**Выполнили студент 215 группы**

**Марков К.А.**

**Практическое занятие 15**

**Тема: «Применение асинхронных делегатов»**

**Цель работы:** научиться использовать делегаты для организации многопоточного приложения.

**Задание:**

1. Создайте приложение на языке C# в MS Visual Studio.

2. В соответствии с индивидуальным вариантом разработайте требуемый тип делегата (пользовательский, библиотечный или лямбда выражение).

3. Реализуйте асинхронное выполнение метода на основе разработанного делегата с возможностью мониторинга процесса выполнения, передачи параметров в метод и получения результата работы метода.



class Program

{

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

static void Main(string[] args)

{

// Вызов асинхронного метода и ожидание завершения

int matrixSize = 5;

AsyncMatrixDifference asyncMethod = (rows, cols) => Task.Run(() => GetMatrixDifference(rows, cols));

var resultTask = asyncMethod(matrixSize, matrixSize);

int result = resultTask.GetAwaiter().GetResult();

// Вывод результата

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

Console.ReadKey();

}

// Метод для вычисления разницы между максимальным и минимальным элементами матрицы

static int GetMatrixDifference(int rows, int cols)

{

var random = new Random();

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

Console.WriteLine("Исходная матрица:");

// Заполнение исходной матрицы случайными числами

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();

}

// 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. Пояснение назначения типа IAsyncResult:

- `IAsyncResult` является интерфейсом, предназначенным для представления асинхронной операции в .NET Framework. Он используется для возвращения информации о состоянии и результате асинхронной операции. Этот интерфейс включает в себя свойства для доступа к результатам выполнения операции, состоянию операции, а также методы для ожидания завершения операции и получения результата.

2. Асинхронные делегаты:

- Асинхронные делегаты (async delegates) представляют собой механизм асинхронного выполнения методов в .NET Framework. Они позволяют вызывать методы асинхронно, то есть без блокирования основного потока выполнения программы.

- Для создания асинхронных делегатов в C# обычно используется ключевое слово `async` перед объявлением метода и ключевое слово `await` при вызове асинхронной операции.

- Асинхронные делегаты упрощают разработку асинхронного кода, позволяя избежать явного создания и управления потоками. Они улучшают отзывчивость приложений и позволяют лучше использовать ресурсы системы.

**Вывод:** научился использовать делегаты для организации многопоточного приложения.