МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Казанский национальный исследовательский технический

университет им. А.Н. Туполева-КАИ»

(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации

*(наименование института (факультета)*

Кафедра динамики процессов и управления

*(наименование кафедры)*

Лабораторная работа № 1

По дисциплине «Основы алгоритмизации и языки программирования»

|  |  |
| --- | --- |
| Выполнил: | обучающийся группы 4165\_ |
|  | Шестаков Михаил |

Казань 2025 г.

**Задание №1**

Разработать калькулятор на языке программирования C#, поддерживающий 5 функций: умножение (\*), деление (/), вычитание (-), сложение (\*), возведение в степень (^). Обязательно добавить обработчик ошибок. Работа калькулятора должна прекращаться при отсутствии ввода пользователем данных и нажатием им на кнопку «Enter». Добавить логирование информации и сохранение ее в отдельный файл с возможностью указать путь сохранения. Также реализовать сохранение последних 3 (успешных) операций пользователя и вывода ее в консоль при вводе команды «h».

**Листинг кода**

Program.cs:

using Calculator3;  
using NLog;  
using NLog.Extensions.Logging;  
using LogLevel = Microsoft.Extensions.Logging.LogLevel;  
  
var builder = Host.CreateApplicationBuilder(args);  
  
builder.Configuration.AddJsonFile("appsettings.json", optional: false, reloadOnChange: true);  
  
string? logRootDir = builder.Configuration["Logging:RootDir"] ?? "./logs";  
  
GlobalDiagnosticsContext.Set("LogRootDir", logRootDir);  
  
builder.Logging.ClearProviders();  
builder.Logging.SetMinimumLevel(LogLevel.**Trace**);  
builder.Logging.AddNLog();  
  
builder.Services.AddTransient<Calculator>();  
  
using var host = builder.Build();  
  
var logger = host.Services.GetRequiredService<ILogger<Program>>();  
logger.LogInformation("Application starting...");  
  
var calculator = host.Services.GetRequiredService<Calculator>();  
calculator.Run();  
  
logger.LogInformation("Application shutting down.");

Calculator.cs:

namespace Calculator3  
{  
 public class Calculator  
 {  
 private readonly ILogger<Calculator> \_logger;  
 private List<string> \_history;  
 private const int **\_maxHistory** = 3;  
  
 public Calculator(ILogger<Calculator> logger)  
 {  
 \_logger = logger;  
 \_history = [];  
 }  
  
 public void Run()  
 {  
 \_logger.LogInformation("Calculator started!");  
 Console.WriteLine("Calculator started!");  
  
 while (true)  
 {  
 Console.Write("Enter the expression: ");  
 var input = Console.ReadLine();  
  
 if (string.IsNullOrWhiteSpace(input))  
 {  
 Console.WriteLine("Task completed!");  
 \_logger.LogInformation("Task completed!");  
 break;  
 }  
  
 try  
 {  
 var parts = input.Split(' ', StringSplitOptions.**RemoveEmptyEntries**);  
  
 if (parts is ["h"])  
 {  
 for (var i = 0; i < Math.Min(\_history.Count, **\_maxHistory**); i++)  
 {  
 Console.WriteLine($"{i + 1}: {\_history[i]}");  
 }  
 continue;  
 }  
  
 if (parts.Length != 3)  
 {  
 Console.WriteLine("Error: write the full format expression: 4 \* 5");  
 \_logger.LogError("Invalid input format. User wrote: '{Input}'", input);  
 }  
 else  
 {  
 if (!double.TryParse(parts[0], out var left) ||  
 !double.TryParse(parts[2], out var right))  
 {  
 Console.WriteLine("Error: operands must be numbers.");  
 \_logger.LogError("Non-numeric operands: '{Operand1}' or '{Operand2}'", parts[0], parts[2]);  
 continue;  
 }  
  
 var op = parts[1];  
 double result;  
  
 switch (op)  
 {  
 case "+":  
 result = left + right;  
 break;  
 case "-":  
 result = left - right;  
 break;  
 case "\*":  
 result = left \* right;  
 break;  
 case "/":  
 if (right == 0)  
 {  
 Console.WriteLine("Error: Division by zero!");  
 \_logger.LogError("Division by zero attempted.");  
 continue;  
 }  
 result = left / right;  
 break;  
 case "^":  
 result = Math.Pow(left, right);  
 break;  
 default:  
 Console.WriteLine($"Error: operator '{op}' not supported!");  
 \_logger.LogError("Unsupported operator: '{Operator}'", op);  
 continue;  
 }  
  
