**Лабораторна робота №2**

ТЕМА 1: ПАРАЛЕЛІЗМ ДАНИХ ТА ПАРАЛЕЛІЗМ ЗАДАЧ (Ч. 1)

Квартюк Олексій

Варіант 2

Завдання:

1. Створити програму, що створює дві задачі, які виконуються паралельно. Затримку методом Sleep() організувати на величину 200мс та пропорційно ідентифікатору задачі.

2. Організувати очікування виконання задач методом WaitAll().

3. Визначити задачу для виконання у вигляді лямбда-виразу.

4. Створити програму паралельних обчислень за допомогою виклику методу Invoke(), де в якості аргументів застосовуються лямбда-вирази

using System;

using System.Threading;

using System.Threading.Tasks;

class Program

{

static void ExecuteTask()

{

int taskIdentifier = Task.CurrentId ?? -1;

Console.WriteLine($"Задача {taskIdentifier} починається.");

for (int i = 0; i < 5; i++)

{

Thread.Sleep(taskIdentifier \* 200);

Console.WriteLine($"Задача {taskIdentifier}, лічильник = {i}");

}

Console.WriteLine($"Задача {taskIdentifier} завершена.");

}

static void Main()

{

Task task1 = new Task(ExecuteTask);

Task task2 = new Task(ExecuteTask);

task1.Start();

task2.Start();

Task.WaitAll(task1, task2);

Console.WriteLine("Всі задачі завершені.");

Task task3 = Task.Factory.StartNew(() =>

{

Console.WriteLine("Задача через лямбда-вираз починається.");

for (int i = 0; i < 5; i++)

{

Thread.Sleep(500);

Console.WriteLine($"Лямбда-вираз, лічильник = {i}");

}

Console.WriteLine("Задача через лямбда-вираз завершена.");

});

task3.Wait();

Parallel.Invoke(

() =>

{

Console.WriteLine("Паралельна задача 1 виконується.");

for (int i = 0; i < 5; i++)

{

Thread.Sleep(500);

Console.WriteLine($"Паралельна задача 1, лічильник = {i}");

}

Console.WriteLine("Паралельна задача 1 завершена.");

},

() =>

{

Console.WriteLine("Паралельна задача 2 виконується.");

for (int i = 0; i < 5; i++)

{

Thread.Sleep(500);

Console.WriteLine($"Паралельна задача 2, лічильник = {i}");

}

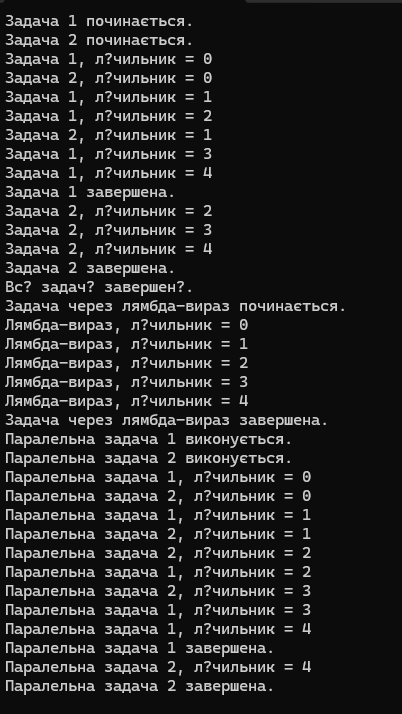
Console.WriteLine("Паралельна задача 2 завершена.");

}

);

}

}



Частина 2:

ТЕМА 2: ПАРАЛЕЛІЗМ ДАНИХ ТА ПАРАЛЕЛІЗМ ЗАДАЧ (Ч. 2)

Завдання

1. Створити програму, що використовує для розпаралелювання метод Parallel.For(). Провести ряд обчислювальних експериментів із різним типом елементів масиву (int, double); різною кількістю елементів масиву; різною складністю обчислень / 10 / / / x x x x x x x e x x e x π π π π = = = = Виміряти час, що витрачається паралельним способом обробки та послідовним. Результати експериментів оформити у вигляді таблиці. Зробити висновки.

2. Модифікувати попередню програму (створивши новий проект у рішенні) таким чином, щоб відбувався вихід з паралельного циклу за умови входження значення елемента у деякий окіл деякого числа (число та відхилення задаються константами).

3. Повторити приклади розпаралелювання за допомогою ForEach().

4. Модифікувати попередню програму (створивши новий проект у рішенні) таким чином, щоб тіло паралельного циклу задавалося лямбда-виразом

using System;

using System.Diagnostics;

using System.Threading.Tasks;

class Program

{

static double[] dataDouble;

static int[] dataInt;

// Обчислення для типу int

static void ComputeFirstInt(int i) => dataInt[i] = dataInt[i] / 10;

static void ComputeSecondInt(int i) => dataInt[i] = dataInt[i] / (int)Math.PI;

static void ComputeThirdInt(int i) => dataInt[i] = (int)(Math.Exp(dataInt[i]) / Math.Pow(dataInt[i], Math.PI));

static void ComputeFourthInt(int i) => dataInt[i] = (int)(Math.Exp(Math.PI \* dataInt[i]) / Math.Pow(dataInt[i], Math.PI));

