Министерство образования и науки Республики Алтай

БПОУ РА «ГОРНО-АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ им. М.З. Гнездилова»

**Практические работы**

Специальность - Информационные системы и программирование

Выполнили студентки 3 курса 1212 группы

Черепанова Дарья Сергеевна и Дымова Милена Юрьевна

Руководитель: Ручко Дмитрий Станиславович

Защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_«\_\_\_»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

**Практическая работа 1.**

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 Pr1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

button2.Cursor = System.Windows.Forms.Cursors.Hand;

button1.Text = "&Print";

label1.Text = "&Print";

label1.TabIndex = 9;

button1.TabIndex = 10;

}

private void btnAdd\_Click(object sender, EventArgs e)

{

if(!string.IsNullOrWhiteSpace(txtName.Text)&& !lstNames.Items.Contains(txtName.Text))

lstNames.Items.Add(txtName.Text);

}

private void Form1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (e.KeyChar >= 48 && e.KeyChar <= 57)

{

MessageBox.Show($"Form.KeyPress: '{e.KeyChar}' pressed.");

switch (e.KeyChar)

{

case (char)49:

case (char)52:

case (char)55:

MessageBox.Show($"Form.KeyPress: '{e.KeyChar}' consumed.");

e.Handled = true;

break;

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

Cursor.Position = PointToScreen(button2.Location);

button1.Text = "Print && Close";

}

private void button2\_Click(object sender, EventArgs e)

{

Cursor.Position = PointToScreen(button1.Location);

//button2.Cursor = System.Windows.Forms.Cursors.Hand;

}

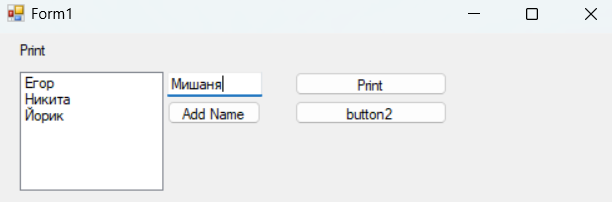
private void lstNames\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}



**Практическая работа 2.**

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 Pr2.1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void btnExit(object sender, EventArgs e)

{

this.Close();

}

private void btnCount(object sender, EventArgs e)

{

int summa = Int32.Parse(txtA.Text) +

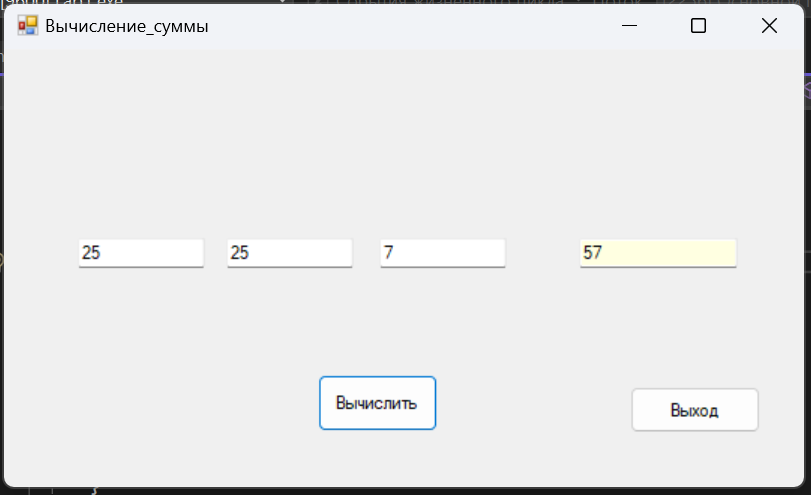
Int32.Parse(txtB.Text) + Int32.Parse(txtC.Text);

txtSumma.Text = summa.ToString();

}

}

}



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 Pr2.2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void MqenuStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

private void черныйToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.txt.BackColor = System.Drawing.Color.Black;

}

private void красныйToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.txt.BackColor = System.Drawing.Color.Red;

}

private void синийToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.txt.BackColor = System.Drawing.Color.Blue;

}

private void зеленыйToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.txt.BackColor = System.Drawing.Color.Green;

}

private void txt\_TextChanged(object sender, EventArgs e)

{

}

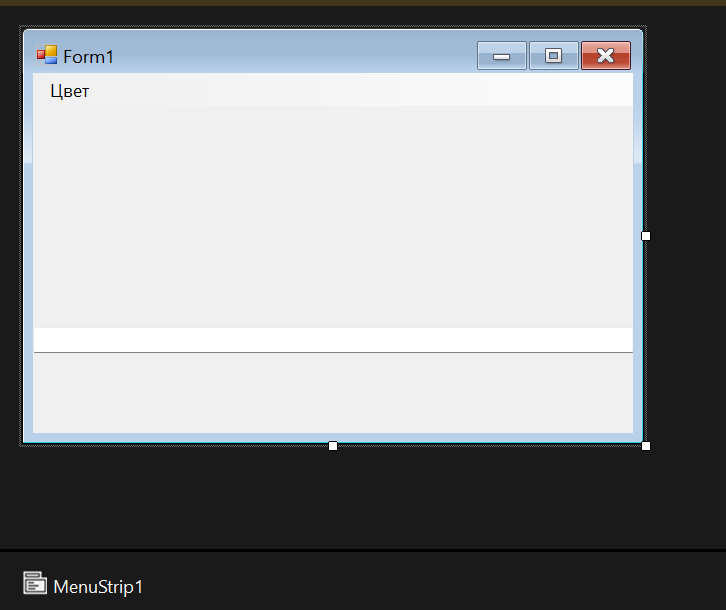
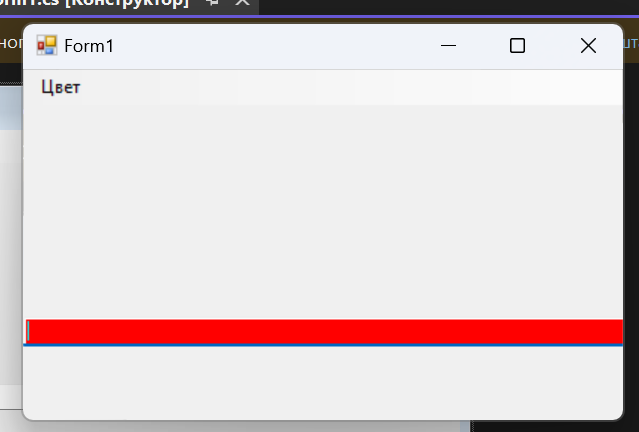
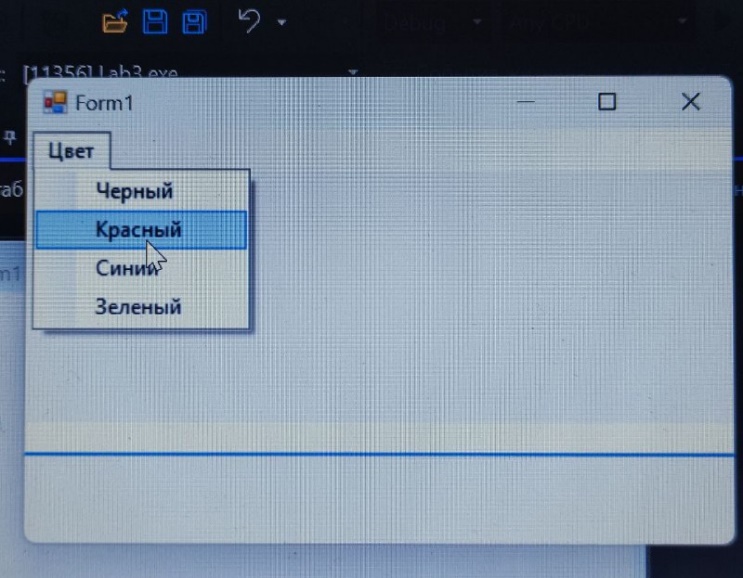
private void Form1\_Load(object sender, EventArgs e)

