Министерство образования Российской Федерации Пензенский государственный университет

Кафедра «Математическое обеспечение и применение ЭВМ»

**ОТЧЕТ**

по лабораторной работе № 6

по курсу «ООП» на тему «Многопоточность и асинхронное программирование»

Выполнил:

студент группы 23ВП1

Рейимов М.

Принял:

к.т.н. Афонин А.Ю.

Пенза 2025

### Название

Многопоточность и асинхронное программирование.

### Цель работы

Научиться работать с многопоточностью и асинхронным программированием.

### Лабораторное задание

Реализовать программу с вызовом асинхронных методов. Реализовать три асинхронных метода: расчет функции «Поиск минимального значения на векторе», вывод системного времени и имитацию процесса загрузки. Пользователь вызывает методы из оконной формы. Работа методов выводится в компоненты оконной формы.

### Листинг

### Файл Form1.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab6

{

public partial class Form1: Form

{

public Form1()

{

InitializeComponent();

}

// Поиск минимума

private async void findMinButton\_Click(object sender, EventArgs e)

{

Random rnd = new Random();

List<int> numbers = new List<int>();

int arraySize = 1000;

for (int i = 0; i < arraySize; i++)

numbers.Add(rnd.Next(-1000, 10000));

int min = await FindMinAsync(numbers);

textBoxMin.Text = $"Минимум: {min}";

}

private async Task<int> FindMinAsync(List<int> numbers)

{

return await Task.Run(() =>

{

Thread.Sleep(1500);

return numbers.Min();

});

}

// Вывод системного времени

private CancellationTokenSource timeTokenSource;

private async void showTimeButton\_Click(object sender, EventArgs e)

{

if (timeTokenSource != null)

{

timeTokenSource.Cancel();

timeTokenSource = null;

showTimeButton.Text = "Показать время";

return;

}

timeTokenSource = new CancellationTokenSource();

showTimeButton.Text = "Остановить время";

await ShowTimeAsync(timeTokenSource.Token);

}

private async Task ShowTimeAsync(CancellationToken token)

{

while (!token.IsCancellationRequested)

{

textBoxTime.Invoke((MethodInvoker)(() =>

{

textBoxTime.Text = DateTime.Now.ToLongTimeString();

}));

await Task.Delay(1000);

}

}

// Имитация загрузки

private async void loadButton\_Click(object sender, EventArgs e)

{

await SimulateLoadingAsync();

}

private async Task SimulateLoadingAsync()

{

progressBar1.Value = 0;

for (int i = 0; i <= 100; i += 10)

{

progressBar1.Value = i;

await Task.Delay(300);

}

MessageBox.Show("Загрузка завершена!", "Готово", MessageBoxButtons.OK, MessageBoxIcon.Information);

progressBar1.Value = 0;

}

}

}

### Пояснительный текст к программе

Данная программа на C# (WinForms) демонстрирует работу с асинхронными задачами:

* **Поиск минимума:** При нажатии на кнопку «Найти минимум» создаётся список из 1000 случайных чисел. Минимум находится в фоновом потоке, чтобы не "зависал" интерфейс.
* **Вывод системного времени:** При нажатии на кнопку «Показать время» в textBoxTime каждую секунду выводится системное время. Повторное нажатие останавливает отображение. Для управления используется CancellationToken.
* **Имитация загрузки:** При нажатии на кнопку «Загрузить» прогресс-бар постепенно заполняется (каждые 300 мс). После завершения отображается сообщение и прогресс-бар сбрасывается.

### Результаты работы программы

Результаты работы программы показаны на рисунках 1-4.

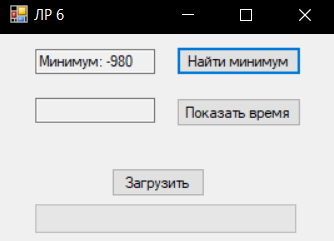


Рисунок 1 – Поиск минимального значения на векторе

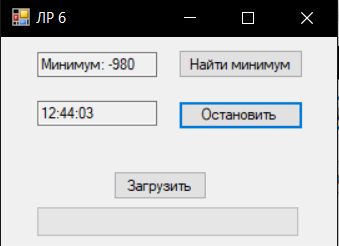


Рисунок 2 – Вывод системного времени

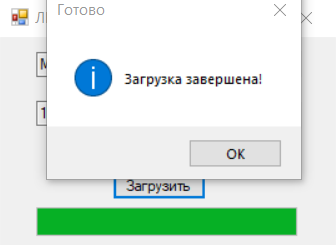


Рисунок 3 – Имитация загрузки (после завершения)

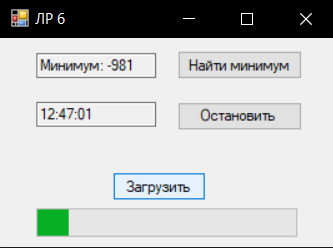


Рисунок 4 – Одновременное выполнение функций

### Диаграмма вариантов использования

### C:\Users\Влад\Downloads\2025-04-14_16-50-56.png

Рисунок 5 - Диаграмма вариантов использования

### Выводы

В ходе выполнения лабораторной работы были созданы 3 асинхронные функции: поиск минимального значения на векторе, вывод системного времени и имитация загрузки. Все они выполняются в разных потоках, независимо друг от друга.