**20 Параллельное программирование**

Задание №1. Дано четырехзначное число. Найти число, образуемое при перестановке первой и последней цифр.

Листинг программы:

var number = 0;

while (true)

{

try

{

Console.Write("Enter 4-digit number: ");

number = int.Parse(Console.ReadLine());

if (number < 1000 || number > 9999)

{

throw new FormatException();

}

FirstTask(number);

SecondTask(number);

ThirdTask(number);

break;

}

catch (Exception e)

{

Console.WriteLine($"An error occured - {e.Message}");

}

}

static string intParser(int num)

{

int lastNumber = num % 10;

int firstNumber = num / 1000;

string SwapNumber = $"{lastNumber}{(num / 10) % 100}{firstNumber}";

return SwapNumber;

}

void FirstTask(int num)

{

Task SwapNum1 = Task.Run(() =>

{

Console.WriteLine($"Task 1 res: {intParser(num)}");

});

SwapNum1.Wait();

}

void SecondTask(int num)

{

Task SwapNum2 = Task.Factory.StartNew(() =>

{

Console.WriteLine($"Task 2 res: {intParser(num)}");

});

SwapNum2.Wait();

}

void ThirdTask(int num)

{

Thread.CurrentThread.Name = "Main\_Thread";

Task SwapNum3 = new Task(() =>

{

Console.WriteLine($"Task 3 res: {intParser(num)}");

}

);

SwapNum3.Start();

SwapNum3.Wait();

}

Таблица 20.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1779 | 9771, 9771, 9771 |

Источник: собственная разработка

Анализ результата разработки библиотеки представлен на рисунке 20.1.

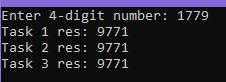


Рисунок 20.1 – Код разработанной библиотеки

Источник: собственная разработка

Задание №2. Создайте массив из 2 задач (объектов класс Task) в каждом объекте выполните вычисление значений и выполните условия:

дождитесь выполнения всех задач;

дождитесь выполнения хот бы одной задачи.

Замедлить выполнение задачи необходимо с помощью Thread.Sleep(n) в методе, выполняемом задачей; где n – время в миллисекундах.

Вычисление значений выполнить в соответствии с функциями:

.

Листинг программы:

int a = 0;

while (true)

{

try

{

Console.Write($"Enter a value: ");

a = int.Parse(Console.ReadLine());

Task[] tasks = new Task[2]

{

new Task(() => Console.WriteLine($"First Task: Z1 - {FirstExpression(a)}")),

new Task(() => {

Console.WriteLine($"Second Task: Z2 - {SecondExpression(a)}");

}

)

};

foreach (var t in tasks)

{

t.Start();

}

int ind = Task.WaitAny(tasks);

Console.WriteLine($"Task {tasks[ind].Id} completed - waiting end.");

Console.WriteLine("\nCurrent Tusks status:");

foreach (var t in tasks)

Console.WriteLine($"Task {t.Id}: {t.Status}");

Console.WriteLine();

Task.WaitAll(tasks);

Console.WriteLine("Tasks completed!");

break;

}

catch (Exception e)

{

Console.WriteLine($"An error occured - {e.Message}");

}

}

double FirstExpression(int a)

{

Thread.Sleep(10);

var z1 = Math.Round((Math.Cos(a) + Math.Cos(2 \* a) + Math.Cos(5 \* a) + Math.Cos(7 \* a)), 4);

return z1;

}

double SecondExpression(int a)

{

Thread.Sleep(200);

var z2 = Math.Round((4 \* Math.Cos(a / 2) \* Math.Cos(5 / 2) \* Math.Cos(4 \* a)), 4);

return z2;

}

Таблица 20.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 67 | -2.0092, -0.0126 |

Источник: собственная разработка

Анализ результатов представлен на рисунке20.2.

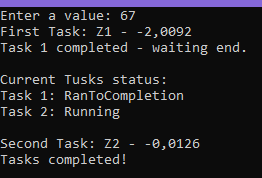


Рисунок 20.2 – Результат работы программы

Источник: собственная разработка

Задание №3. Создайте два объекта класса Task. Первый объект возвращает результат вычисления суммы цифр четырёхзначного числа. Второй объект является задачей продолжения первого объекта и выводит результат первой задачи на консоль.

Листинг программы:

int x = 0;

while (true)

{

try

{

int Sum = 0;

Console.Write($"Enter number: ");

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

Task[] tasks = new Task[2]

{

new Task(() => {

Sum = NumbersSuming(x);

}),

new Task(() => Console.WriteLine("1"))

};

tasks[1] = tasks[0].ContinueWith(task =>

{

Console.WriteLine($"Result of summing numbers: {Sum}");

});

tasks[0].Start();

tasks[1].Wait();

break;

}

catch (Exception e)

{

Console.WriteLine($"An error occured - {e.Message}");

}

}

static int NumbersSuming(int num)

{

var numbers = new int[] { num / 1000, (num / 100) % 10, (num % 100) / 10, num % 10 };

return numbers.Sum();

}

Таблица 20.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 9188 | 26 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 20.3.

. 

Рисунок 20.3 – Результат работы программы

Источник: собственная разработка

Задание №4. Выполните параллельное вычисления значений функции sin x - tg x для каждого значения на отрезке [А, B]. Значение A = -4, значение B = 6 Использовать метод Parallel.For.

Листинг программы:

try

{

const int A = -4;

const int B = 6;

Console.Write("Enter x value: ");

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

Parallel.For(A, B, Task);

}

catch (Exception e)

{

Console.WriteLine($"An error occured - {e.Message}");

}

void Task(int x)

{

Console.WriteLine($"Result X = {Math.Round(Math.Sin(x) - Math.Tan(x), 4)}");

}

Таблица 20.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| -4, 6, 7 | -3.0943, 1.9146, 0, 0.7159, 3.0943, -1.9146, 2.4216, -0.7159, -0.2837, 0.2837 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 20.4.

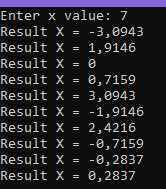


Рисунок 20.4 – Результат работы программы

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: {46, 83, 15, 39}

Листинг программы:

try

{

var Summ = 0;

var multiple = 1;

ParallelLoopResult result = Parallel.ForEach<int>(new List<int>() { 46, 83, 15, 39 }, Expression);

void Expression(int a, ParallelLoopState end)

{

Console.WriteLine($"Result of summing numbers = {Summ += a}");

Console.WriteLine($"Result of multiplying numbers = {multiple \*= a}");

if (Summ > 131)

{

Console.WriteLine("Cycle interrupted (Summ is over 39)");

end.Break();

}

}

}

catch (Exception e)

{

Console.WriteLine($"An error occured - {e.Message}");

}

Таблица 20.5 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 46, 83, 15, 39 | 46, 129, 144, 183, 2233530, 57270, 3818, 46 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 20.5.

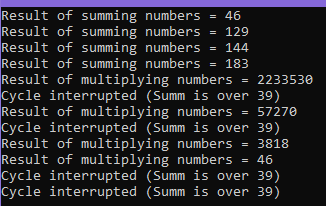


Рисунок 20.5 – Результат работы программы

Источник: собственная разработка