
Чиварзин А. "___"_____2022 г.
Коротенко Е.

Москва - 2022

1. Общие сведения.

Наименование: “Программа посылки сообщений через com-порты Чат”.

Программа выполняется на языке программирования C# и работает под управлением операционной системы Windows 98 и выше.

2. Назначение разработки.

Программа должна реализовывать функцию передачи текстовых файлов между двумя ПЭВМ, соединенными через интерфейс RS-232C нуль-модемным кабелем.

4. Используемые технические средства.

Программа должна работать на IBM-совместимой ЭВМ следующей конфигурации:

1. Центральный процессор Perntium I или выше;
2. Объем оперативной памяти 32 Мб;
3. Видеоадаптер и монитор VGA и выше;
4. Стандартная клавиатура;
5. Свободного пространства на жестком диске 2Мб;

Для работы программы требуются два IBM-совместимых компьютера, соединенных нуль-модемным кабелем через интерфейс RS-232C.

5. Входные и выходные данные.

5.1. Входные данные.

Входными данными является текстовое сообщение, набранное пользователем.

5.2. Выходные данные.

Выходными данными являются:

1. текст переданного сообщения на ПЭВМ;
2. сообщения об ошибках и выполнении передачи.

6. Спецификация данных.

6.1. Внутренние данные.

Данные указаны без учета стартовых и стоповых байтов.

Запрос на соединение:

Наименование	Тип поля	Размер (байт)
UPLINK	Byte	1

Поддержание соединения:

Наименование	Тип поля	Размер (байт)
LINKACTIVE	Byte	1

Положительная квитанция:

Наименование	Тип поля	Размер (байт)
ACK	Byte	1

Разрыв соединения:

Наименование	Тип поля	Размер (байт)
DOWNLINK	Byte	1

Информационный блок:

Наименование	Тип поля	Размер (байт)
DAT	Byte	1
Data	AnsiString	Sizeof(Data)

6.2. Структура сообщения.

Программа работает с текстовыми сообщениями стандарта ANSI размером не более 255 символов.

7. Спецификация функций.

void WriteData(string msg,FrameType CurrentFrameType, bool msg_no_display)
– процедура записи в порт;

void DisplayData(MessageType type, string msg) – процедура вывода данных с порта на экран;

bool OpenPort() – функция открытия порта;

void SetParityValues(object obj) – процедура заполнения выпадающего списка «Бит четности» в форме «Чат» значениями бита четности;

void SetStopBitValues(object obj) - процедура заполнения выпадающего списка «Стопбиты» в форме «Чат» значениями количества стоповых бит;

void SetPortNameValues(object obj) - процедура заполнения выпадающего списка «Порт» в форме «Чат» именами доступных COM-портов;

void comPort_DataReceived(object sender, SerialDataReceivedEventArgs e) – процедура, которая вызывается, когда в буфере появляются данные;

bool ClosePort() – функция закрытия порта;

void cmdOpen_Click(object sender, EventArgs e) **–** процедура обработки нажатия кнопки «Открыть порт» в форме «Чат»;

void cmdClose_Click(object sender, EventArgs e) - процедура обработки нажатия кнопки «Закрыть порт» в форме «Чат»;

void cmdConnect_Click(object sender, EventArgs e) **–** процедура обработки нажатия кнопки «Подключиться» в форме «Чат»;

void cmdDisconnect_Click(object sender, EventArgs e) **–** процедура обработки нажатия кнопки «Разъединить» в форме «Чат»;

void cmdSend_Click(object sender, EventArgs e) **–** процедура обработки нажатия кнопки «Отправить» в форме «Чат»;

void frmMain_Load(object sender, EventArgs e) – процедура обработки события загрузки формы «Чат»;

void SetDefaults() – процедура установки параметров соединения по умолчанию;

void LoadValues() – процедура загрузки параметров COM-порта;

void SetControlState() – процедура установки состояний органов управления при первой загрузке формы «Чат»;

void Tick(object sender, EventArgs e) – процедура обработки события истечения интервала ожидания кадра ACK_UPLINK;

