**Выполнили студент 215 группы**

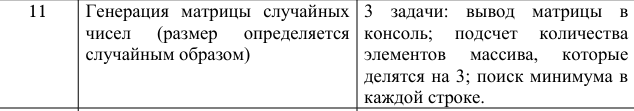
**Марков К.А.**

**Практическое занятие 22**

**Тема: «ЗАДАЧИ ПРОДОЛЖЕНИЯ»**

**Цель работы:** научиться использовать задачи продолжения в многопоточных программах.

**Задание:**

class Program

{

static void Main(string[] args)

{

Random rand = new Random();

int rows = rand.Next(3, 10);

int cols = rand.Next(3, 10);

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

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

{

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

{

matrix[i, j] = rand.Next(1, 100);

}

}

Task printTask = Task.Run(() => PrintMatrix(matrix));

Task<int> countDivisibleByThreeTask = printTask.ContinueWith(antecedent =>

{

return CountDivisibleByThree(matrix);

});

countDivisibleByThreeTask.ContinueWith(antecedent =>

{

Console.WriteLine($"Количество элементов, делящихся на 3: {antecedent.Result}");

FindMinInEachRow(matrix);

});

countDivisibleByThreeTask.Wait(); // Ожидаем завершения задачи подсчёта

Console.ReadKey();

}

static void PrintMatrix(int[,] matrix)

{

Console.WriteLine("Вывод матрицы:");

for (int i = 0; i < matrix.GetLength(0); i++)

{

for (int j = 0; j < matrix.GetLength(1); j++)

{

Console.Write($"{matrix[i, j]} ");

}

Console.WriteLine();

}

}

static int CountDivisibleByThree(int[,] matrix)

{

int count = 0;

for (int i = 0; i < matrix.GetLength(0); i++)

{

for (int j = 0; j < matrix.GetLength(1); j++)

{

if (matrix[i, j] % 3 == 0)

{

count++;

}

}

}

return count;

}

static void FindMinInEachRow(int[,] matrix)

{

Console.WriteLine("Минимумы в каждой строке:");

for (int i = 0; i < matrix.GetLength(0); i++)

{

int min = int.MaxValue;

for (int j = 0; j < matrix.GetLength(1); j++)

{

if (matrix[i, j] < min)

{

min = matrix[i, j];

}

}

Console.WriteLine($"Строка {i + 1}: {min}");

}

}

}

**Вопросы:**

Вопрос 1: Задача продолжения (ContinueWith( )) в .NET — это механизм в библиотеке Task Parallel Library (TPL), который позволяет автоматически начать выполнение новой задачи после завершения предыдущей. Этот механизм используется для организации последовательной цепочки задач, где каждая последующая задача запускается только после выполнения и, возможно, в зависимости от результата предыдущей.

Метод **ContinueWith** в .NET, используемый для создания задачи продолжения, принимает несколько важных параметров, которые позволяют управлять поведением и контекстом выполнения продолжения. Вот основные параметры, которые может принимать этот метод:

1. **Action<Task> или Action<Task, Object> (делегаты)**
2. **Object (state object)**:

**Вывод:** научился использовать задачи продолжения в многопоточных программах.