{

}

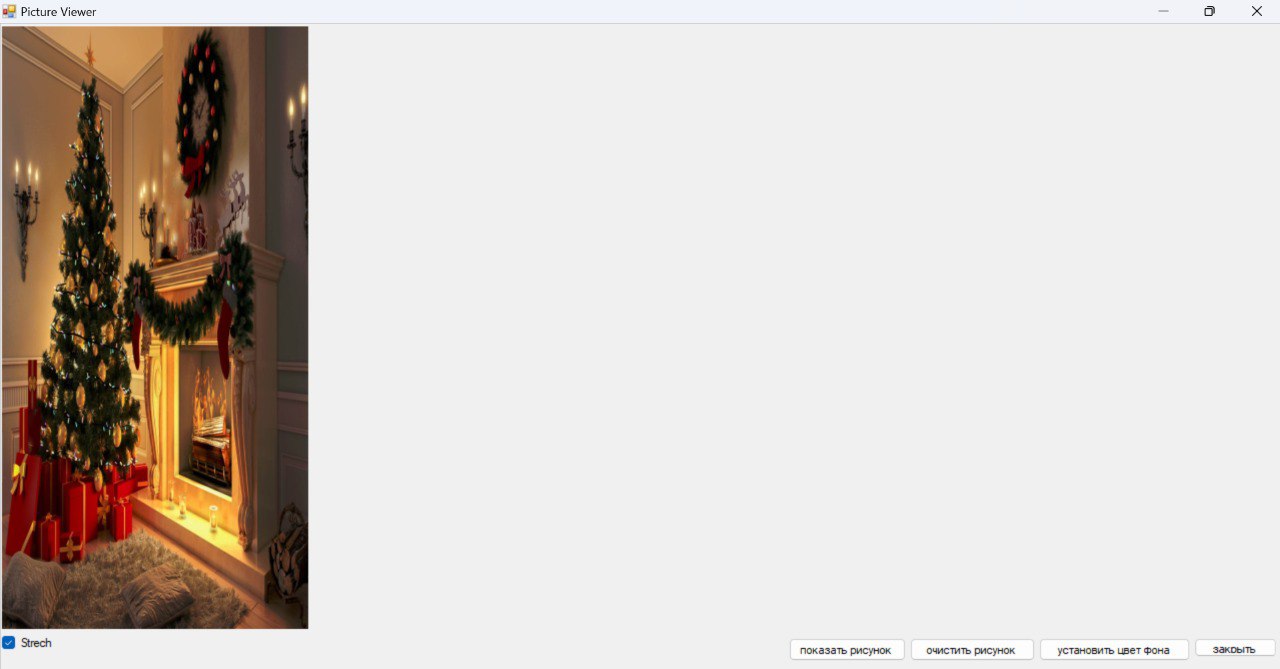
}

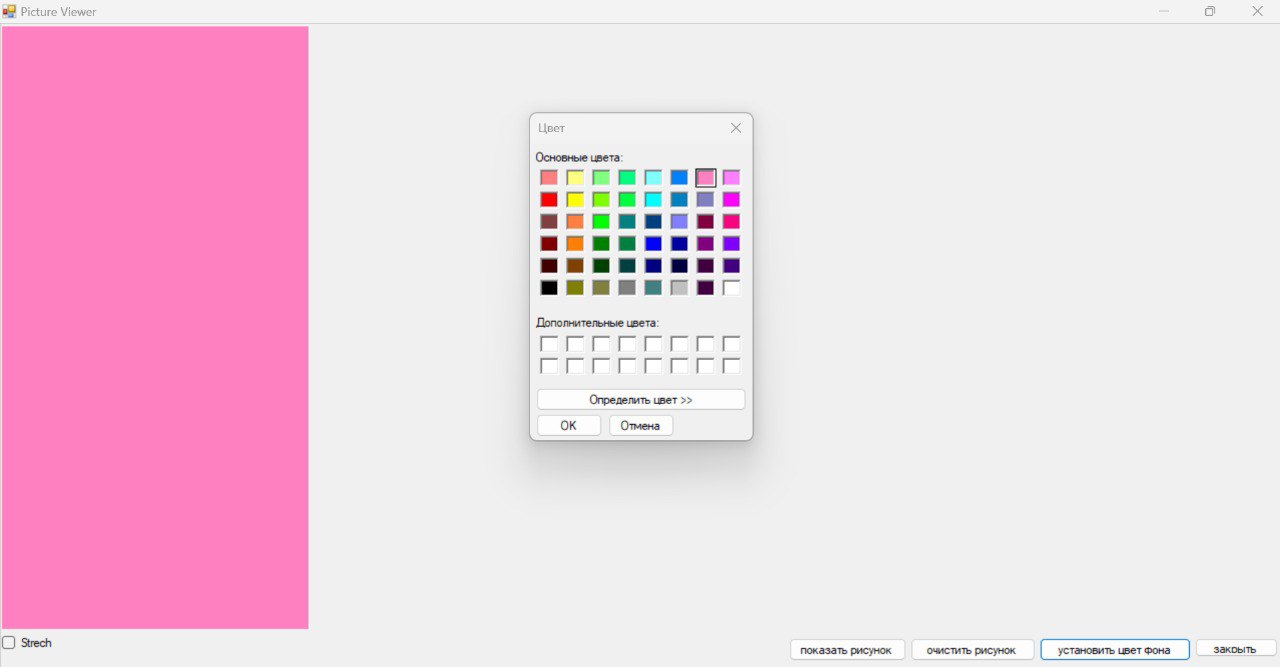
}

****

**Практическая работа 3.**

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 Pr3  
{  
    public partial class Form1 : Form  
    {  
        public Form1()  
        {  
            InitializeComponent();  
        }  
  
        private void showButton\_Click(object sender, EventArgs e)  
        {  
            if (openFileDialog1.ShowDialog()==DialogResult.OK)  
            {  
                pictureBox1.Load(openFileDialog1.FileName);  
            }  
        }  
  
        private void clearButton\_Click(object sender, EventArgs e)  
        {  
            pictureBox1.Image = null;  
        }  
  
        private void backraundButton\_Click(object sender, EventArgs e)  
        {  
            if(colorDialog1.ShowDialog()== DialogResult.OK)  
            {  
                pictureBox1.BackColor = colorDialog1.Color;  
            }  
        }  
  
        private void closeButton\_Click(object sender, EventArgs e)  
        {  
            this.Close();  
        }  
  
        private void checkBox1\_CheckedChanged(object sender, EventArgs e)  
        {  
            if (checkBox1.Checked)  
                pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;  
            else  
                pictureBox1.SizeMode = PictureBoxSizeMode.Normal;  
        }  
    }  
}





**Практическая работа 4.**

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 Pr4.1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