 Console.WriteLine($"= {result}");  
 \_history.Add($"{input} = {result}");  
 if (\_history.Count > **\_maxHistory**)  
 {  
 \_history.RemoveRange(0, \_history.Count - **\_maxHistory**);  
 }  
 \_logger.LogInformation("{Expression} = {Result}", input, result);  
 }  
 }  
 catch (Exception e)  
 {  
 Console.WriteLine($"Unexpected error: {e.Message}");  
 \_logger.LogError(e, "Unexpected exception occurred");  
 }  
 }  
 }  
 }  
}

Nlog.config:

<?xml version="1.0" encoding="utf-8"?>  
  
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 throwExceptions="true">  
  
 <extensions>  
 <add assembly="NLog.DiagnosticSource" />  
 </extensions>  
  
 <variable name="DirectoryLogs" value="${configsetting:item=Logging.RootDir}" />  
 <variable name="DefaultLayout"  
 value="${longdate} ${level:uppercase=true} ${ndlc}${newline}${activity:property=TraceId} ${logger:shortName=true} ${message}${when:when='${exception}' != '':inner=${newline} ${exception:format=tostring}}"  
 />  
  
 <targets>  
  
 <target name="file" xsi:type="File"  
 layout="${DefaultLayout}"  
 fileName="${DirectoryLogs}/${shortdate}.txt"  
 maxArchiveDays="5"  
 maxArchiveFiles="4"  
 archiveNumbering="Sequence"  
 archiveAboveSize="52428800"  
 keepFileOpen="false"  
 encoding="utf-8" />  
  
 </targets>  
  
 <rules>  
 <logger name="\*" minlevel="Info" writeTo="file" />  
 </rules>  
</nlog>

Appsettings.json:

{  
 "Logging": {  
 "LogLevel": {  
 "Default": "Information",  
 "Microsoft.AspNetCore": "Warning"  
 },  
 "RootDir": "C:/.tmp/logs/study/calculator"  
 },  
 "AllowedHosts": "\*"  
}

**Объяснение работы программы**

Было разработано веб-приложение, в котором я решил применить библиотеку NLog, поскольку уже имел опыт работы с ней. Для удобного подключения использовался builder. В настройках приложения была указана директория для сохранения лог-файлов, а также создан отдельный конфигурационный файл NLog.config, где заданы минимальный уровень логирования, формат и максимальный размер файлов и другие параметры.

В основном файле запуска реализовано подключение конфигурации и самой библиотеки NLog. Отдельно был разработан модуль калькулятора, в который добавлено логирование после выполнения каждой операции. Тип вычисления определяется конструкцией switch-case.

Также реализован список, в котором сохраняются последние три успешных вычисления. Удаление лишних записей производится методом RemoveRange, который очищает элементы с индекса 0 до длины списка минус 3. Просмотреть сохранённые вычисления можно с помощью команды «h».

**Задание №2**

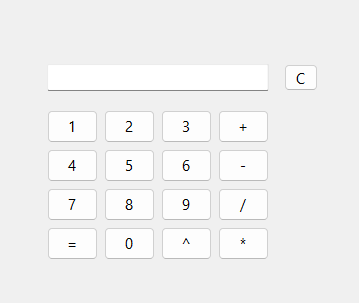
Разработать графический калькулятор на языке программирования C#, поддерживающий 5 функций: умножение (\*), деление (/), вычитание (-), сложение (\*), возведение в степень (^). Обязательно добавить обработчик ошибок. Калькулятор должен быть реализован с использованием Windows Form.

