**20 ПАРАЛЛЕЛЬНОЕ ПРОГРАМИРОВАНИЕ**

Задание 1. Создайте и реализуйте метод решения задачи и выполните его в объектах класса Task используя три варианта создания объектов класса Task: дано четырехзначное число. Найти сумму его цифр.

Листинг программы:

try

{

var secNum = 0;

Console.Write("Введмте четырехзначное число: ");

var str = Console.ReadLine();

var number = int.Parse(str);

Task task = new Task(() =>

{

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

{

secNum += number % 10;

number /= 10;

}

Console.WriteLine($"Result (v1) = {secNum}");

});

task.Start();

task.Wait();

var task2 = Task.Factory.StartNew(() =>

{

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

{

secNum += number % 10;

number /= 10;

}

Console.WriteLine($"Result (v2) = {secNum}");

});

task2.Wait();

Task task3 = Task.Run(() =>

{

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

{

secNum += number % 10;

number /= 10;

}

Console.WriteLine($"Result (v3) = {secNum}");

});

task3.Wait();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Таблица 20.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1234 | Result (v1) = 10  Result (v2) = 10  Result (v3) = 10 |

Анализ результатов:

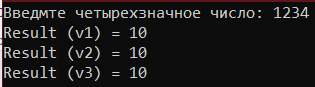


Рисунок 20.1 – Анализ результата приложения

Источник: собственная разработка

Задание 2. Создайте массив из 2 задач (объектов класс Task) в каждом

объекте выполните вычисление значения функций и выполните условия: дождитесь выполнения всех задач; дождитесь выполнения хот бы одной задачи. Замедлить выполнение задачи можно с помощью Thread.Sleep(n) в методе, выполняемом задачей; где n – время в миллисекундах.

Листинг программы:

try

{

Task[] tasks = new Task[2];

var x = 2;

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

{

if (i == 0)

{

tasks[i] = Task.Factory.StartNew(() =>

{

double result = (Math.Sin((Math.PI / 2) + 3 \* x)) / (1 - Math.Sin(3 \* x - Math.PI));

Thread.Sleep(2000);

Console.WriteLine(result);

});

tasks[i].Wait();

}

else

{

tasks[i] = Task.Factory.StartNew(() =>

{

double result = 1 / Math.Tan(((5 / 4) \* Math.PI) + ((3 / 2) \* x));

Thread.Sleep(2500);

Console.WriteLine(result);

});

tasks[i].Wait();

}

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Таблица 20.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 1,3324881179798564  -0,4576575543602856 |

Анализ результатов:



Рисунок 20.2 – Анализ результата приложения

Источник: собственная разработка