file1.Filter = "(\*.jpg)|\*.jpg";

}

private void btn\_Click(object sender, EventArgs e)

{

string fname;

file1.ShowDialog();

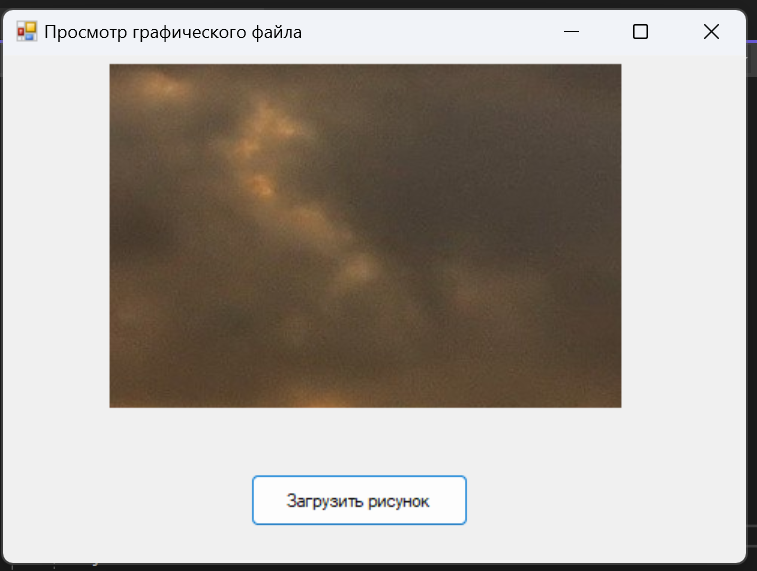
fname = file1.FileName;

pct.Image = Image.FromFile(fname);

}

}

}



using System.IO;

namespace Pr4.2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button4\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void btnAdd\_Click(object sender, EventArgs e)

{

lst.Items.Add(txt.Text);

txt.Clear();

}

private void btnOpen\_Click(object sender, EventArgs e)

{

string fileName = txt.Text;

if (string.IsNullOrEmpty(fileName))

{

MessageBox.Show("Введите имя файла");

return;

}

if (!fileName.EndsWith(".txt"))

{

fileName += ".txt";

}

try

{

using (StreamReader reader = new StreamReader(fileName))

{

lstFromfile.Items.Clear();

string line;

while ((line = reader.ReadLine()) != null)

{

lstFromfile.Items.Add(line);

}

}

MessageBox.Show("Список успешно загружен из файла " + fileName);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке из файла: " + ex.Message);

}

}

private void btnSave\_Click(object sender, EventArgs e)

{

string fileName = txt.Text;

if (string.IsNullOrEmpty(fileName))

{

MessageBox.Show("Введите имя файла");

return;

}

if (!fileName.EndsWith(".txt"))

{

fileName += ".txt";

}

try

{

using (StreamWriter writer = new StreamWriter(fileName))

{

foreach (string item in lst.Items)

{

writer.WriteLine(item);

}

}

MessageBox.Show("Список успешно сохранен в файл " + fileName);

}

catch (Exception ex)