void Tick_mytimer1(object sender, EventArgs e) – процедура обработки события истечения интервала ожидания кадра ACK_DOWNLINK;

void Tick_mytimer2(object sender, EventArgs e) - процедура обработки события монитора активности соединения Назначение обработчика - отправка по истечении интервала кадра ACTIVELINK и монитор соединения (монитор статуса comm.LinkActive);

void rtbDisplay_TextChanged(object sender, EventArgs e) – процедура обработки события изменения текста в диалоговом окне на форме «Чат», синхронна с событием comm.DataReceived(исключая случаев "Port Opened AT DateTime.NOW" и отображения события отправки кадров);

void frmMain_FormClosing(object sender, FormClosingEventArgs e) – процедура обработки события во время закрытия формы «Чат»;

void frmMain_FormClosed(object sender, FormClosedEventArgs e) - – процедура обработки события закрытие формы «Чат»;

void аквaToolStripMenuItem_Click(object sender, EventArgs e) – процедура обработки нажатия пункта меню «Аква» меню «Вид»;

void золотойToolStripMenuItem_Click(object sender, EventArgs e) – процедура обработки нажатия пункта меню «Золотой» меню «Вид»;

void стандартныйToolStripMenuItem_Click(object sender, EventArgs e) – процедура обработки нажатия пункта меню «Стандартный» меню «Вид»;

void toolStripMenuItem4_Click(object sender, EventArgs e) – процедура обработки нажатия пункта меню «О программе» меню «Справка»;

void открытьФайлИсторииToolStripMenuItem_Click(object sender, EventArgs e) - процедура обработки нажатия пункта меню «Открыть файл истории» меню «История»;

void openFileDialog1_FileOk(object sender, EventArgs e) – процедура открытия окна-диалога для выбора файла;

void button1_Click(object sender, EventArgs e) – процедура обработки нажатия кнопки «Ок» на форме «О программе»;

void Form3_Load(object sender, EventArgs e) – процедура обработки события загрузки формы «История»;

void HammCode(byte a, byte n, out byte code) - процедура кодирования кодовой комбинации кодом Хэмминга.

int CountBit(byte a) - функция перевода чисел из десятичного представления в двоичное;

bool HammDecode(byte a,out byte code,byte n) – процедура декодирования;

List<byte> CodingStr(string str) – функция кодирования строки текста;

string DeCodingStr(byte[] str) – функция декодирования исходной строки байтов в строку, введенную пользователем.

Листинг основных функций

Connection.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.IO.Ports;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WahChat
{
    class Connection
    {
        public bool isPortsOpened = false;

        private byte boundByte = 0xFF;

        private SerialPort incomePort;
        private SerialPort outcomePort;

        public bool isMaster;

        private List<byte> bytesBuffer = new List<byte>();

        public Connection(string incomePortName, string outcomePortName, bool
isMaster)
        {
            this.isPortsOpened = OpenPorts(incomePortName, outcomePortName,
isMaster);

            // ..
        }

        /// <summary>
        /// Открытие портов
        /// </summary>
        private bool OpenPorts(string incomePortName, string outcomePortName, bool
isMaster)
        {
            // Создаем объекты портов.
            this.incomePort = new SerialPort(incomePortName);
            this.outcomePort = new SerialPort(outcomePortName);

            this.isMaster = isMaster;

            // Настраиваем порты.
            this.incomePort.Parity = Parity.Even;
            this.incomePort.Handshake = Handshake.RequestToSend;
            this.incomePort.BaudRate = 9600;
            //this.incomePort.ReadBufferSize = 4 * 1024; // TODO: Надо пересчитать
размер буфера.
            this.incomePort.DataReceived += new
SerialDataReceivedEventHandler(RecieveBytes);

            this.outcomePort.Parity = Parity.Even;
            this.outcomePort.Handshake = Handshake.RequestToSend;
            this.outcomePort.BaudRate = 9600;

            // Открываем порты.
            try
            {
                this.incomePort.Open();
```

