# Цель и задачи лабораторной работы

**Цель:** научиться использовать механизм ожидания завершения работы асинхронного метода с использованием типа IAsyncResult и тайм-аута.

**Задачи:**

1. Научиться использовать механизм тайм-аутов;
2. Научиться выводить информацию о ходе выполнения асинхронного метода;
3. Научиться отслеживать выполнение асинхронного метода.

# Реализация индивидуального задания

Согласно варианту задания, требуется реализовать приложение с использованием тайм-аута (модифицировать приложение из второй лабораторной). Дополнительно реализовать механизм вывода информации в консоль о ходе решения задачи асинхронным методом). тип делегата со следующей сигнатурой: ***Action<Func<int>, List<float>>***

В моём варианте делегат задаётся лямбда-выражением и принимает два параметра: строку и символ.

Метод возвращает логическое значение, указывающее существует ли заданный символ в строке.

## Листинг программного кода

***namespace*** lab3

{

***class*** Program

    {

**const** **short** DELAY\_ms **=** 500;

**const** **short** TIMESPAN\_s **=** 5;

**public** **static** **string**[] SplitString(**string** input, **int** countParts)

        {

**int** partLength **=** input.Length **/** (countParts **-** 1);

**string**[] parts **=** **new** **string**[countParts];

**int** currentPart **=** 0;

*for* (**int** i **=** 0; i **<** input.Length; i **+=** partLength)

            {

**int** charsCount **=** 0;

**int** currentIndex **=** i;

*// Подсчет реальных символов с учетом Unicode*

*while* (charsCount **<** partLength **&&** currentIndex **<** input.Length)

                {

**char** c **=** input[currentIndex];

                    charsCount **+=** **char**.IsSurrogate(c) **?** 2 **:** 1;

                    currentIndex**++**;

                }

                parts[currentPart] **=** input.Substring(i, currentIndex **-** i);

                currentPart**++**;

            }

*return* parts;

        }

**public** ***delegate*** Task<**bool**?> AsyncDelegate(**string** str, **char** symbol, IProgress<**string**> progress, CancellationToken cancellationToken);

**public** **static** **async** Task Main(**string**[] args)

        {

**const** **string** text **=** "asjhdkajsdjkskjfldfghjghgasdhsadugkjashdjkashdjkhgdsagdhasfdgasdgsajhdtashdjashdsajdhjas;das;jkdasjkdpkjashdjklashdjhasgdhjagsdjhas;dlkjv";

*// u, v*

**const** **char** symbol **=** ']';

**const** **int** countParts **=** 10;

            TimeSpan overallTimeout **=** TimeSpan.FromSeconds(TIMESPAN\_s);

**string**[] parts **=** SplitString(text, countParts);

            AsyncDelegate lambdaDelegate **=** **async** (str, symbol, progress, cancelToken) **=>**

            {

                progress**?**.Report("Start");

**char**[] charArray **=** str.ToCharArray();

**int** strLength **=** str.Length;

*for* (**int** i **=** 0; i **<** strLength; i**++**)

                {

                    cancelToken.ThrowIfCancellationRequested();

                    progress**?**.Report($"Шаг {i **+** 1} из {strLength} возможных");

*if* (charArray[i] **==** symbol)

                    {

                        progress**?**.Report("Найдено!");

*return* true;

                    }

**await** Task.Delay(DELAY\_ms, cancelToken);

                }

                progress**?**.Report("Finish");

*return* false;

            };

***var*** cancellationSource **=** **new** CancellationTokenSource();

            cancellationSource.CancelAfter(overallTimeout);

***var*** tasks **=** **new** List<Task<**bool**?>>();

*for* (**int** i **=** 0; i **<** parts.Length; i**++**)

            {

**int** processId **=** i **+** 1;

***var*** progress **=** **new** Progress<**string**>(message **=>**

                {

                    Console.WriteLine($"[{DateTime.Now**:**HH**:**mm**:**ss.fff}] [Процесс {processId}] {message}");

                });

                tasks.Add(lambdaDelegate(parts[i], symbol, progress, cancellationSource.Token));

            }

*// Найден ли символ*

**bool** found **=** false;

*// Флаг для отслеживания тайм-аута*

**bool** timeoutOccurred **=** false;

*while* (tasks.Count **>** 0)

            {

*try*

                {

***var*** result **=** **await** Task.WhenAny(tasks);

*if* ((**bool**)**await** result)

                    {

                        cancellationSource.Cancel();

                        found **=** true;

*break*;

                    }

                }

*catch* (OperationCanceledException) *when* (cancellationSource.Token.IsCancellationRequested)

                {

                    timeoutOccurred **=** true;

                    Console.WriteLine($"\nВнимание: Превышено общее время выполнения ({overallTimeout.TotalSeconds} сек.).");

*break*;

                }

            }

*// (found, timeoutOccurred) = await CheckTasks(overallTimeout, cancellationSource, tasks, found, timeoutOccurred);*