{

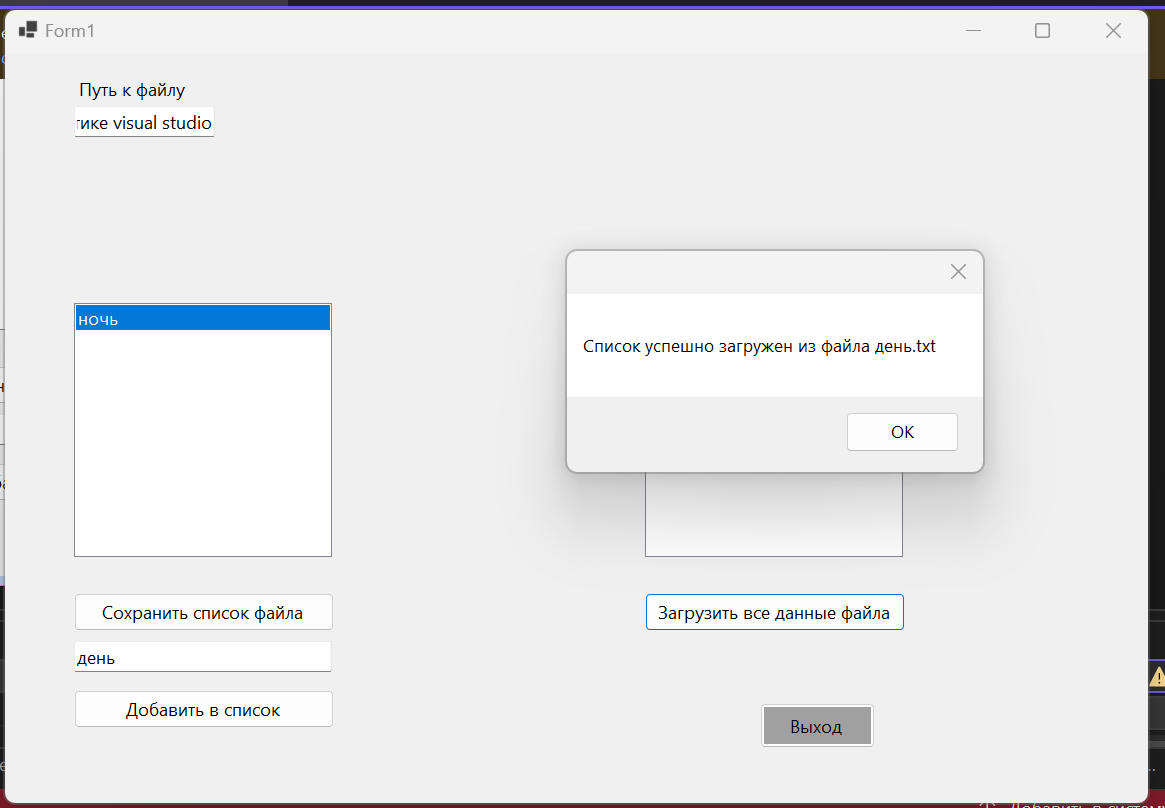
MessageBox.Show("Ошибка при сохранении в файл: " + ex.Message);

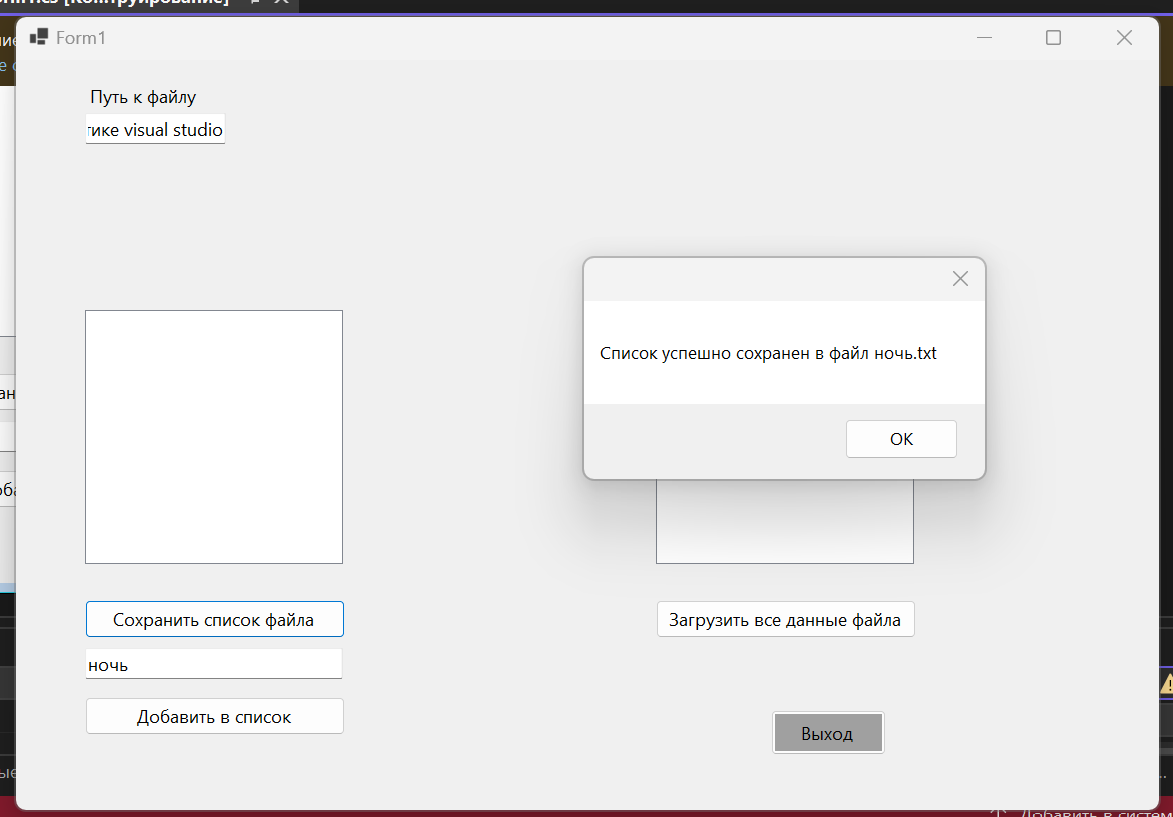
}

}

}

}





**Практическая работа 5.**

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 Pr5

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

file1.Filter = "(\*.jpg)|\*.jpg";

}

private void AddBtn\_Click(object sender, EventArgs e)

{

string fname;

file1.ShowDialog();

fname = file1.FileName;

pct.Image = Image.FromFile(fname);

lb.Text = String.Format(fname);

}

private void BtnSave\_Click(object sender, EventArgs e)

{

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

string savePath = saveFileDialog1.FileName;

