**Параллельное программирование**

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

Листинг программы:

static void Main(string[] args)

{

// Первый способ создания Task

Task firstTask = new Task(() =>

{

Write("Введите число для firstTask: ");

var digit = ToInt32(ReadLine());

GetDigitsOfDigit(digit);

});

firstTask.Start();

firstTask.Wait();

// Второй способ создания Task

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

{

Write("Введите число для secondTask: ");

var digit = ToInt32(ReadLine());

GetDigitsOfDigit(digit);

});

secondTask.Wait();

// Третий способ создания Task

Task thirdTask = Task.Run(() =>

{

Write("Введите число для secondTask: ");

var digit = ToInt32(ReadLine());

GetDigitsOfDigit(digit);

});

thirdTask.Wait();

ReadKey();

}

Анализ результатов:

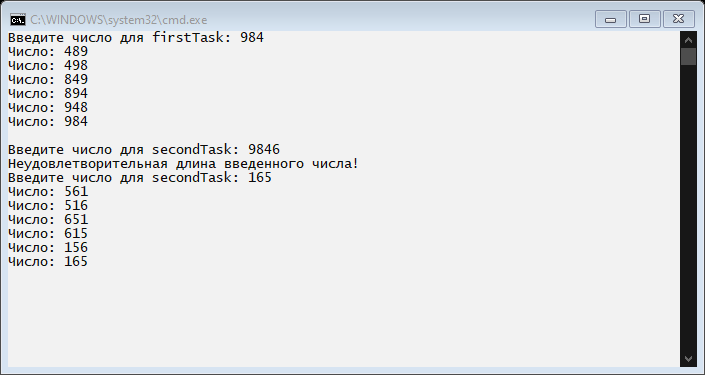


Рисунок 1.1 – Результат работы программы

**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 хотя бы одной задачи. Замедлить выполнение задачи можно с помощью Thread.Sleep(n) в методе, выполняемом задачей; где n – время в миллисекундах.

Листинг программы:

static void Main(string[] args)

{

Task[] tasks = new Task[2];

MathManager mathManager = new MathManager();

tasks[0] = Task.Run(() =>

{

WriteLine($"FirstAction: {mathManager.FirstAction(15):f2}\n");

Thread.Sleep(1000);

});

tasks[0].Wait();

tasks[1] = Task.Run(() => WriteLine($"SecondAction: {mathManager.SecondAction(20):f2}\n"));

tasks[1].Wait();

ReadKey();

}

Анализ результатов:

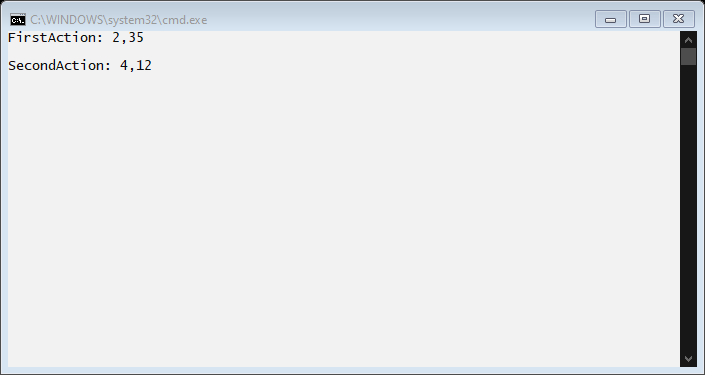


Рисунок 1.2 – Результат работы программы

**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. Дано трехзначное число, в котором все цифры различны. Получить шесть чисел, образованных при перестановке цифр заданного числа.

