Лабораторная работа № 9

Тема: Логирование в Windows Forms.

Цель: закрепить теоретические знания и применить их на практике.

Задание:

Вариант №1: Приложение «Архиватор-загрузчик»

Сценарий: Пользователь вводит путь к файлу для его «обработки» (чтения данных).

Задание 1: Локальная защита (Try-Catch)
Реализуйте кнопку «Прочитать файл».
Оберните код чтения файла (File.ReadAllText) в try-catch.
Обработайте специфическое исключение FileNotFoundException (файл не найден) и общее Exception.
Выведите пользователю MessageBox с понятным описанием проблемы.

Задание 2: Глобальный перехват
В файле Program.cs настройте подписку на Application.ThreadException.
Добавьте на форму «секретную» кнопку, которая намеренно вызывает ошибку (например, обращение к пустому объекту string s = null; s.ToLower();), не обернутую в try-catch.
Убедитесь, что глобальный обработчик ловит эту ошибку и не дает программе закрыться.

Задание 3: Журналирование (Logging)
Создайте метод WriteLog(string message).
При каждой ошибке (и в try-catch, и в глобальном обработчике) записывайте в файл session.log: [Дата и Время] [Тип ошибки] [Сообщение].
Добавьте запись об успешном запуске программы при старте формы.

Вариант №2: Приложение «Математический анализатор»

Сценарий: Пользователь вводит числа для сложных вычислений.

Задание 1: Локальная защита (Try-Catch)
Реализуйте кнопку «Вычислить результат».
Оберните парсинг чисел (int.Parse) и операцию деления в try-catch.
Обработайте FormatException (введены буквы вместо цифр) и DivideByZeroException.
Выведите информативное окно MessageBox с иконкой Warning.

Задание 2: Глобальный перехват
В файле Program.cs настройте обработку Application.ThreadException.
Добавьте на форму кнопку, которая вызывает «непредвиденную» ошибку (например, выход за границы массива int[] a = {1}; int b = a[10];), без локального try-catch.
Глобальный обработчик должен вывести сообщение: «Произошел критический сбой, детали в логе».

Задание 3: Журналирование (Logging)
Реализуйте запись всех исключений в файл errors.txt.
Используйте File.AppendAllText, чтобы данные не затирались.
Каждая запись должна содержать: Метку времени, Название исключения (ex.GetType().Name) и Текст ошибки.
Записывайте в лог факт успешного выполнения расчета, если ошибок не возникло.

Отчет должен содержать (см. образец):

- номер и тему лабораторной работы;
- фамилию, номер группы студента и вариант задания;
- скриншоты окна Visual Studio с исходным кодом программы и комментариями;
- скриншоты с результатами выполнения программ;
- пояснения, если необходимо;

- выводы.
Отчеты в формате **pdf** отправлять на email: **colledge20education23@gmail.com**

Шпаргалка на тему "Обработка исключений и логирование"

1. Работа с файлами (System.IO)
Для записи лога используйте File.AppendAllText. Он сам открывает и закрывает файл.

```
// Добавить строку в файл (создаст файл, если его нет)
File.AppendAllText("log.txt", "Текст ошибки" + Environment.NewLine);
```

2. Получение данных об ошибке
В блоке catch (Exception ex), объект ex — ваш главный помощник:

- ex.Message — краткое описание (например, "Деление на ноль").
- ex.GetType().Name — техническое название ошибки (например, DivideByZeroException).
- ex.StackTrace — подробный путь (в какой строке кода всё случилось).

3. Форматирование даты
Чтобы лог был красивым, используйте DateTime.Now:

```
string time = DateTime.Now.ToString("dd.MM.yyyy HH:mm:ss");
```

Полный пример готового приложения.
Этот код представляет собой простую форму с одной кнопкой. В нем реализованы все три уровня из лабораторной: локальный перехват, логирование в файл и глобальная защита.
Шаг 1: Файл Program.cs (Глобальная защита)