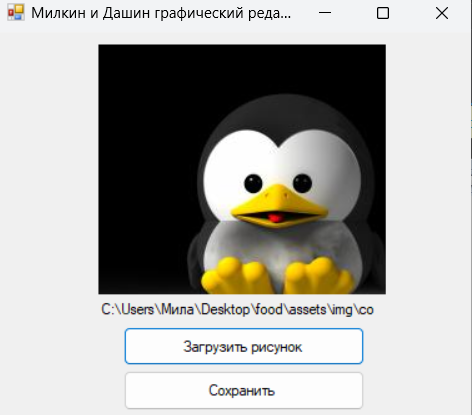
pct.Image.Save(savePath, System.Drawing.Imaging.ImageFormat.Jpeg);

}

}

}

}



**Практическая работа 6.**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net;

namespace Pr6.1

{

internal class Program

{

static async Task Main(string[] args)

{

string filePath = @"D:\index.html";

string responseText;

try

{

using (var reader = new StreamReader(filePath))

{

responseText = await reader.ReadToEndAsync();

}

Console.WriteLine("index.html загружен:");

Console.WriteLine(responseText);

}

catch (Exception ex)

{

Console.WriteLine($"ERROR: {ex.Message}");

return;

}

HttpListener listener = new HttpListener();

listener.Prefixes.Add("http://localhost:8080/");

listener.Start();

Console.WriteLine("Сервер запущен. Ожидание запросов...");

while (true)

{

try

{

var context = await listener.GetContextAsync();

var request = context.Request;

var response = context.Response;

Console.WriteLine($"Получен запрос: {request.Url}");

byte[] buffer = Encoding.UTF8.GetBytes(responseText);

response.ContentLength64 = buffer.Length;

using (var output = response.OutputStream)

{

await output.WriteAsync(buffer, 0, buffer.Length);

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при обработке запроса: {ex.Message}");

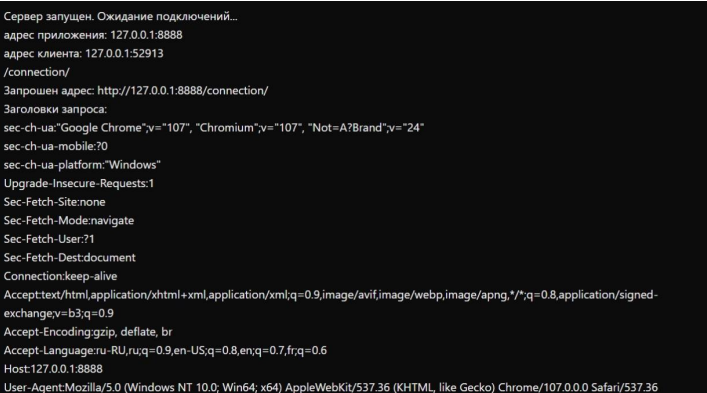
}

}

}

}

}



using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

namespace Pr6.2

{

internal class Program

{

static async Task Main(string[] args)

{

string logFilePath = @"D:\request\_log.txt";

HttpListener listener = new HttpListener();

listener.Prefixes.Add("http://127.0.0.1:8888/connection/");

listener.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений.... адрес приложения: 127.0.0.1:8888");

while (true)

{

try

{

var context = await listener.GetContextAsync();

var request = context.Request;

var response = context.Response;

string clientAddress = request.RemoteEndPoint.ToString();

string requestUrl = request.Url.ToString();

string headers = GetRequestHeaders(request);

string logData = $"адрес клиента: {clientAddress}\n" +

$"{requestUrl}\n" +

$"Заголовки запроса:\n{headers}\n";

Console.WriteLine(logData);

File.AppendAllText(logFilePath, logData);

byte[] buffer = Encoding.UTF8.GetBytes("Request received");

response.ContentLength64 = buffer.Length;

using (var output = response.OutputStream)

{

await output.WriteAsync(buffer, 0, buffer.Length);

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при обработке запроса: {ex.Message}");

}

}

}

private static string GetRequestHeaders(HttpListenerRequest request)

{

var headers = new StringBuilder();

foreach (string key in request.Headers)

{

headers.AppendLine($"{key}: {request.Headers[key]}");

}

return headers.ToString();

}

}

}

**Практическая работа 7.**

using System.Net;

using System.Net.Sockets;

ServerObject server = new ServerObject();

await server.ListenAsync();

class ServerObject

{

TcpListener tcpListener = new TcpListener(IPAddress.Any, 8888);

List<ClientObject> clients = new List<ClientObject>();

protected internal void RemoveConnection(string id)

{

ClientObject? client = clients.FirstOrDefault(c => c.Id == id);

if (client != null) clients.Remove(client);

client?.Close();

}

protected internal async Task ListenAsync()

{

try

{

tcpListener.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений...");

while (true)

{

TcpClient tcpClient = await tcpListener.AcceptTcpClientAsync();

ClientObject clientObject = new ClientObject(tcpClient, this);

clients.Add(clientObject);

Task.Run(clientObject.ProcessAsync);

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

Disconnect();

}

}

protected internal async Task BroadcastMessageAsync(string message, string id)

{

foreach (var client in clients)

{

if (client.Id != id)

{

await client.Writer.WriteLineAsync(message);

await client.Writer.FlushAsync();

}

}

}

protected internal void Disconnect()

{

foreach (var client in clients)

{

client.Close();

}

tcpListener.Stop();

}

}

class ClientObject

{

protected internal string Id { get; } = Guid.NewGuid().ToString();

protected internal StreamWriter Writer { get; }

protected internal StreamReader Reader { get; }

TcpClient client;

ServerObject server;

public ClientObject(TcpClient tcpClient, ServerObject serverObject)

{

client = tcpClient;

server = serverObject;

var stream = client.GetStream();

Reader = new StreamReader(stream);

Writer = new StreamWriter(stream);

}

public async Task ProcessAsync()

{

try

{

string? userName = await Reader.ReadLineAsync();

string? message = $"{userName} вошел в чат";

await server.BroadcastMessageAsync(message, Id);

Console.WriteLine(message);

while (true)

