**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 результата разработанного приложения представлен на рисунке 20.1.

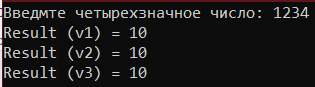


Рисунок 20.1 – Анализ результата приложения

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 результата разработанного приложения представлен на рисунке 20.2.



Рисунок 20.2 – Анализ результата приложения

Источник: собственная разработка

Задание №3. Создайте два объекта класса Task. Первый объект возвращает результат вычисления, второй объект является задачей продолжения первого объекта и выводит результат первой задачи на консоль. Задания для реализации метода выполняемого в первом объекте класса Task: дано двузначное число. Определить его первую и последнюю цифры.

Листинг программы:

try

{

Console.Write("Введите двузначное число: ");

var str = Console.ReadLine();

int number = int.Parse(str);

var lastNum = 0;

var firstNum = 0;

if (str.Length == 2)

{

Task task1 = new Task(() =>

{

lastNum = number % 10;

firstNum = number / 10;

});

Task task2 = task1.ContinueWith(t => Console.WriteLine($"First number = {firstNum}\nLast number = {lastNum}"));

task1.Start();

task2.Wait();

}

else

{

Console.WriteLine("Введено не двузначное число!");

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Таблица 20.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 12 | First number = 1  Last number = 2 |

Анализ результата разработанного приложения представлен на рисунке 20.3.

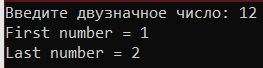


Рисунок 20.3 – Анализ результата приложения

Источник: собственная разработка

Задание №4. Выполните параллельное вычисления значений функции для каждого значения на отрезке [А,B]. Используя метод Parallel.For. Параметр A – 0, параметр B – 0. Функция – x-sinx.

Листинг программы:

try

{

const int A = 0;

const int B = 10;

Console.Write("Введите х: ");

var x = int.Parse(Console.ReadLine());

Parallel.For(A, B, Task);

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

void Task(int x)

{

Console.WriteLine($"Result = {x - Math.Sin(x)}");

}

Таблица 20.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 10 | Result = 0,1585290151921035  Result = 1,0907025731743183  Result = 6,279415498198926  Result = 6,343013401281211  Result = 0  Result = 8,587881514758243  Result = 2,8588799919401326  Result = 4,756802495307928  Result = 5,958924274663138  Result = 7,0106417533766185 |

Анализ результата разработанного приложения представлен на рисунке 20.4.

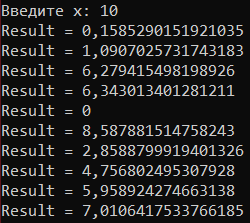


Рисунок 20.4 – Анализ результата приложения

Источник: собственная разработка

Задание №5. Выполните прерывание выполнения метода Parallel.Foreach при некотором условии. Функции суммы и произведение чисел от 0 до N где N значения из массива (списка). Массив значений – {1,5,8,40,70}.

Листинг программы:

try

{

var multV = 1;

var summV = 0;

ParallelLoopResult result = Parallel.ForEach<int>(new List<int>() { 1, 3, 5, 8 }, Mult);

void Mult(int n, ParallelLoopState pls)

{

Console.WriteLine($"Result mult = {multV \*= n}");

Console.WriteLine($"Result summ = {summV += n}");

if (multV > 20)

{

Console.WriteLine("Цикл прерван (mult > 20)");

pls.Break();

}

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Таблица 20.5 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1,5,8,40,70 | Result mult = 120  Result mult = 120  Result summ = 9  Цикл прерван (mult > 20)  Result summ = 8  Цикл прерван (mult > 20)  Result mult = 5  Result summ = 14  Цикл прерван (mult > 20)  Result mult = 15  Result summ = 17  Цикл прерван (mult > 20) |

Анализ результата разработанного приложения представлен на рисунке 20.5.

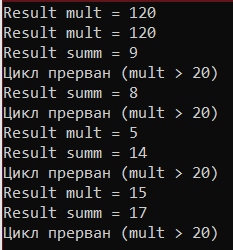


Рисунок 20.5 – Анализ результата приложения

Источник: собственная разработка