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

Застосування методу Parallel.For(). Найпростіша форма методу має вигляд:

```
public static ParallelLoopResult For(int fromInclusive, int toExclusive, Action<int> body)
```

де *fromInclusive* означає початкове значення того, що відповідає змінній управління циклом; воно називається також ітераційним, або індексним, значенням; а *toExclusive* - значення, на одиницю більше кінцевого. На кожному кроці циклу змінна управління циклом збільшується на одиницю. Отже, цикл поступово просувається від початкового значення *fromInclusive* до кінцевого значенням *toExclusive* мінус одиниця. Код, що циклічно виконується, вказується методом, переданим через параметр *body*. Цей метод повинен бути сумісний з делегатом *Action <int>*, який декларується в такий спосіб:

```
public delegate void Action<int>(T obj)
```

Для методу *For()* узагальнений параметр T повинен бути, звичайно, типу *int*. Значення, яке передається через параметр *obj*, буде наступним значенням змінної управління циклом. А метод, який передається через параметр *body*, може бути іменованих або анонімним.

Приклад програми, що використовує для розпаралелювання метод Parallel.For().

```
using System;
using System.Threading.Tasks;
using System.Diagnostics;

namespace ParallelFor
{
      class Program
      {
            static double[] data;

            //Метод, що служить в якості тіла паралельного циклу.
            static void MyTransform(int i)
            {
                  data[i] /= 10;

                  if (data[i] < 10000) data[i] = 0;
                  if ((data[i] >= 10000) &(data[i]<20000)) data[i] = 100;
                  if ((data[i] >= 20000) & (data[i] < 30000)) data[i] = 200;
                  if (data[i] > 30000) data[i] = 300;
            }

            static void Main(string[] args)
            {
                  Console.WriteLine("Main Thread is starting.");

                  Stopwatch sw=new Stopwatch();

                  data = new double[100000000];
```

```
                sw.Start();
                //Ініціювати дані в звичайному циклі for
                for (int i = 0; i < data.Length; i++)
                {
                     data[i] = i;
                }
                sw.Stop();

                Console.WriteLine("Serial initialization of cycle=
"+sw.Elapsed.TotalSeconds+" seconds.");

                sw.Reset();
                sw.Start();
                //Розпаралелити цикл методом Parallel.For
                Parallel.For(0, data.Length, MyTransform);
                sw.Stop();
                Console.WriteLine("Parallel transformation = " +
sw.Elapsed.TotalSeconds + " seconds.");

                sw.Reset();
                sw.Start();
                for (int i = 0; i < data.Length; i++)
                {
                     MyTransform(i);
                }
                sw.Stop();

                Console.WriteLine("Serial Transformation = " +
sw.Elapsed.TotalSeconds + " seconds.");

                Console.WriteLine("Main() is done.");
                Console.ReadLine();
           }
     }
}
```

Метод `Parallel.For()` повертає екземпляр об'єкта типу *`ParallelLoopResult`*. Це структура, в якій визначені дві властивості:

```
public bool IsCompleted { get; }
public Nullable<long> LowestBreakIteration { get; }
```

Властивість *`IsCompleted`* матиме логічне значення *`true`*, якщо виконано всі кроки циклу (при нормальному завершенні циклу).

Властивість *`LowestBreakIteration`* буде містити найменше значення змінної управління циклом, якщо цикл перерветься завчасно викликом методу *`ParallelLoopState.Break()`*.

Для доступу до об'єкту типу *`ParallelLoopState`* слід використовувати форму методу `Parallel.For()`, делегат якого приймає в якості другого параметра поточний стан циклу.

public static ParallelLoopResult For(int fromInclusive, int toExclusive, Action<int, ParallelLoopState> body)

Для передчасного завершення циклу слід скористатися методом *Break()*, що викликається для екземпляра об'єкта типу *ParallelLoopState* всередині тіла циклу, що визначається параметром *body*. Переривання паралельного циклу `Parallel.For()`, нерідко виявляється корисним при пошуку даних – якщо шукане значення знайдено, то продовжувати виконання циклу немає ніякої потреби.

```
static void MyTransform(int i, ParallelLoopState pls)
{
      //завершити цикл, якщо знайдено від'ємне значення
      if (data[i] < 0) pls.Break();
      …
}
static void Main(string[] args)
{
      Console.WriteLine("Main Thread is starting.");

      data = new double[100000000];

      //Ініціювати дані в звичайному циклі for
      for (int i = 0; i < data.Length; i++)
      {
            data[i] = i;
      }

      //Помістити від'ємне значення в масив
      data[1000] = -10;

      //Розпаралелити цикл методом Parallel.For
      ParallelLoopResult loopResult= Parallel.For(0, data.Length, MyTransform);

      //Перевірити, чи завершився цикл
      if (!loopResult.IsCompleted)
            Console.WriteLine("ParallelFor was aborted with negative value on
iteration " + loopResult.LowestBreakIteration);

      Console.WriteLine("Main() is done.");
      Console.ReadLine();
}
```

Результат виконання:

```
Main Thread is starting.
ParallelFor was aborted with negative value on iteration 1000
Main() is done.
```

Застосування методу ForEach(). Паралельний варіант циклу *foreach()*. Форма оголошення:

public static ParallelLoopResult ForEach<TSource>(IEnumerable<TSource> source, Action<TSource> body)

де *source* означає колекцію даних, що обробляються в циклі, а *body* – метод, що буде виконуватися на кожному кроці циклу. Цей метод приймає в якості свого аргументу значення або посилання на кожен елемент масиву, що обробляється в циклі, але не його індекс. В результаті повертається відомості про стан циклу.

Аналогічно до `Parallel.For()`, паралельне виконання циклу *ForEach()* можна зупинити, викликавши метод *Break()* для екземпляра об'єкта типу *ParallelLoopState* всередині тіла циклу, що визначається параметром *body*.

public static ParallelLoopResult ForEach<TSource>(IEnumerable<TSource> source, Action<TSource, ParallelLoopState> body)

Приклад програми:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ParallelForEachBreak
{
      class Program
      {
            static double[] data;

            //Метод, що служить в якості тіла паралельного циклу.
            static void MyTransform(double v, ParallelLoopState pls)
            {
                  //завершити цикл, якщо знайдено від'ємне значення
                  if (v < 0) pls.Break();

                  Console.WriteLine("Value is :" + v);
            }
            static void Main(string[] args)
            {
                  Console.WriteLine("Main Thread is starting.");

                  data = new double[100000000];

                  //Ініціювати дані в звичайному циклі for
                  for (int i = 0; i < data.Length; i++)
                  {
                        data[i] = i;
                  }

                  //Помістити від'ємне значення в масив
                  data[100000] = -10;

                  //Розпаралелити цикл методом Parallel.For
                  ParallelLoopResult loopResult= Parallel.ForEach(data, MyTransform);
```

```
                //Перевірити, чи завершився цикл
                if (!loopResult.IsCompleted)
                     Console.WriteLine("ParallelFor was aborted with negative value
on iteration " + loopResult.LowestBreakIteration);

                Console.WriteLine("Main() is done.");
                Console.ReadLine();
           }
      }
}
```

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
$$x = x / 10$$
$$x = x / \pi$$
$$x = e^x / x^\pi$$
$$x = e^{\pi x} / x^\pi$$
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.*