{

try

{

message = await Reader.ReadLineAsync();

if (message == null) continue;

message = $"{userName}: {message}";

Console.WriteLine(message);

await server.BroadcastMessageAsync(message, Id);

}

catch

{

message = $"{userName} покинул чат";

Console.WriteLine(message);

await server.BroadcastMessageAsync(message, Id);

break;

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

server.RemoveConnection(Id);

}

}

protected internal void Close()

{

Writer.Close();

Reader.Close();

client.Close();

}

}

using System.Net.Sockets;

string host = "127.0.0.1";

int port = 8888;

using TcpClient client = new TcpClient();

Console.Write("Введите свое имя: ");

string? userName = Console.ReadLine();

Console.WriteLine($"Добро пожаловать, {userName}");

StreamReader? Reader = null;

StreamWriter? Writer = null;

try

{

client.Connect(host, port;

Reader = new StreamReader(client.GetStream());

Writer = new StreamWriter(client.GetStream());

if (Writer is null || Reader is null) return;

Task.Run(() => ReceiveMessageAsync(Reader));

await SendMessageAsync(Writer);

Reader = new StreamReader(client.GetStream());

Writer = new StreamWriter(client.GetStream());

if (Writer is null || Reader is null) return;

Task.Run(() => ReceiveMessageAsync(Reader));

await SendMessageAsync(Writer);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Writer?.Close();

Reader?.Close();

async Task SendMessageAsync(StreamWriter writer)

{

await writer.WriteLineAsync(userName);

await writer.FlushAsync();

Console.WriteLine("Для отправки сообщений введите сообщение и нажмите Enter");

while (true)

{

string? message = Console.ReadLine();

await writer.WriteLineAsync(message);

await writer.FlushAsync();

}

}

async Task ReceiveMessageAsync(StreamReader reader)

{

while (true)

{

try

{

string? message = await reader.ReadLineAsync();

if (string.IsNullOrEmpty(message)) continue;

PrintAsync(message);

}

catch

{

break;

}

}

}

async Task PrintAsync(string message)

{

if (OperatingSystem.IsWindows())

{

var position = Console.GetCursorPosition();

int left = position.Left;

int top = position.Top;

Console.MoveBufferArea(0, top, left, 1, 0, top + 1);

Console.SetCursorPosition(0, top);

Console.WriteLine(message);

Console.SetCursorPosition(left, top + 1);

}

else Console.WriteLine(message);

}

using System.Net.Sockets;

string host = "127.0.0.1";

int port = 8888;

using TcpClient client = new TcpClient();

Console.Write("Введите свое имя: ");

string? userName = Console.ReadLine();

Console.WriteLine($"Добро пожаловать, {userName}");

StreamReader? Reader = null;

StreamWriter? Writer = null;

try

{

client.Connect(host, port);

Reader = new StreamReader(client.GetStream());

Writer = new StreamWriter(client.GetStream());

if (Writer is null || Reader is null) return;

Task.Run(() => ReceiveMessageAsync(Reader));

await SendMessageAsync(Writer);

Reader = new StreamReader(client.GetStream());

Writer = new StreamWriter(client.GetStream());

if (Writer is null || Reader is null) return;

Task.Run(() => ReceiveMessageAsync(Reader));

await SendMessageAsync(Writer);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Writer?.Close();

Reader?.Close();

async Task SendMessageAsync(StreamWriter writer)

{

await writer.WriteLineAsync(userName);

await writer.FlushAsync();

Console.WriteLine("Для отправки сообщений введите сообщение и нажмите Enter");

while (true)

{

string? message = Console.ReadLine();

await writer.WriteLineAsync(message);

await writer.FlushAsync();

}

}

async Task ReceiveMessageAsync(StreamReader reader)

{

while (true)

{

try

{

string? message = await reader.ReadLineAsync();

if (string.IsNullOrEmpty(message)) continue;

PrintAsync(message);

}

catch

{

break;

}

}

}

async Task PrintAsync(string message)

{

if (OperatingSystem.IsWindows())

{

var position = Console.GetCursorPosition();

int left = position.Left;

int top = position.Top

Console.MoveBufferArea(0, top, left, 1, 0, top + 1);

Console.SetCursorPosition(0, top);

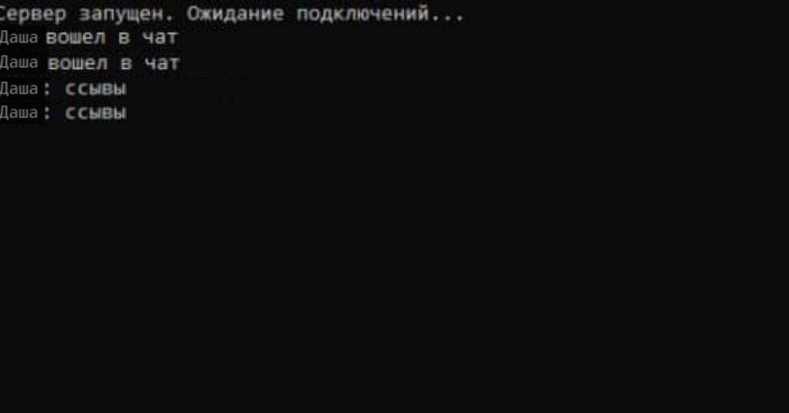
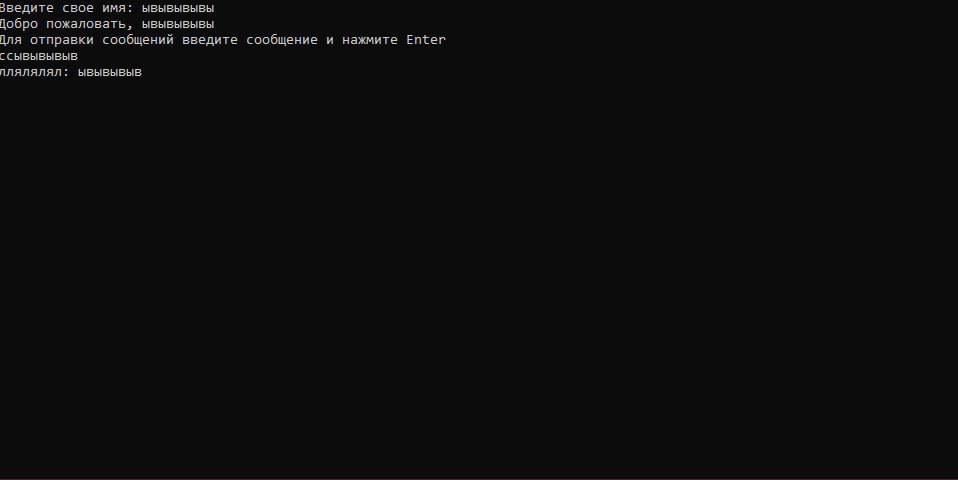
Console.WriteLine(message);

