**20 ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ**

Задание №1. Дано трехзначное число, в котором все цифры различны. Получить шесть чисел, образованных при перестановке цифр заданного числа.

Листинг программы:

var num = 0;

while (true)

{

try

{

Console.Write("Enter num: ");

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

if (num < 100 || num > 999)

{

throw new FormatException();

}

TaskVariant1(num);

TaskVariant2(num);

TaskVariant3(num);

break;

}

catch (Exception ex)

{

Console.WriteLine($"Some error! ({ex})");

}

}

static int[] intParser(int num)

{

var variants = new int[][] {

new int []{ 0, 1, 2},

new int []{ 0, 2, 1},

new int []{ 2, 0, 1},

new int []{ 2, 1, 0},

new int []{ 1, 2, 0},

new int []{ 1, 0, 2},

};

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

var result = new int[variants.GetLength(0)];

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

{

var item = variants[i];

result[i] = digits[item[0]] \* 100 + digits[item[1]] \* 10 + digits[item[2]];

}

return result;

}

void TaskVariant1(int num)

{

Task ex1 = Task.Run(() =>

{

var intVariation = intParser(num);

Console.WriteLine("Resault: ");

foreach (var item in intVariation)

{

Console.WriteLine($"{item} | Task - 1");

}

});

ex1.Wait();

}

void TaskVariant2(int num)

{

Task ex2 = Task.Factory.StartNew(() =>

{

var intVariation = intParser(num);

Console.WriteLine("Resault: ");

foreach (var item in intVariation)

{

Console.WriteLine($"{item} | Task - 2");

}

});

ex2.Wait();

}

void TaskVariant3(int num)

{

Thread.CurrentThread.Name = "Main";

Task taskA = new Task(() =>

{

var intVariation = intParser(num);

Console.WriteLine("Resault: ");

foreach (var item in intVariation)

{

Console.WriteLine($"{item} | Task - 3");

}

}

);

taskA.Start();

taskA.Wait();

}

Таблица 20.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 564 | 564, 546, 456, 465, 645, 654 |

Источник: собственная разработка

Анализ результата разработки библиотеки представлен на рисунке 20.1.

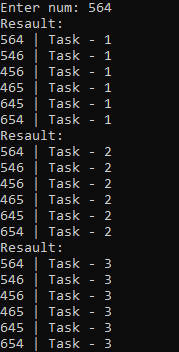


Рисунок 20.1 – Результат работы программы

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 x = 0;

while (true)

{

try

{

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

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

Task[] tasks= new Task[2]

{

new Task(() => Console.WriteLine($"Task - 1 | Z1 - {FormulaZ1(x)}")),

new Task(() => {

Console.WriteLine($"Task - 2 | Z2 - {FormulaZ2(x)}");

}

) };

foreach (var t in tasks)

{

t.Start();

}

int index = Task.WaitAny(tasks);

Console.WriteLine("Wait ended because task #{0} completed.",

tasks[index].Id);

Console.WriteLine("\nCurrent Status of Tasks:");

foreach (var t in tasks)

Console.WriteLine(" Task {0}: {1}", t.Id, t.Status);

Console.WriteLine();

Task.WaitAll(tasks);

Console.WriteLine("Tasks completed");

break;

}

catch (Exception ex)

{

Console.WriteLine($"Some error! ({ex.Message})");

}

}

double FormulaZ1(int x)

{

Thread.Sleep(10);

var z1 = (Math.Pow(x, 2) + 2 \* x - 3 + (x + 1) \* Math.Sqrt(Math.Pow(x, 2) - 9)) /

(Math.Pow(x, 2) - 2 \* x - 3 + (x - 1) \* Math.Sqrt(Math.Pow(x, 2) - 9));

return z1;

}

double FormulaZ2(int x)

{

Thread.Sleep(200);

var z2 = Math.Sqrt((x + 3) / (x - 3));

return z2;

}

Таблица 20.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 56 | 1.05508, 1 |

Источник: собственная разработка

Анализ результатов представлен на рисунке20.2.

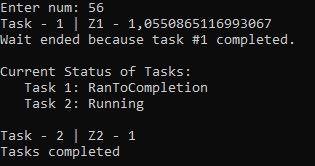


Рисунок 20.2 – Результат работы программы

Источник: собственная разработка

Задание №3. Создайте два объекта класса Task. Первый объект возвращает результат в виде шести чисел, образованных при перестановке цифр заданного трёхзначного числа, в котором все цифры различны. Второй объект является задачей продолжения первого объекта и выводит результат первой задачи на консоль.

Листинг программы:

int x = 0;

while (true)

{

try

{

int[] formula = { };

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

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

Task[] tasks = new Task[2]

{

new Task(() => {

formula = intParser(x);

}),

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

};

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

{

foreach (var item in formula) {

Console.WriteLine(item);

}

});

tasks[0].Start();

tasks[1].Wait();

Console.WriteLine("Конец метода Main");

break;

}

catch (Exception ex)

{

Console.WriteLine($"Some error! ({ex.Message})");

}

}

static int[] intParser(int num)

{

var variants = new int[][]

{

new int []{ 0, 1, 2},

new int []{ 0, 2, 1},

new int []{ 2, 0, 1},

new int []{ 2, 1, 0},

new int []{ 1, 2, 0},

new int []{ 1, 0, 2},

};

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

var result = new int[variants.GetLength(0)];

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

{

var item = variants[i];

result[i] = digits[item[0]] \* 100 + digits[item[1]] \* 10 + digits[item[2]];

}

return result;

}

Таблица 20.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 781 | 781, 718, 178, 187, 817, 871 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 20.3.

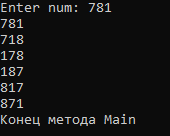
. 

Рисунок 20.3 – Результат работы программы

Источник: собственная разработка

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

Листинг программы:

Parallel.For(-6, 6, Square);

void Square(int n)

{

Console.WriteLine($"Выполняется задача {Task.CurrentId}");

Console.WriteLine($"sin(x) - cox(x) = {Math.Sin(n) - Math.Cos(n)}");

Thread.Sleep(2000);

}

Таблица 20.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| -6, 6 | 1.1311, 0.8488, -0.6807, -0.3011 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 20.4.

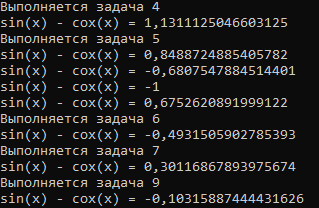


Рисунок 20.4 – Результат работы программы

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: {1, 4, 61, 85}.

Листинг программы:

try

{

var multV = 1;

var summV = 0;

ParallelLoopResult result = Parallel.ForEach<int>(new List<int>() { 1, 4, 61, 85 }, 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, 4, 61, 85 | 20740, 90, 244, 86, 151 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 20.5.

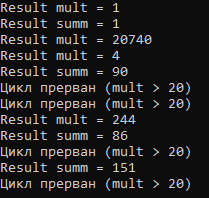


Рисунок 20.5 – Результат работы программы

Источник: собственная разработка