```
                this.outcomePort.Open();
            } catch (Exception ex)
            {
                //Здесь должна быть ошибка невозможности использовать COM-порты
                throw ex;
            }

            return (this.incomePort.IsOpen && this.outcomePort.IsOpen);
        }

        /// <summary>
        /// Закрытие портов
        /// </summary>
        private bool ClosePorts()
        {
            // Закрываем порты.
            this.incomePort.Close();
            this.outcomePort.Close();

            return (this.incomePort.IsOpen && this.outcomePort.IsOpen);
        }

        /// <summary>
        /// Пересылка байтов
        /// </summary>
        public void SendBytes(List<byte> list)
        {
            // TODO: Кодирование
            List<byte> hamm = list;

            // Делаем так, чтобы внутри кадра не встречалось boundByte.
            List<byte> safeList = new List<byte>(hamm.Count);
            foreach (var b in hamm)
            {
                if ((b & 0x7F) == 0x7F)
                {
                    safeList.Add(0x7F);
                    safeList.Add((byte)(b & 0x80));
                }
                else
                {
                    safeList.Add(b);
                }
            }

            // Добавляем стартовый и конечный байт
            safeList.Insert(0, boundByte);
            safeList.Add(boundByte);

            if (this.outcomePort.WriteBufferSize - this.outcomePort.BytesToWrite <
safeList.Count)
            {
                // Если сообщение не влезло в порт, то надо что-то с этим делать.
                // То ли очередь придумать, то ли ещё что.
                return;
            }

            byte[] arr = safeList.ToArray();
            this.outcomePort.Write(arr, 0, arr.Length);
        }

        /// <summary>
        /// Получение байтов
        /// </summary>
        public void RecieveBytes(object sender, SerialDataReceivedEventArgs e)
        {
            int bytes = incomePort.BytesToRead;
            byte[] comBuffer = new byte[bytes];
```

```
            // Записываем в массив данные от ком порта.
            incomePort.Read(comBuffer, 0, bytes);

            foreach (byte incomeByte in comBuffer)
            {
                if (incomeByte == boundByte)
                {
                    if (this.bytesBuffer.Count > 0)
                    {
NetworkService.GetSharedService().HandleMessage(this.bytesBuffer);
                    }

                    this.bytesBuffer = new List<byte>();
                }
                else
                {
                    this.bytesBuffer.Add(incomeByte);
                }
            }
        }
    }
}
```

HammCoder.cs (кодирование и декодирование)

```
using System;
using System.Collections.Generic;
using System.Text;

namespace WahChat
{
    class HammCoder
    {
        static void HammCode(byte a, byte n, out byte code)
        {
            a <<= 3;

            while (CountBit(a) != CountBit(n)) n <<= 1;
            code = a;
            do
            {
                a ^= n;
                do
                {
                    n >>= 1;
                }
                while (CountBit(a) != CountBit(n));
            } while (a > 7);
            code |= a;
        }

        static int CountBit(byte a)
        {
            int count = 0;
            while (!(a == 0))
            {
                a /= 2;
                ++count;
            }

            return (count);
        }

        static bool HammDecode(byte a, out byte code, byte n)
        {
            while (CountBit(a) != CountBit(n)) n <<= 1;
            code = a;
            do
```

```
        {
            code ^= n;
            do
            {
                n >>= 1;
            }
            while (CountBit(code) != CountBit(n));
        }
        while (code > 7);

        if (code == 0)
        {
            code = (byte)(a >> 3);
            return (true);
        }
        else return (false);
    }


    static public List<byte> CodeString(string str)
    {
        char[] ChrArr;
        byte LShort, HShort;
        byte LWord, HWord;
        byte Gx = 11;
        byte Code;
        List<byte> CodingString = new List<byte>();

        ChrArr = str.ToCharArray();

        short[] ShortArr = new short[str.Length];

        for (int i = 0; i < ChrArr.Length; i++)
        {
            ShortArr[i] = (short)ChrArr[i];
        }

        for (int i = 0; i < ShortArr.Length; i++)
        {
            LShort = (byte)(ShortArr[i] & 0x00FF);
            HShort = (byte)((ShortArr[i] & 0xFF00) >> 8);

