**ПРАКТИЧЕСКОЕ ЗАДАНИЕ 20**

**ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ**

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 число, образуемое при перестановке двух первых цифр заданного числа.

Листинг программы

try

{

var secNum = 0;

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

var str = Console.ReadLine();

var number = int.Parse(str);

var f1 = 0;

var f2 = 0;

var f3 = 0;

var f4 = 0;

var temp = 0;

Task task = new Task(() =>

{

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

{

temp = number;

f1 = number / 1000;

temp = number - f1 \* 1000;

f2 = temp / 100;

temp -= f2 \* 100;

f3 = temp / 10;

temp -= f3 \* 10;

f4 = temp;

secNum = f2 \* 1000 + f1 \* 100 + f3 \* 10 + f4;

}

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

});

task.Start();

task.Wait();

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

{

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

{

temp = number;

f1 = number / 1000;

temp = number - f1 \* 1000;

f2 = temp / 100;

temp -= f2 \* 100;

f3 = temp / 10;

temp -= f3 \* 10;

f4 = temp;

secNum = f2 \* 1000 + f1 \* 100 + f3 \* 10 + f4;

}

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

});

task2.Wait();

Task task3 = Task.Run(() =>

{

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

{

temp = number;

f1 = number / 1000;

temp = number - f1 \* 1000;

f2 = temp / 100;

temp -= f2 \* 100;

f3 = temp / 10;

temp -= f3 \* 10;

f4 = temp;

secNum = f2 \* 1000 + f1 \* 100 + f3 \* 10 + f4;

}

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

});

task3.Wait();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1234 | 2134 |

Анализ результатов:

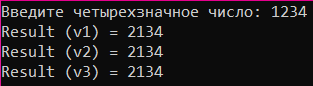


Рисунок 1.1 – Результат работы программы

Задание 2. Создайте массив из 2 задач (объектов класс Task) в каждом объекте выполните вычисление значения функций и выполните условия: 1. Дождитесь выполнения всех задач; 2. 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( 2 \* x)) + (Math.Sin(5 \* x)) - (Math.Sin(3 \* x))) / ((Math.Cos(x) + 1 - (2 \* (Math.Pow(Math.Sin(x), 2))) \* 2 \* x));

Thread.Sleep(2000);

Console.WriteLine(result);

});

tasks[i].Wait();

}

else

{

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

{

double result = (2 \* Math.Sin( x ));

Thread.Sleep(2500);

Console.WriteLine(result);

});

tasks[i].Wait();

}

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Таблица 1.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 0,1693674858778092  1,8185948536513634 |

Анализ результатов:



Рисунок 1.2 – Результат работы программы

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 трёхзначное число. Найти сумму его первой и последней цифр.

Листинг программы:

try

{

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

var str = Console.ReadLine();

int number = int.Parse(str);

var lastNum = 0;

var firstNum = 0;

var summa = 0;

if (str.Length == 3)

{

Task task1 = new Task(() =>

{

lastNum = number % 10;

firstNum = number / 100;

summa = lastNum + firstNum;

});

Task task2 = task1.ContinueWith(t => Console.WriteLine($"Summa = {summa}\n"));

task1.Start();

task2.Wait();

}

else

{

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

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Таблица 1.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 123 | 4 |

Анализ результатов:



Рисунок 1.3 – Результат работы программы

Задание 4. Выполните параллельное вычисления значений функции для каждого значения на отрезке [А,B]. Используя метод Parallel.For.



Листинг программы:

try

{

const int A = -8;

const int B = 5;

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 = {Math.Pow(Math.Cos(x), 2)}");

}

Таблица 1.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 | Result = 0,9219269793662459  Result = 0,17317818956819406  Result = 0,021170259838307684  Result = 1  Result = 0,5683686091039167  Result = 0,08046423546177377  Result = 0,2919265817264289  Result = 0,4272499830956933  Result = 0,2919265817264289  Result = 0,4272499830956933  Result = 0,17317818956819406  Result = 0,9800851433251829  Result = 0,9800851433251829 |

Анализ результатов:

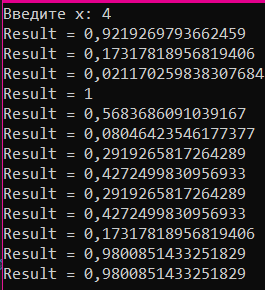


Рисунок 1.4 – Результат работы программы

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).



Листинг программы:

try

{

var multV = 1;

var summV = 0;

ParallelLoopResult result = Parallel.ForEach<int>(new List<int>() { 3, 7, 93, 90 }, Mult);

void Mult(int n, ParallelLoopState pls)

{

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

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

if (multV > 35)

{

Console.WriteLine("Цикл прерван (mult > 35)");

pls.Break();

}

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Таблица 1.5 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Result mult = 93  Result summ = 93  Цикл прерван (mult > 35)  Result mult = 651  Result summ = 100  Цикл прерван (mult > 35)  Result mult = 58590  Result summ = 190  Цикл прерван (mult > 35)  Result mult = 175770  Result summ = 193  Цикл прерван (mult > 35) |

Анализ результатов:

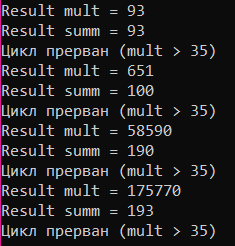


Рисунок 1.5 – Результат работы программы