**Листинг кода**

Form1.cs:

namespace Calculator2;  
  
public partial class Form1 : Form  
{  
 public Form1()  
 {  
 InitializeComponent();  
 }  
  
 private TextBox \_input;  
  
 private void button1\_Click(object sender, EventArgs e)  
 {  
 \_input.Text += "1";  
 }  
  
 private void button8\_Click(object sender, EventArgs e)  
 {  
 \_input.Text += "2";  
 }  
  
 private void button7\_Click(object sender, EventArgs e)  
 {  
 \_input.Text += "3";  
 }  
  
 private void button6\_Click(object sender, EventArgs e)  
 {  
 \_input.Text += "4";  
 }  
  
 private void button5\_Click(object sender, EventArgs e)  
 {  
 \_input.Text += "5";  
 }  
  
 private void button4\_Click(object sender, EventArgs e)  
 {  
 \_input.Text += "6";  
 }  
  
 private void button3\_Click(object sender, EventArgs e)  
 {  
 \_input.Text += "7";  
 }  
  
 private void button2\_Click(object sender, EventArgs e)  
 {  
 \_input.Text += "8";  
 }  
  
 private void button10\_Click(object sender, EventArgs e)  
 {  
 \_input.Text += "9";  
 }  
  
 private void button9\_Click(object sender, EventArgs e)  
 {  
 if (!string.IsNullOrEmpty(\_input.Text)) \_input.Text += "0";  
 }  
  
 private void button11\_Click(object sender, EventArgs e)  
 {  
 if (!string.IsNullOrEmpty(\_input.Text)) \_input.Text += " ^ ";  
 }  
  
 private void button16\_Click(object sender, EventArgs e)  
 {  
 if (!string.IsNullOrEmpty(\_input.Text)) \_input.Text += " + ";  
 }  
  
 private void button15\_Click(object sender, EventArgs e)  
 {  
 if (!string.IsNullOrEmpty(\_input.Text)) \_input.Text += " - ";  
 }  
  
 private void button14\_Click(object sender, EventArgs e)  
 {  
 if (!string.IsNullOrEmpty(\_input.Text)) \_input.Text += " / ";  
 }  
  
 private void button13\_Click(object sender, EventArgs e)  
 {  
 if (!string.IsNullOrEmpty(\_input.Text)) \_input.Text += " \* ";  
 }  
  
 private void button12\_Click(object sender, EventArgs e)  
 {  
 try  
 {  
 var parts = \_input.Text!.Split(' ', StringSplitOptions.RemoveEmptyEntries);  
 if (parts.Length != 3)  
 {  
 \_input.Text = "Error: write the full format expression: 4 \* 5";  
 }  
 else  
 {  
 \_input.Text = parts[1] switch  
 {  
 "+" => $"{double.Parse(parts[0]) + double.Parse(parts[2])}",  
 "-" => $"{double.Parse(parts[0]) - double.Parse(parts[2])}",  
 "\*" => $"{double.Parse(parts[0]) \* double.Parse(parts[2])}",  
 "/" => $"{double.Parse(parts[0]) / double.Parse(parts[2])}",  
 "^" => $"{Math.Pow(double.Parse(parts[0]), double.Parse(parts[2]))}",  
 \_ => $"Error: not find {parts[1]}!",  
 };  
 }  
 }  
 catch (Exception err)  
 {  
 \_input.Text = err.Message;  
 }  
 }  
  
 private void button17\_Click(object sender, EventArgs e)  
 {  
 \_input.Text = "";  
 }  
}

Form1.cs (визуал):



Program.cs:

namespace Calculator2;  
  
static class Program  
{  
 */// <summary>  
 /// The main entry point for the application.  
 /// </summary>* [STAThread]  
 static void Main()  
 {  
 *// To customize application configuration such as set high DPI settings or default font,  
 // see https://aka.ms/applicationconfiguration.* ApplicationConfiguration.Initialize();  
 Application.Run(new Form1());  
 }  
}

**Объяснение работы программы**

Было создано приложение на основе шаблона Windows Forms App. В дизайнере форм был разработан интерфейс калькулятора: добавлены кнопки для ввода цифр, арифметических операций, очистки экрана, а также текстовое поле для отображения вводимых данных, сообщений об ошибках и результата вычислений.

Для каждой кнопки было реализовано событие Click. При нажатии на кнопки с цифрами или операциями (кроме кнопки «=») соответствующие символы добавляются в текстовое поле. При нажатии на кнопку очистки («C») содержимое текстового поля сбрасывается — оно заменяется на пустую строку.

Основные вычисления выполняются при нажатии на кнопку «=». В этот момент содержимое текстового поля разбивается по пробелам на отдельные части (операнды и операторы). Затем на основе этих частей формируется выражение, которое вычисляется средствами C#. Полученный результат заменяет текущее содержимое текстового поля.