            LWord = (byte)(LShort & 0x0F);
            HWord = (byte)((LShort & 0xF0) >> 4);

            HammCode(LWord, Gx, out Code);
            CodingString.Add(Code);
            HammCode(HWord, Gx, out Code);
            CodingString.Add(Code);

            LWord = (byte)(HShort & 0x0F);
            HWord = (byte)((HShort & 0xF0) >> 4);

            HammCode(LWord, Gx, out Code);
            CodingString.Add(Code);
            HammCode(HWord, Gx, out Code);
            CodingString.Add(Code);
        }

        return (CodingString);
    }

    /// <summary>
    static public string DecodeString(byte[] str)
    {
        char ch;
        string DecodedString = "";
        byte LWord, HWord; // младшая и старшие части символа
        short LShort, HShort;
```

```
            byte Gx = 11;

            for (int i = 0; i < str.Length; i++)
            {
                if (!HammDecode(str[i], out LWord, Gx)) return ("");
                if (!HammDecode(str[++i], out HWord, Gx)) return ("");

                LShort = (short)((HWord << 4) | LWord);

                if (!HammDecode(str[++i], out LWord, Gx)) return ("");
                if (!HammDecode(str[++i], out HWord, Gx)) return ("");
                HShort = (short)((HWord << 4) | LWord);

                ch = Convert.ToChar((HShort << 8) | LShort);
                Console.WriteLine("ch={0}", ch);
                DecodedString += ch;
            }

            return (DecodedString);
        }
    }
}
```

Frame.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WahChat
{
    class Frame
    {
        /// <summary>
        /// Тип кадра
        /// </summary>
        public enum Type: byte
        {
            /// Установка логического соединения.
            Link,
            /// Получение списка пользователей.
            Ask,
            /// Отправка сообщений пользователя.
            Data,
            /// Отправка запроса на переотправку сообщения.
            Error,
            /// Разъединение соединения.
            Downlink
        }

        public Type type;
        public List<byte> data;

        public int authorID;
        public string message;

        public Frame()
```

```
        {
            // ..
        }

        public Frame(List<byte> data)
        {
            this.data = data;

            byte typeByte = data[0];
            switch (typeByte)
            {
                case (byte)Type.Link:
                    this.type = Type.Link;
                    break;

                case (byte)Type.Ask:
                    this.type = Type.Ask;
                    break;

                case (byte)Type.Data:
                    this.type = Type.Data;
                    this.authorID = (int)data[1];

                    byte[] byteArray = data.ToArray();

                    int messageLength = data.Count - 2;
                    byte[] messageData = new byte[messageLength];

                    Array.Copy(byteArray, 2, messageData, 0, messageLength);

                    this.message = System.Text.Encoding.UTF8.GetString(messageData,
0, messageData.Length);

                    break;

                case (byte)Type.Error:
                    this.type = Type.Error;
                    break;

                case (byte)Type.Downlink:
                    this.type = Type.Downlink;
                    break;

                default:
                    this.type = Type.Link;
                    break;
            }

        }

        public Frame(Type type)
        {
            List<byte> data = new List<byte>();
            data.Add((byte)type);
            this.data = data;
        }
    }
}
```

NetworkService.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.IO.Ports;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;
using WahChat.Messages;

namespace WahChat
{
    class NetworkService
    {
        public Label notificationLabel;
        public Button connectButton;
        public ListBox chatBox; //Отправленные со станции сообщения
        public ListBox chatInBox; //Поступившие на станцию сообщения от других
станций

        private NetworkService()
        {
            // ..
        }

        private static readonly NetworkService _sharedService = new NetworkService();

        public static NetworkService GetSharedService()
        {
            return _sharedService;
        }

        /// Текущее соединение
        public Connection currentConnection;

        /// Текущая сессия
        public Session currentSession;