// Обчислення для типу double

static void ComputeFirstDouble(int i) => dataDouble[i] = dataDouble[i] / 10;

static void ComputeSecondDouble(int i) => dataDouble[i] = dataDouble[i] / Math.PI;

static void ComputeThirdDouble(int i) => dataDouble[i] = Math.Exp(dataDouble[i]) / Math.Pow(dataDouble[i], Math.PI);

static void ComputeFourthDouble(int i) => dataDouble[i] = Math.Exp(Math.PI \* dataDouble[i]) / Math.Pow(dataDouble[i], Math.PI);

static void Main()

{

// Задаємо різні розміри масивів для експериментів

int[] sizes = { 100000, 1000000, 10000000 };

foreach (var size in sizes)

{

// Створення масивів для типу double

dataDouble = new double[size];

for (int i = 0; i < size; i++)

dataDouble[i] = i + 1;

Console.WriteLine($"Results for array size: {size}");

// Послідовне обчислення для типу double

Stopwatch sw = new Stopwatch();

sw.Start();

for (int i = 0; i < size; i++)

ComputeFirstDouble(i);

sw.Stop();

Console.WriteLine($"Serial ComputeFirst (double): {sw.Elapsed.TotalSeconds} sec");

// Паралельне обчислення для типу double

sw.Restart();

Parallel.For(0, size, ComputeFirstDouble);

sw.Stop();

Console.WriteLine($"Parallel ComputeFirst (double): {sw.Elapsed.TotalSeconds} sec");

// Послідовне обчислення для другого виразу (double)

sw.Restart();

for (int i = 0; i < size; i++)

ComputeSecondDouble(i);

sw.Stop();

Console.WriteLine($"Serial ComputeSecond (double): {sw.Elapsed.TotalSeconds} sec");

// Паралельне обчислення для другого виразу (double)

sw.Restart();

Parallel.For(0, size, ComputeSecondDouble);

sw.Stop();

Console.WriteLine($"Parallel ComputeSecond (double): {sw.Elapsed.TotalSeconds} sec");

// Послідовне обчислення для третього виразу (double)

sw.Restart();

for (int i = 0; i < size; i++)

ComputeThirdDouble(i);

sw.Stop();

Console.WriteLine($"Serial ComputeThird (double): {sw.Elapsed.TotalSeconds} sec");

// Паралельне обчислення для третього виразу (double)

sw.Restart();

Parallel.For(0, size, ComputeThirdDouble);

sw.Stop();

Console.WriteLine($"Parallel ComputeThird (double): {sw.Elapsed.TotalSeconds} sec");

// Послідовне обчислення для четвертого виразу (double)

sw.Restart();

for (int i = 0; i < size; i++)

ComputeFourthDouble(i);

sw.Stop();

Console.WriteLine($"Serial ComputeFourth (double): {sw.Elapsed.TotalSeconds} sec");

// Паралельне обчислення для четвертого виразу (double)

sw.Restart();

Parallel.For(0, size, ComputeFourthDouble);

sw.Stop();

Console.WriteLine($"Parallel ComputeFourth (double): {sw.Elapsed.TotalSeconds} sec");

// Створення масивів для типу int

dataInt = new int[size];

for (int i = 0; i < size; i++)

dataInt[i] = i + 1;

// Послідовне обчислення для типу int

sw.Restart();

for (int i = 0; i < size; i++)

ComputeFirstInt(i);

sw.Stop();

Console.WriteLine($"Serial ComputeFirst (int): {sw.Elapsed.TotalSeconds} sec");

// Паралельне обчислення для типу int

sw.Restart();

Parallel.For(0, size, ComputeFirstInt);

sw.Stop();

Console.WriteLine($"Parallel ComputeFirst (int): {sw.Elapsed.TotalSeconds} sec");

// Послідовне обчислення для другого виразу (int)

sw.Restart();

for (int i = 0; i < size; i++)

ComputeSecondInt(i);

sw.Stop();

Console.WriteLine($"Serial ComputeSecond (int): {sw.Elapsed.TotalSeconds} sec");

// Паралельне обчислення для другого виразу (int)

sw.Restart();

Parallel.For(0, size, ComputeSecondInt);

sw.Stop();

Console.WriteLine($"Parallel ComputeSecond (int): {sw.Elapsed.TotalSeconds} sec");

// Послідовне обчислення для третього виразу (int)

sw.Restart();

for (int i = 0; i < size; i++)

ComputeThirdInt(i);

sw.Stop();

Console.WriteLine($"Serial ComputeThird (int): {sw.Elapsed.TotalSeconds} sec");

// Паралельне обчислення для третього виразу (int)

sw.Restart();

Parallel.For(0, size, ComputeThirdInt);

sw.Stop();

Console.WriteLine($"Parallel ComputeThird (int): {sw.Elapsed.TotalSeconds} sec");

// Послідовне обчислення для четвертого виразу (int)

sw.Restart();

for (int i = 0; i < size; i++)

ComputeFourthInt(i);

sw.Stop();