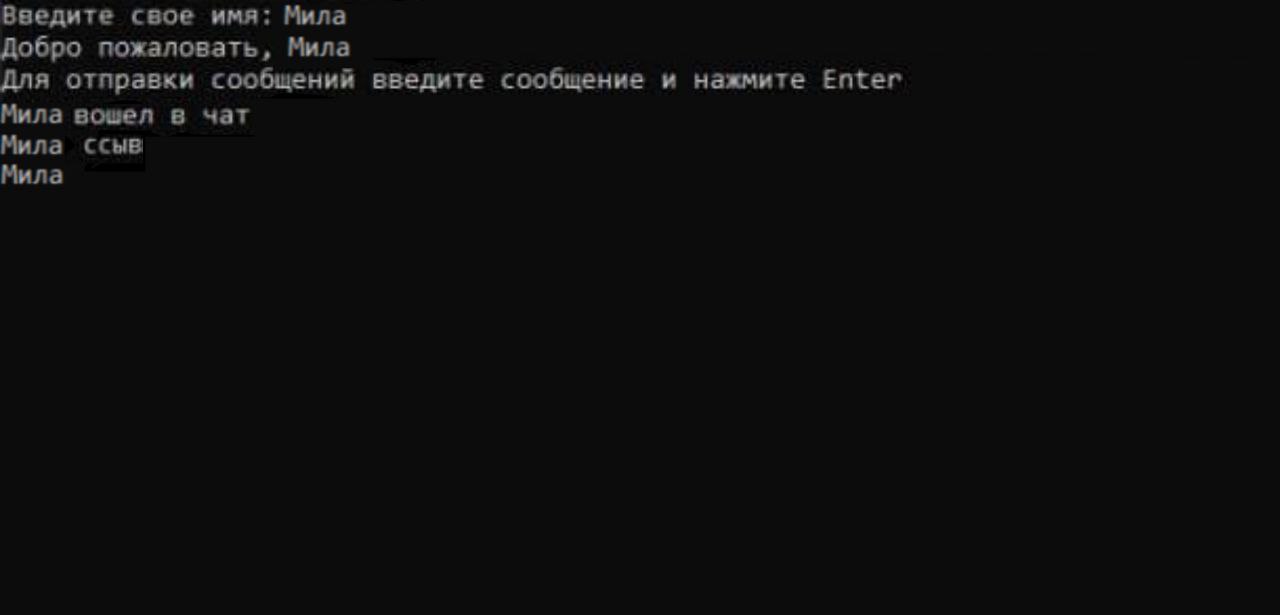
Console.SetCursorPosition(left, top + 1);

}

else Console.WriteLine(message);

}

****

****

**Практическая работа 8**

using System.Net;

using System.Net.Sockets;

using System.Text;

IPAddress localAddress = IPAddress.Parse("127.0.0.1");

Console.Write("Введите свое имя: ");

string? username = Console.ReadLine();

Console.Write("Введите порт для приема сообщений: ");

if (!int.TryParse(Console.ReadLine(), out var localPort)) return;

Console.Write("Введите порт для отправки сообщений: ");

if (!int.TryParse(Console.ReadLine(), out var remotePort)) return;

Console.WriteLine();

Task.Run(ReceiveMessageAsync);

await SendMessageAsync();

async Task SendMessageAsync()

{

using UdpClient sender = new UdpClient();

Console.WriteLine("Для отправки сообщений введите сообщение и нажмите Enter");

while (true)

{

var message = Console.ReadLine();

if (string.IsNullOrWhiteSpace(message)) break;

message = $"{username}: {message}";

byte[] data = Encoding.UTF8.GetBytes(message);

await sender.SendAsync(data, new IPEndPoint(localAddress, remotePort));

}

}

async Task ReceiveMessageAsync()

{

using UdpClient receiver = new UdpClient(localPort);

while (true)

{

var result = await receiver.ReceiveAsync();

var message = Encoding.UTF8.GetString(result.Buffer);

Print(message);

}

}

void Print(string message)

{

if (OperatingSystem.IsWindows())

{

var position = Console.GetCursorPosition();

int left = position.Left;

int top = position.Top;

Console.MoveBufferArea(0, top, left, 1, 0, top + 1);

Console.SetCursorPosition(0, top);