        /// <summary>
        /// Создание соединения
        /// </summary>
        public void CreateConnection(string incomePortName, string outcomePortName,
bool isMaster)
        {
            this.currentConnection = new Connection(incomePortName, outcomePortName,
isMaster);

            // формирование LINK кадра..
            // отправка LINK кадра..
        }

        /// <summary>
        /// Закрытие соединения
        /// </summary>
        public void CloseConnection()
        {
            Frame frame = new Frame(Frame.Type.Downlink);
            this.SendFrame(frame);
        }

        /// <summary>
        /// Обработка пришедшего сообщения
        /// </summary>
        public void HandleMessage(List<byte> message)
        {
            Frame frame = new Frame(message);
            this.HandleFrame(frame);
        }

        /// <summary>
        /// Обработка пришедшего кадра
        /// </summary>
        public void HandleFrame(Frame frame)
        {
            switch (frame.type)
```

```
            {
                case Frame.Type.Link:

                    this.notificationLabel.Invoke((MethodInvoker)delegate {

                        // Running on the UI thread
                        this.notificationLabel.Text = "Соединение установлено";
                        this.connectButton.Text = "Войти";
                    });

                    // Если станция не ведущая, то отправляем дальше
                    if (currentConnection.isMaster == false)
                    {
                        this.SendFrame(frame);
                    }

                    break;

                case Frame.Type.Ask:

                    // Если станция не ведущая, то отправляем дальше
                    if (currentConnection.isMaster == false)
                    {
                        this.SendFrame(frame);
                    }

                    break;

                case Frame.Type.Data:

                    ChatMessage msg = new ChatMessage(DateTime.Now.ToString("hh:mm"),
frame.authorID, frame.message);
                    // Сравниваем ID сессии и автора сообщения. В зависимости от него
помещаем сообщение во входящие или отправленные.
                    if (NetworkService.GetSharedService().currentSession.username ==
msg.authorID)
                    {
                        this.chatBox.Invoke((MethodInvoker)delegate {

                            // Running on the UI thread
                            this.chatBox.Items.Add(msg.ToString());
                        });
                    } else
                    {
                        this.chatInBox.Invoke((MethodInvoker)delegate {

                            // Running on the UI thread
                            this.chatInBox.Items.Add(msg.ToString());
                        });
                    }

                    // Если станция не ялвяется отправителем, то отправляем дальше
                    if (currentSession.username != frame.authorID)
                    {
                        this.SendFrame(frame);
                    }

                    break;

                case Frame.Type.Error:

                    // Если станция не ведущая, то отправляем дальше
                    if (currentConnection.isMaster == false)
                    {
                        this.SendFrame(frame);
                    }

                    break;
```

```
                case Frame.Type.Downlink:

                    // Если станция не ведущая, то отправляем дальше
                    if (currentConnection.isMaster == false)
                    {
                        this.SendFrame(frame);
                    }

                    System.Windows.Forms.Application.Exit();

                    break;
            }
        }

        public void SendFrame(Frame frame)
        {
            this.currentConnection.SendBytes(frame.data);
        }

        public void SendMessage(string message)
        {
            byte[] byteStr = System.Text.Encoding.UTF8.GetBytes(message);

            List<byte> data = new List<byte>();

            data.Add((byte)Frame.Type.Data);
            data.Add((byte)currentSession.username);

            foreach (byte b in byteStr)
            {
                data.Add(b);
            }

            Frame frame = new Frame();
            frame.data = data;

            this.SendFrame(frame);
        }

        /// <summary>
        /// Список доступных портов
        /// </summary>
        public string[] GetPortsNames()
        {
            return SerialPort.GetPortNames();
        }

        /// <summary>
        /// Создание сессии
        /// </summary>
        public void CreateSession(int username)
        {
            this.currentSession = new Session(username);

            // формирование кадра с username..
            // отправка кадра с username..
        }

        /// <summary>
        /// Закрытие сессии
        /// </summary>
        public void CloseSession()
        {
            // ..
        }
    }
}
```

Session.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WahChat
{
    class Session
    {
        public int username;

        public Session(int username)
        {
            this.username = username;
        }
    }
}
```

