**Выполнили студент 215 группы**

**Марков К.А.**

**Практическое занятие 16**

**Тема: «Применение асинхронных делегатов»**

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

**Задание:**

1. Создайте консольное приложение C# в Visual Studio.

2. Реализуйте приложение с использованием тайм-аута (модифицируйте приложение, полученное в ходе выполнения лабораторной работы №2).

3. Реализуйте механизм вывода информации в консоль о ходе решения задачи асинхронным методом.

class Program

{

delegate Task<int> AsyncMatrixDifference(int rows, int cols);

static async Task Main(string[] args)

{

// Вызов асинхронного метода и ожидание завершения с тайм-аутом

int matrixSize = 5;

AsyncMatrixDifference asyncMethod = (rows, cols) => GetMatrixDifferenceAsync(rows, cols);

var resultTask = asyncMethod(matrixSize, matrixSize);

// Ожидание выполнения задачи с тайм-аутом

var timeoutTask = Task.Delay(TimeSpan.FromSeconds(5)); // Тайм-аут в 5 секунд

var completedTask = await Task.WhenAny(resultTask, timeoutTask);

if (completedTask == timeoutTask)

{

Console.WriteLine("Время ожидания истекло. Процесс выполнения прерван.");

return;

}

// Вывод результата

int result = await resultTask;

Console.WriteLine($"Разница между максимальным и минимальным элементами матрицы: {result}");

Console.ReadKey();

}

// Асинхронный метод для вычисления разницы между максимальным и минимальным элементами матрицы

static async Task<int> GetMatrixDifferenceAsync(int rows, int cols)

{

var random = new Random();

int[,] matrix = new int[rows, cols];

Console.WriteLine("Исходная матрица:");

// Заполнение исходной матрицы случайными числами и вывод информации о ходе выполнения

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

{

for (int j = 0; j < cols; j++)

{

matrix[i, j] = random.Next(100); // Заполнение случайными значениями от 0 до 99

Console.Write(matrix[i, j] + "\t");

}

Console.WriteLine();

await Task.Delay(100); // Задержка для имитации работы

Console.WriteLine($"Обработано строк: {i + 1} из {rows}");

}

// Нахождение максимального и минимального элементов

int max = int.MinValue;

int min = int.MaxValue;

foreach (var num in matrix)

{

max = Math.Max(max, num);

min = Math.Min(min, num);

}

Console.WriteLine($"Максимальный элемент: {max}");

Console.WriteLine($"Минимальный элемент: {min}");

// Возвращение разницы между максимальным и минимальным элементами

return max - min;

}

}

**Вопросы:**

1. Для чего применяется тип IAsyncResult?

- Тип `IAsyncResult` применяется для представления асинхронной операции и предоставляет механизм для отслеживания и контроля выполнения асинхронных операций в .NET Framework. Этот тип используется для получения информации о состоянии асинхронной операции, доступа к результатам ее выполнения и управления временем ожидания завершения операции.

2. Как реализовать ожидание завершения выполнения асинхронного метода с использованием тайм-аута?

- Для ожидания завершения выполнения асинхронного метода с использованием тайм-аута можно воспользоваться методом `Task.WhenAny`. Этот метод позволяет ожидать завершения одной из задач, которая завершится первой, либо задачи, представляющей асинхронную операцию, либо задачи с тайм-аутом. Если завершается задача с тайм-аутом, то это означает, что время ожидания истекло.

- В случае использования асинхронных методов, предоставляемых средствами .NET Framework, таких как `Task<T>` или `async/await`, можно воспользоваться перегрузкой метода `Task.Wait(Int32)` для ожидания завершения задачи с тайм-аутом.

3. Пояснение назначения метода WaitOne( ):

- Метод `WaitOne()` является частью класса `WaitHandle` и используется для блокировки текущего потока выполнения до тех пор, пока не будет получен сигнал от объекта синхронизации, к которому этот метод применяется.

- Этот метод часто используется в многопоточных приложениях для организации ожидания событий или сигналов от других потоков или процессов.

- Метод `WaitOne()` может принимать параметр, указывающий время ожидания сигнала в миллисекундах. Если время ожидания истекает до получения сигнала, метод возвращает `false`, и выполнение продолжается.

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