Лабораторная работа №22
Асинхронное программирование

1 Цель работы

1.1 Научиться реализовывать и запускать асинхронные операции на C#.
1.2 Научиться выполнять вычисления, используя асинхронные операции.
1.3 Научиться выполнять ввод и вывод данных, используя асинхронные операции.

2 Литература

2.1 https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/async/
2.2 https://metanit.com/sharp/tutorial/ - гл. 16

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).
3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Последовательное и параллельное выполнение

5.1.1 Создать в консольном приложении метод для вычисления a в степени x (a – любое, x – целое), используя стандартную математическую функцию.

Метод должен выводить результат вычислений на экран в следующем виде (вместо переменных должны быть значения переменных):

a^x=result

5.1.2 Вызвать метод для вычисления a^x, используя последовательный асинхронный вызов для трех различных наборов параметров методов.

5.1.3 Вызвать метод для вычисления a^x, используя параллельный асинхронный вызов для трех различных наборов параметров методов.

5.2 Реализация асинхронной записи в файл

5.2.1 Создать в консольном приложении асинхронный метод, выполняющий запись N случайных целых чисел в файл (параметры метода: имя файла и количество чисел N).

N должно быть большим (100 000 – 500 000).

Образец заполнения файла:

Число 1: 156
Число 2: 269
…
Число N: 1278

Для записи использовать метод WriteLineAsync:

```
using (StreamWriter writer = new(fileName, false))
{
    await writer.WriteLineAsync(текст);
}
```

В начале метода вывести сообщение *«запись в файл fileName начата»*, в конце метода – *«запись в файл fileName закончена»*.

5.2.2 Вызвать метод записи файла в асинхронном режиме и после него вывести строку «Конец программы».

5.3 Реализация асинхронного чтения из файла

5.3.1 Создать в консольном приложении асинхронный метод, выполняющий построчное чтение данных из файла (параметры метода: имя файла).

Для построчного чтения можно использовать метод ReadLineAsync:

```
using (StreamReader reader = new(fileName))
{
    string line;
    while ((line = await reader.ReadLineAsync()) != null)
    {
        // ...
    }
}
```

В начале метода вывести сообщение *«чтение из файла fileName начато»*, в конце метода – *«чтение из файла fileName закончено»*.

Каждую строку из файла выводить в следующем виде:

имя файла: считанная строка

5.3.2 Вызвать метод чтения файла в асинхронном параллельном режиме для трех разных объемных файлов (можно использовать файлы, созданные в п.5.2). Полученные данные вывести на экран.

5.4 Комбинирование выполнения задач I/O и CPU

5.4.1 Создать в консольном приложении метод для вычисления хэша указанного набора данных:

```
string CalculateHash(byte[] data)
{
    using var sha256 = SHA256.Create();
    return Convert.ToHexString(sha256.ComputeHash(data));
}
```

5.4.2 Создать в консольном приложении асинхронный метод для вычисления и возврата хэша указанного в параметрах файла:

- считать данные файла, используя File.ReadAllBytesAsync,
- запустить в отдельном потоке вычисление и возврат хэша данных файла, используя CalculateHash.

Для информативности метод должен возвращать данные в следующем формате:

имяФайла : хэш

5.4.3 Протестировать метод вычисления хэша файла, запустив его в цикле для каждого файла определенной директории (например, с заданиями по МДК.01.01).

5.5 Отмена длительной операции

5.5.1 Реализовать отмену длительной асинхронной операции (чтения файла или вычисления хэша файла) по истечении определенного времени.

5.5.2 Реализовать перехват исключений при работе асинхронных методов.

6 Порядок выполнения работы

6.1 Запустить MS Visual Studio и создать консольное приложение C#. Выполнить все задания из п.5 в одном решении LabWork22.

При разработке считать, что пользователь ввел данные требуемого типа, остальные возможные ошибки обрабатывать.

При выполнении заданий использовать минимально возможное количество команд и переменных и выполнять форматирование и рефакторинг кода.

6.2 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 Какие ключевые слова используются в C# для работы с асинхронными вызовами?

8.2 Какие типы возврата могут быть у асинхронных методов и для чего предназначен каждый из типов?

8.3 Как вызвать метод в асинхронном режиме?

8.4 Как указать, что в методе могут быть асинхронные вызовы?

8.5 Как обработать исключения, возникшие в асинхронных вызовах