Login.cs (форма авторизации и подключения)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WahChat
{
    public partial class Login : Form
    {
        public Login()
        {
            InitializeComponent();

            NetworkService.GetSharedService().notificationLabel =
this.notificationLabel;
            NetworkService.GetSharedService().connectButton = this.loginButton;

            this.loginButton.Text = "Подключиться";

            if (NetworkService.GetSharedService().currentConnection.isPortsOpened)
            {
                this.notificationLabel.Text = "Порты открыты";
            }
            else
            {
                this.notificationLabel.Text = "Ошибка открытия портов";
            }
        }

        private void loginButton_Click(object sender, EventArgs e)
        {
            int username = int.Parse(textBox.Text);
            NetworkService.GetSharedService().CreateSession(username);

            Frame frame = new Frame(Frame.Type.Link);
            NetworkService.GetSharedService().SendFrame(frame);
```

```
            this.notificationLabel.Text = "Отправка..";

            this.Hide();

            Chat chatForm = new Chat();
            chatForm.Show();
        }

        private void Login_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Setup.cs (форма настроек соединения)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WahChat
{
    public partial class Setup : Form
    {
        public Setup()
        {
            InitializeComponent();
        }

        private void Setup_Load(object sender, EventArgs e)
        {
            // Показываем список COM-портов.
            string[] portNames = NetworkService.GetSharedService().GetPortsNames();
            foreach (string portName in portNames)
            {
                incomePortBox.Items.Add(portName);
                outcomePortBox.Items.Add(portName);
            }

            //Добавим список скоростей COM-портов
            box_speed1.Items.Clear();
            box_speed2.Items.Clear();
            box_speed1.Items.Add(21600);
            box_speed2.Items.Add(21600);
        }

        private void connectButton_Click(object sender, EventArgs e)
        {
            // Проверяем, выставлены ли порты.
            if ((incomePortBox.SelectedItem != null) && (outcomePortBox.SelectedItem
!= null))
            {

                string incomePort = incomePortBox.SelectedItem.ToString();
```

```
                string outcomePort = outcomePortBox.SelectedItem.ToString();

                // Если порты одинаковые, то показываем ошибку.
                if (incomePort == outcomePort)
                {
                    MessageBox.Show("Выберите различные COM-порты", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                    return;
                }

                // Стартуем соединение.
                NetworkService.GetSharedService().CreateConnection(incomePort,
outcomePort, checkBox.Checked);

                this.Hide();

                Login loginForm = new Login();
                loginForm.Show();
            }
            else
            {
                MessageBox.Show("Оба COM-порта должны быть выбраны", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            }
        }
    }
}
```

Chat.cs (форма чата)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WahChat
{
    public partial class Chat : Form
    {
        public Chat()
        {
            InitializeComponent();

            NetworkService.GetSharedService().chatBox = this.chatBox; //Отправленные
            NetworkService.GetSharedService().chatInBox = this.chatInbox; //Входящие

            this.usernameLabel.Text = String.Format("Подключен как \"{0}\"",
NetworkService.GetSharedService().currentSession.username);
            this.chatBox.SelectionMode = SelectionMode.None;
            this.chatInbox.SelectionMode = SelectionMode.None;
        }

        private void SendButton_Click(object sender, EventArgs e)
        {
            string message = messageBox.Text;
            NetworkService.GetSharedService().SendMessage(message);

            messageBox.Clear();
        }

        private void closeButton_Click(object sender, EventArgs e)
        {
```

```
            NetworkService.GetSharedService().CloseConnection();
        }

        private void Chat_Load(object sender, EventArgs e)
        {

        }

        private void button_about_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Выполнена в рамках курса \"Сетеввые технологии в АСОИУ\"\nИсполнители:\tКрротенко Е.А. ИУ5Ц-81Б\n\t\tПылаев Б.А. ИУ5Ц-82Б\n\t\tЧиварзин А.Е. ИУ5Ц-82Б\nПреподаватель:\tГалкин В.А.", "Разработка");
        }
    }
}
```