Console.WriteLine($"Serial ComputeFourth (int): {sw.Elapsed.TotalSeconds} sec");

// Паралельне обчислення для четвертого виразу (int)

sw.Restart();

Parallel.For(0, size, ComputeFourthInt);

sw.Stop();

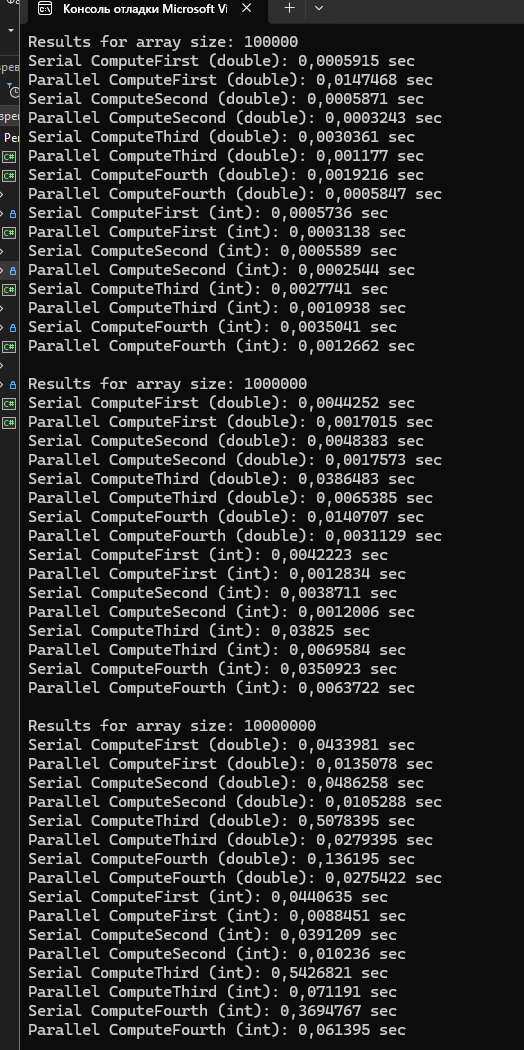
Console.WriteLine($"Parallel ComputeFourth (int): {sw.Elapsed.TotalSeconds} sec");

Console.WriteLine(); // Пустий рядок між різними розмірами масивів

}

}

}



using System;

using System.Diagnostics;

using System.Threading.Tasks;

class Program

{

static double[] data;

const double Target = 1000;

const double Epsilon = 5;

static void Compute(int i, ParallelLoopState state)

{

data[i] = i / 10;

data[i] = Math.Sin(i) \* 100;

if (Math.Abs(data[i] - Target) < Epsilon) state.Break();

}

static void Main()

{

int size = 1000000;

data = new double[size];

Stopwatch sw = new Stopwatch();

sw.Start();

ParallelLoopResult result = Parallel.For(0, size, Compute);

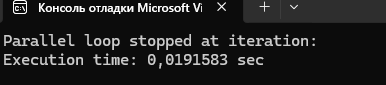
sw.Stop();

Console.WriteLine("Parallel loop stopped at iteration: " + result.LowestBreakIteration);

Console.WriteLine("Execution time: " + sw.Elapsed.TotalSeconds + " sec");

}

}



using System;

using System.Diagnostics;

using System.Threading.Tasks;

class Program

{

static double[] data;

static void Compute(int i)

{

data[i] = i / 10.0;

}

static void Main()

{

int size = 1000000;

data = new double[size];

Stopwatch sw = new Stopwatch();

sw.Start();

// Використовуємо Parallel.ForEach для обробки елементів масиву

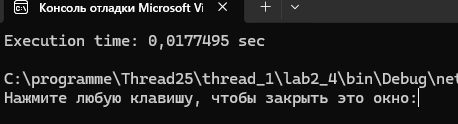
Parallel.ForEach(data, (value, state, index) => Compute((int)index));

sw.Stop();

Console.WriteLine("Execution time: " + sw.Elapsed.TotalSeconds + " sec");

}

}



using System;

using System.Diagnostics;

using System.Threading.Tasks;

class Program

{

static double[] data;

static void Main()

{

int size = 1000000;

data = new double[size];

Stopwatch sw = new Stopwatch();

sw.Start();

Parallel.For(0, size, i => { data[i] = Math.PI \* Math.Exp(-data[i]); });

Parallel.For(0, size, i => { data[i] = -data[i]/10; });

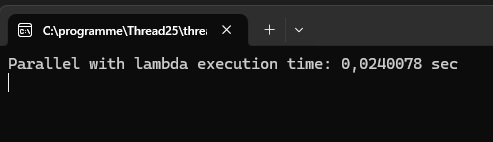
sw.Stop();

Console.WriteLine("Parallel with lambda execution time: " + sw.Elapsed.TotalSeconds + " sec");

Console.ReadLine();

}

}



Висновок: Отже, було використано методи \*\*Parallel.For()\*\* та \*\*Parallel.ForEach()\*\* для розпаралелювання обчислень. Проведено аналіз ефективності паралельної обробки у порівнянні з послідовною. Описано механізми керування паралельними циклами, зокрема дострокове завершення через \*\*Break()\*\*.