**const** **string** HAPPYMESSAGE **=** "GG!";

**const** **string** SADMESSAGE **=** "Not GG...";

*// Вывод финального сообщения*

**string** endmessage **=** found **?** HAPPYMESSAGE **:** (timeoutOccurred **?** SADMESSAGE **+** " (Тайм-аут)" **:** SADMESSAGE);

            Console.WriteLine(endmessage);

        }

    }

}

## Описание кода

**Ключевые изменения:**

* Добавлен общий лимит времени выполнения (TIMESPAN\_s = 5 сек)
* Обработка отмены по тайм-ауту через CancelAfter()
* Улучшенная обработка исключений с отслеживанием причин отмены

**Основные механизмы**

**1. Система тайм-аута**

TimeSpan overallTimeout = TimeSpan.FromSeconds(TIMESPAN\_s);

cancellationSource.CancelAfter(overallTimeout);

* Автоматическая отмена всех задач через 5 секунд
* Централизованное управление временем выполнения

**2. Обработка прерываний**

catch (OperationCanceledException) when (cancellationSource.Token.IsCancellationRequested)

{

timeoutOccurred = true;

Console.WriteLine($"\nВнимание: Превышено общее время выполнения...");

}

* Различает отмену по тайм-ауту и ручную отмену
* Устанавливает флаг timeoutOccurred для финального решения

**3. Улучшенный вывод результатов**

Три возможных исхода:

* **"GG!"** - символ найден
* **"Not GG..."** - символ не найден
* **"Not GG... (Тайм-аут)"** - превышено время выполнения