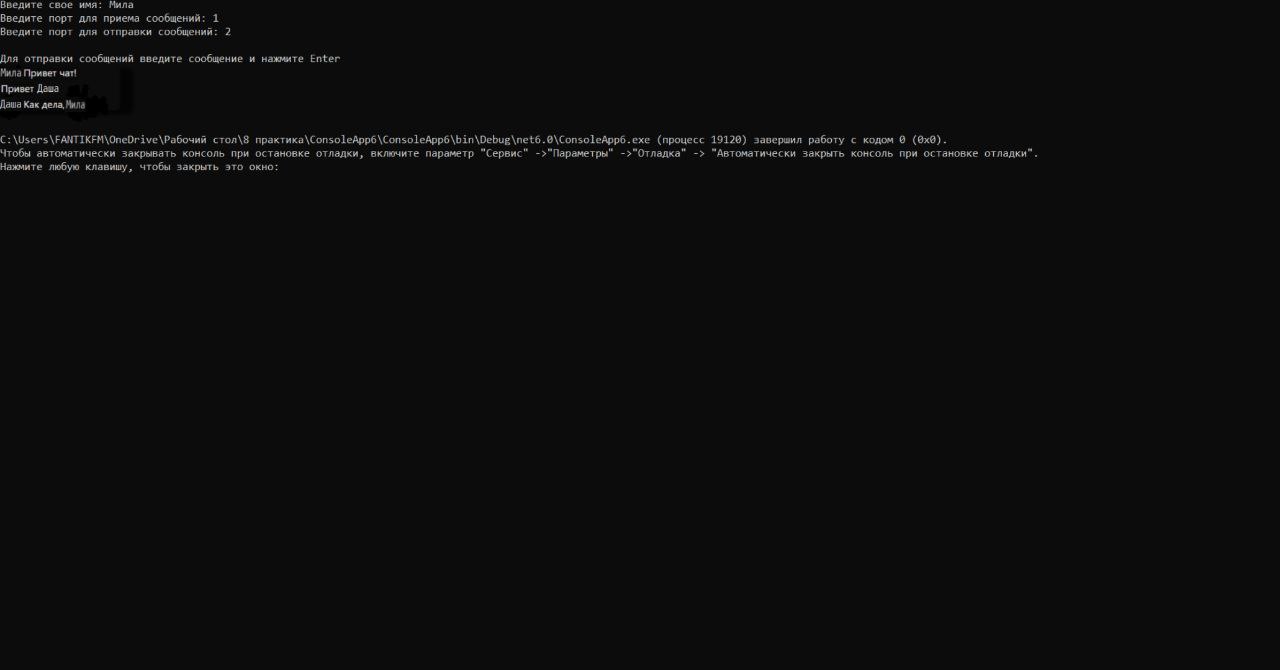
Console.WriteLine(message);

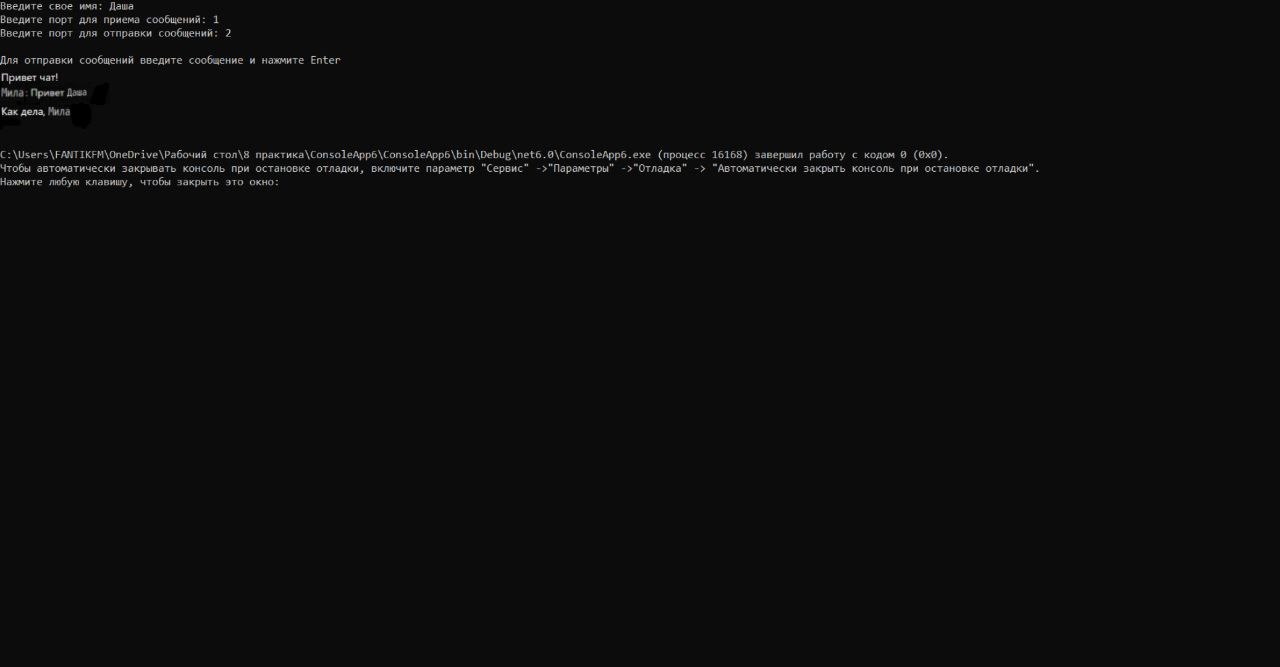
Console.SetCursorPosition(left, top + 1);

}

else Console.WriteLine(message);

}**.**





using System.Net;

using System.Net.Sockets;

using System.Text;

IPAddress localAddress = IPAddress.Parse("127.0.0.1");

Console.Write("Введите свое имя: ");

string? username = Console.ReadLine();

Console.Write("Введите порт для приема сообщений: ");

if (!int.TryParse(Console.ReadLine(), out var localPort)) return;

Console.Write("Введите порт для отправки сообщений: ");

if (!int.TryParse(Console.ReadLine(), out var remotePort)) return;

Console.WriteLine();

Task.Run(ReceiveMessageAsync);

await SendMessageAsync();

async Task SendMessageAsync()

{

using UdpClient sender = new UdpClient();

Console.WriteLine("Для отправки сообщений введите сообщение и нажмите Enter");

while (true)

{

var message = Console.ReadLine();

if (string.IsNullOrWhiteSpace(message)) break;

message = $"{username}: {message}";

byte[] data = Encoding.UTF8.GetBytes(message);

await sender.SendAsync(data, new IPEndPoint(localAddress, remotePort));

}

}

async Task ReceiveMessageAsync()

{

using UdpClient receiver = new UdpClient(localPort);

while (true)

{

var result = await receiver.ReceiveAsync();

var message = Encoding.UTF8.GetString(result.Buffer);

Print(message);

}

}

void Print(string message)

{

if (OperatingSystem.IsWindows())

{

var position = Console.GetCursorPosition();

int left = position.Left;

int top = position.Top;

Console.MoveBufferArea(0, top, left, 1, 0, top + 1);

Console.SetCursorPosition(0, top);

Console.WriteLine(message);

Console.SetCursorPosition(left, top + 1);

}

else Console.WriteLine(message);

}