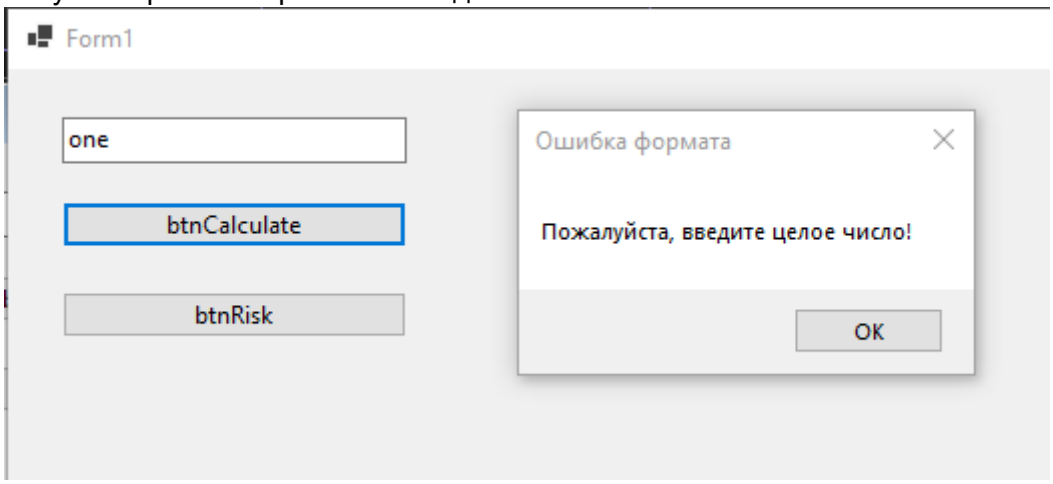
```
internal static class Program
{
    [STAThread]
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);

        // ГЛОБАЛЬНЫЙ ПЕРЕХВАТ: Подписка на ошибки в интерфейсе
        Application.ThreadException += new
ThreadExceptionEventHandler(GlobalThreadExceptionHandler);
        Application.SetUnhandledExceptionMode(UnhandledExceptionMode.CatchException);

        Application.Run(new Form1());
    }

    static void GlobalThreadExceptionHandler(object sender, ThreadExceptionEventArgs e)
    {
        // Записываем в лог, что мы "проспали" ошибку в коде
        string logMsg = $"[КРИТИЧЕСКИЙ СБОЙ] {DateTime.Now}: {e.Exception.Message}\n";
        File.AppendAllText("app_errors.log", logMsg);

        MessageBox.Show("Произошла непредвиденная ошибка. Мы сохранили детали в лог-файл.",
                        "Упс!", MessageBoxButtons.OK, MessageBoxIcon.Stop);
    }
}
```

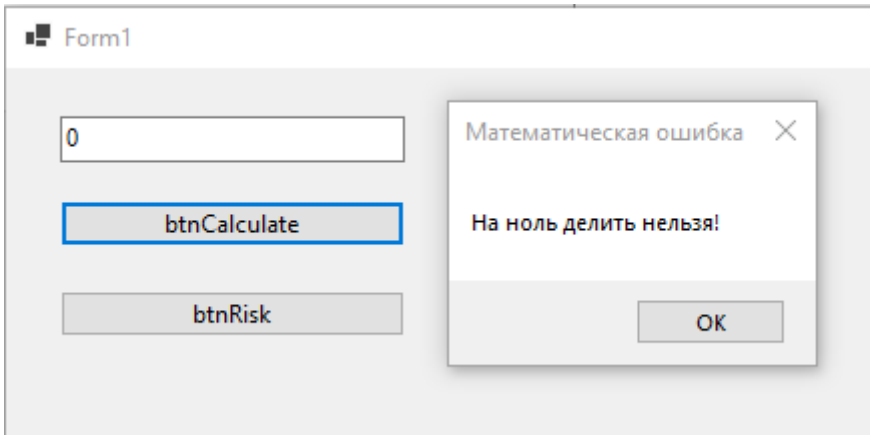
Шаг 2: Файл Form1.cs (Логика и Локальный перехват)
Пример кода для формы, где есть текстовое поле и кнопки: `btnCalculate` и `btnRisk`.

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    // Метод для записи в лог (чтобы не дублировать код)
    private void Log(string type, string message)
    {
```

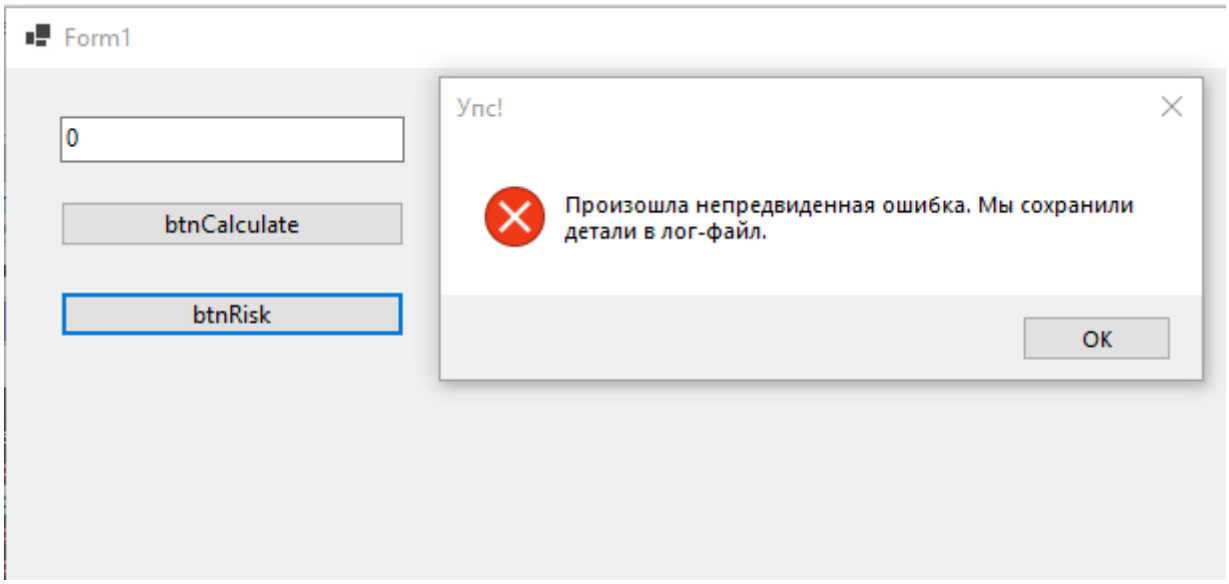
```
            string line = $"[{DateTime.Now:G}] [{type}]
{message}{Environment.NewLine}";
            File.AppendAllText("app_errors.log", line);
        }

        private void btnCalculate_Click(object sender, EventArgs e)
        {
            try
            {
                // Опасный код: парсинг и деление
                int x = int.Parse(txtInput.Text);
                int result = 100 / x;