## Результат работы программы

[19:13:17.112] [Процесс 5] Start

[19:13:17.064] [Процесс 1] Start

[19:13:17.068] [Процесс 1] Шаг 1 из 15 возможных

[19:13:17.082] [Процесс 2] Start

[19:13:17.088] [Процесс 2] Шаг 1 из 15 возможных

[19:13:17.094] [Процесс 3] Start

[19:13:17.099] [Процесс 3] Шаг 1 из 15 возможных

[19:13:17.103] [Процесс 4] Start

[19:13:17.157] [Процесс 10] Шаг 1 из 2 возможных

[19:13:17.116] [Процесс 5] Шаг 1 из 15 возможных

[19:13:17.121] [Процесс 6] Start

[19:13:17.125] [Процесс 6] Шаг 1 из 15 возможных

[19:13:17.157] [Процесс 7] Шаг 1 из 15 возможных

[19:13:17.157] [Процесс 8] Start

[19:13:17.157] [Процесс 7] Start

[19:13:17.157] [Процесс 8] Шаг 1 из 15 возможных

[19:13:17.157] [Процесс 9] Start

[19:13:17.157] [Процесс 9] Шаг 1 из 15 возможных

[19:13:17.157] [Процесс 10] Start

[19:13:17.108] [Процесс 4] Шаг 1 из 15 возможных

[19:13:17.605] [Процесс 7] Шаг 2 из 15 возможных

[19:13:17.605] [Процесс 2] Шаг 2 из 15 возможных

[19:13:17.605] [Процесс 3] Шаг 2 из 15 возможных

[19:13:17.605] [Процесс 5] Шаг 2 из 15 возможных

[19:13:17.605] [Процесс 9] Шаг 2 из 15 возможных

[19:13:17.605] [Процесс 10] Шаг 2 из 2 возможных

[19:13:17.605] [Процесс 4] Шаг 2 из 15 возможных

[19:13:17.605] [Процесс 8] Шаг 2 из 15 возможных

[19:13:17.605] [Процесс 6] Шаг 2 из 15 возможных

[19:13:17.605] [Процесс 1] Шаг 2 из 15 возможных

[19:13:18.109] [Процесс 5] Шаг 3 из 15 возможных

[19:13:18.109] [Процесс 1] Шаг 3 из 15 возможных

[19:13:18.109] [Процесс 6] Шаг 3 из 15 возможных

[19:13:18.109] [Процесс 7] Шаг 3 из 15 возможных

[19:13:18.109] [Процесс 4] Шаг 3 из 15 возможных

[19:13:18.109] [Процесс 10] Finish

[19:13:18.109] [Процесс 9] Шаг 3 из 15 возможных

[19:13:18.109] [Процесс 3] Шаг 3 из 15 возможных

[19:13:18.109] [Процесс 8] Шаг 3 из 15 возможных

[19:13:18.109] [Процесс 2] Шаг 3 из 15 возможных

[19:13:18.612] [Процесс 2] Шаг 4 из 15 возможных

[19:13:18.612] [Процесс 7] Шаг 4 из 15 возможных

[19:13:18.612] [Процесс 4] Шаг 4 из 15 возможных

[19:13:18.612] [Процесс 6] Шаг 4 из 15 возможных

[19:13:18.612] [Процесс 5] Шаг 4 из 15 возможных

[19:13:18.612] [Процесс 9] Шаг 4 из 15 возможных

[19:13:18.612] [Процесс 8] Шаг 4 из 15 возможных

[19:13:18.612] [Процесс 3] Шаг 4 из 15 возможных

[19:13:18.612] [Процесс 1] Шаг 4 из 15 возможных

[19:13:19.117] [Процесс 1] Шаг 5 из 15 возможных

[19:13:19.117] [Процесс 6] Шаг 5 из 15 возможных

[19:13:19.117] [Процесс 4] Шаг 5 из 15 возможных

[19:13:19.117] [Процесс 8] Шаг 5 из 15 возможных

[19:13:19.117] [Процесс 2] Шаг 5 из 15 возможных

[19:13:19.117] [Процесс 3] Шаг 5 из 15 возможных

[19:13:19.117] [Процесс 5] Шаг 5 из 15 возможных

[19:13:19.117] [Процесс 7] Шаг 5 из 15 возможных

[19:13:19.117] [Процесс 9] Шаг 5 из 15 возможных

[19:13:19.622] [Процесс 5] Шаг 6 из 15 возможных

[19:13:19.622] [Процесс 2] Шаг 6 из 15 возможных

[19:13:19.622] [Процесс 9] Шаг 6 из 15 возможных

[19:13:19.622] [Процесс 1] Шаг 6 из 15 возможных

[19:13:19.622] [Процесс 6] Шаг 6 из 15 возможных

[19:13:19.622] [Процесс 3] Шаг 6 из 15 возможных

[19:13:19.622] [Процесс 8] Шаг 6 из 15 возможных

[19:13:19.622] [Процесс 7] Шаг 6 из 15 возможных

[19:13:19.622] [Процесс 4] Шаг 6 из 15 возможных

[19:13:20.127] [Процесс 6] Шаг 7 из 15 возможных

[19:13:20.127] [Процесс 1] Шаг 7 из 15 возможных

[19:13:20.127] [Процесс 7] Шаг 7 из 15 возможных

[19:13:20.127] [Процесс 8] Шаг 7 из 15 возможных

[19:13:20.127] [Процесс 5] Шаг 7 из 15 возможных

[19:13:20.127] [Процесс 4] Шаг 7 из 15 возможных

[19:13:20.127] [Процесс 2] Шаг 7 из 15 возможных

[19:13:20.127] [Процесс 3] Шаг 7 из 15 возможных

[19:13:20.127] [Процесс 9] Шаг 7 из 15 возможных

[19:13:20.632] [Процесс 8] Шаг 8 из 15 возможных

[19:13:20.632] [Процесс 7] Шаг 8 из 15 возможных

[19:13:20.632] [Процесс 2] Шаг 8 из 15 возможных

[19:13:20.632] [Процесс 9] Шаг 8 из 15 возможных

[19:13:20.632] [Процесс 1] Шаг 8 из 15 возможных

[19:13:20.632] [Процесс 6] Шаг 8 из 15 возможных

[19:13:20.632] [Процесс 5] Шаг 8 из 15 возможных

[19:13:20.632] [Процесс 3] Шаг 8 из 15 возможных

[19:13:20.632] [Процесс 4] Шаг 8 из 15 возможных

[19:13:21.138] [Процесс 4] Шаг 9 из 15 возможных

[19:13:21.138] [Процесс 6] Шаг 9 из 15 возможных

[19:13:21.138] [Процесс 8] Шаг 9 из 15 возможных

[19:13:21.138] [Процесс 2] Шаг 9 из 15 возможных

[19:13:21.138] [Процесс 9] Шаг 9 из 15 возможных

[19:13:21.138] [Процесс 5] Шаг 9 из 15 возможных

[19:13:21.138] [Процесс 3] Шаг 9 из 15 возможных

[19:13:21.138] [Процесс 7] Шаг 9 из 15 возможных

[19:13:21.138] [Процесс 1] Шаг 9 из 15 возможных

[19:13:21.642] [Процесс 8] Шаг 10 из 15 возможных

[19:13:21.642] [Процесс 1] Шаг 10 из 15 возможных

[19:13:21.642] [Процесс 3] Шаг 10 из 15 возможных

[19:13:21.642] [Процесс 7] Шаг 10 из 15 возможных

[19:13:21.642] [Процесс 2] Шаг 10 из 15 возможных

[19:13:21.642] [Процесс 6] Шаг 10 из 15 возможных

[19:13:21.642] [Процесс 5] Шаг 10 из 15 возможных

[19:13:21.642] [Процесс 9] Шаг 10 из 15 возможных

[19:13:21.642] [Процесс 4] Шаг 10 из 15 возможных

Внимание: Превышено общее время выполнения (5 сек.).

Not GG... (Тайм-аут)

# Контрольные вопросы

1. **Для чего применяется тип IAsyncresult?**
   1. Позволяет отслеживать состояние асинхронной операции
   2. Предоставляет механизмы ожидания завершения через AsyncWaitHandle
   3. Содержит информацию о том, завершена ли операция (IsCompleted)
2. **Как реализовать ожидание завершения выполнения асинхронного метода с использованием тайм-аута?**
   1. CancellationTokenSource
   2. Task.WhenAny с Task.Delay
3. **Поясните назначение метода WaitOne( ).**
   1. Ожидание завершения асинхронной операции