Листинг программы:

static void Main(string[] args)

{

DigitManager digitManager = new DigitManager();

var resultTask = Task<List<int>>.Run(() =>

{

return digitManager.GetDigits(568);

}).Result;

//resultTask.Start();

//var numbers = resultTask.Result;

Task writerTask = new Task(() => digitManager.WriteDigits(resultTask));

writerTask.Start();

ReadKey();

}

Анализ результатов:

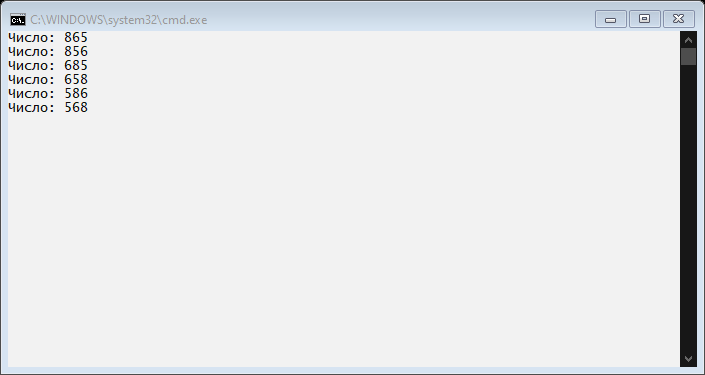


Рисунок 1.3 – Результат работы программы

**Задание №4.** Выполните параллельное вычисления значений функции для каждого значения на отрезке [А,B]. Используя метод Parallel.For.



Листинг программы:

class Program

{

private static void SinCos(int digit, Action<string> action)

{

action($"{Math.Cos(digit) - Math.Sin(digit):f2}");

}

static void Main(string[] args)

{

Parallel.For(-6, 6, (str) => SinCos(str, WriteLine));

ReadKey();

}

}

Анализ результатов:

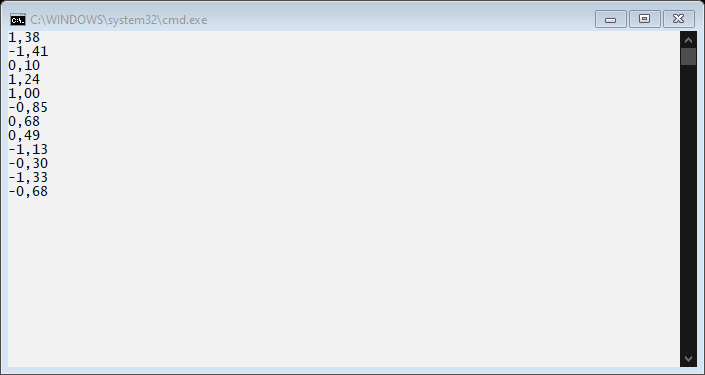


Рисунок 1.4 – Результат работы программы

**Задание №4.** 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:

static void Main(string[] args)

{

var digits = new List<int>() { 1, 4, 61, 85, 99, 150 };

double sqrt = default;

double degree = default;

ParallelLoopResult loopResult = Parallel.ForEach(digits, (str,state) => MakeSecondDegree(str, degree, WriteLine,state));

if (!loopResult.IsCompleted)

{ WriteLine($"Выполнение цикла завершено на итерации {loopResult.LowestBreakIteration}"); }

ParallelLoopResult loopResult1 = Parallel.ForEach(digits, (str, state) => CheckSqrt(str, sqrt, WriteLine, state));

if (!loopResult1.IsCompleted)

{ WriteLine($"Выполнение цикла завершено на итерации {loopResult.LowestBreakIteration}"); }

ReadKey();

}

Анализ результатов:

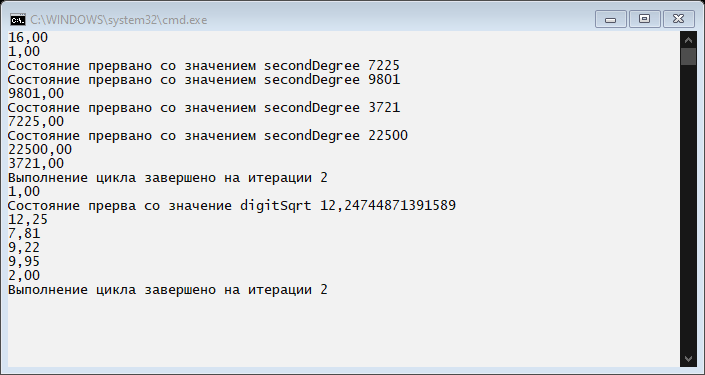


Рисунок 1.4 – Результат работы программы