                MessageBox.Show($"Результат: {result}");
                Log("INFO", "Успешное вычисление");
            }
            catch (FormatException ex)
            {
                // Локальный перехват ошибки ввода
                Log("WARNING", "Пользователь ввел не число");
                MessageBox.Show("Пожалуйста, введите целое число!", "Ошибка
формата");
            }
            catch (DivideByZeroException ex)
            {
                // Локальный перехват деления на ноль
                Log("ERROR", "Попытка деления на ноль");
                MessageBox.Show("На ноль делить нельзя!", "Математическая ошибка");
            }
            catch (Exception ex)
            {
                // Любая другая непредвиденная ошибка
                Log("FATAL", ex.Message);
                throw; // Пробрасываем выше, чтобы сработал глобальный обработчик
            }
        }

        private void btnRisk_Click(object sender, EventArgs e)
        {
            // Кнопка для проверки ГЛОБАЛЬНОЙ обработки
            // Здесь нет try-catch, поэтому ошибку поймает Program.cs
            string s = null;
            int len = s.Length;
        }
    }
```

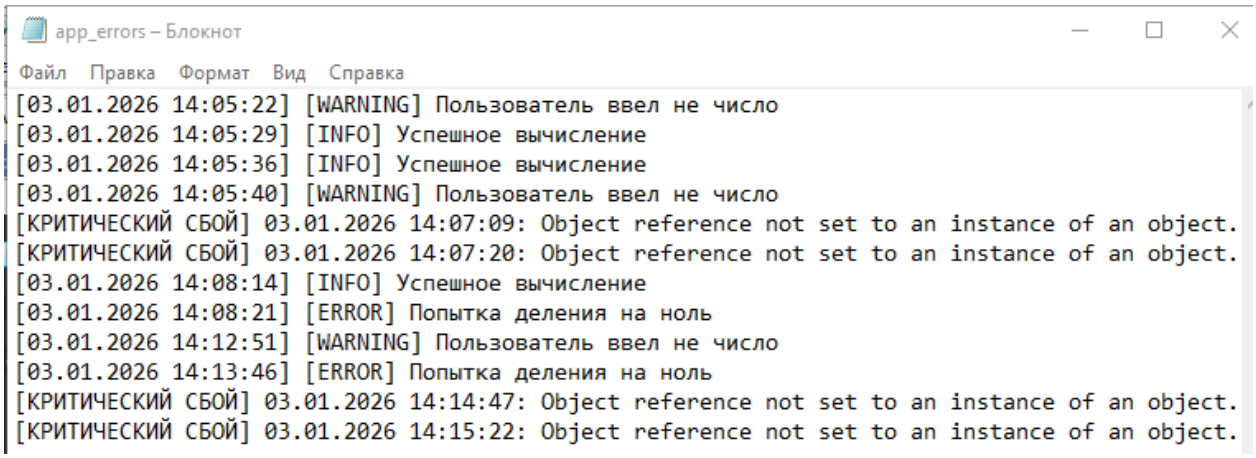
Результат работы в режиме отладки:




Form1
0
btnCalculate
btnRisk
Математическая ошибка
На ноль делить нельзя!
OK

.exe файл:


Form1
0
btnCalculate
btnRisk
Упс!
Произошла непредвиденная ошибка. Мы сохранили детали в лог-файл.
OK


app_errors – Блокнот
Файл Правка Формат Вид Справка
[03.01.2026 14:05:22] [WARNING] Пользователь ввел не число
[03.01.2026 14:05:29] [INFO] Успешное вычисление
[03.01.2026 14:05:36] [INFO] Успешное вычисление
[03.01.2026 14:05:40] [WARNING] Пользователь ввел не число
[КРИТИЧЕСКИЙ СБОЙ] 03.01.2026 14:07:09: Object reference not set to an instance of an object.
[КРИТИЧЕСКИЙ СБОЙ] 03.01.2026 14:07:20: Object reference not set to an instance of an object.
[03.01.2026 14:08:14] [INFO] Успешное вычисление
[03.01.2026 14:08:21] [ERROR] Попытка деления на ноль
[03.01.2026 14:12:51] [WARNING] Пользователь ввел не число
[03.01.2026 14:13:46] [ERROR] Попытка деления на ноль
[КРИТИЧЕСКИЙ СБОЙ] 03.01.2026 14:14:47: Object reference not set to an instance of an object.
[КРИТИЧЕСКИЙ СБОЙ] 03.01.2026 14:15:22: Object reference not set to an instance of an object.