**ЛАБОРАТОРНА РОБОТА #2 ЧАСТИНА 2. ПАРАЛЕЛІЗМ ДАНИХ ТА ПАРАЛЕЛІЗМ ЗАДАЧ. (Ч. 1)**

Іваницька Анна, МІТ 41

**Завдання для виконання:**

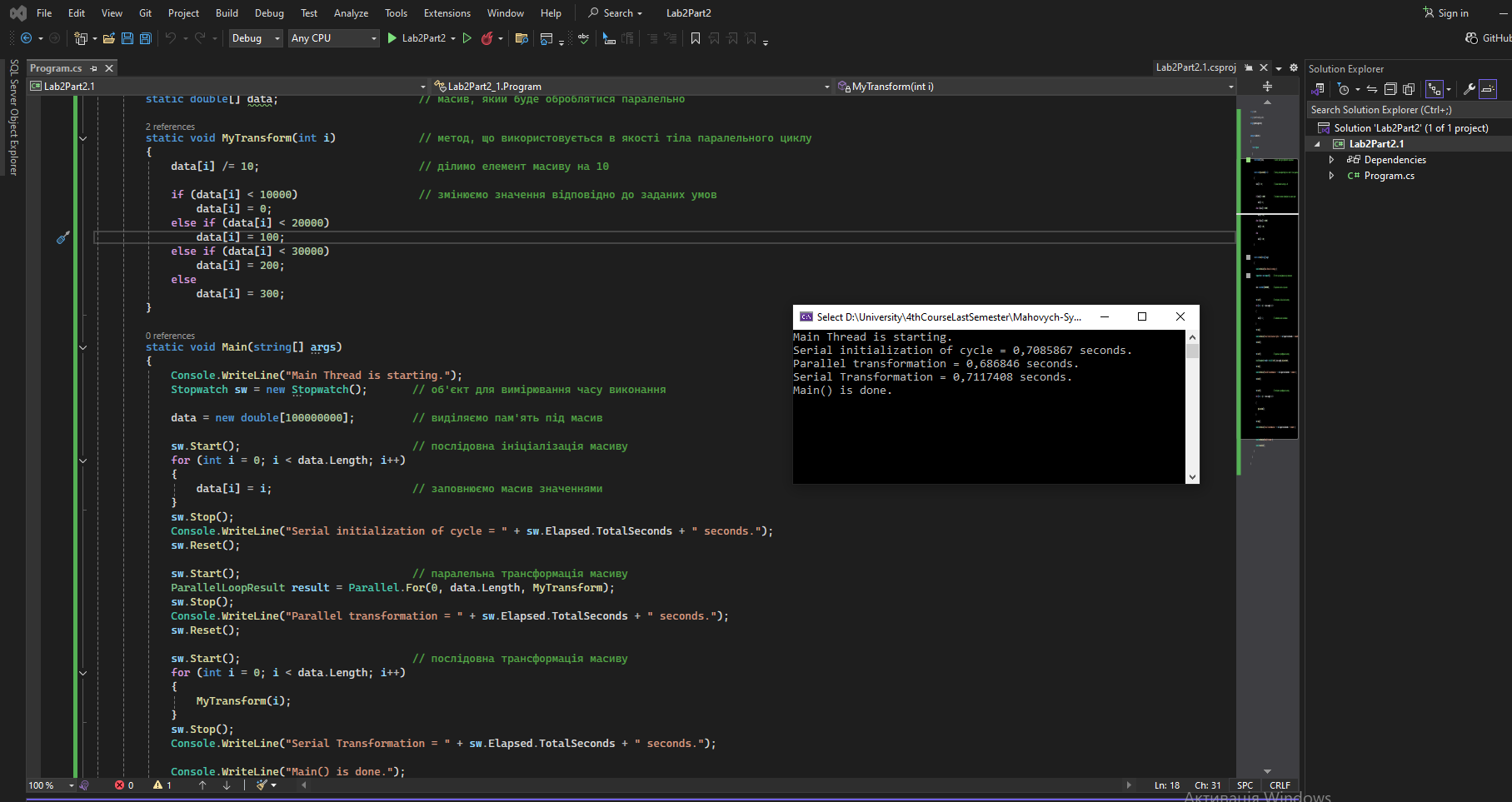
1. Повторити задачі
2. Створити програму, що використовує для розпаралелювання метод Parallel.For(). Провести ряд обчислювальних експериментів із різним типом елементів масиву (int, double); різною кількістю елементів масиву; різною складністю обчислень

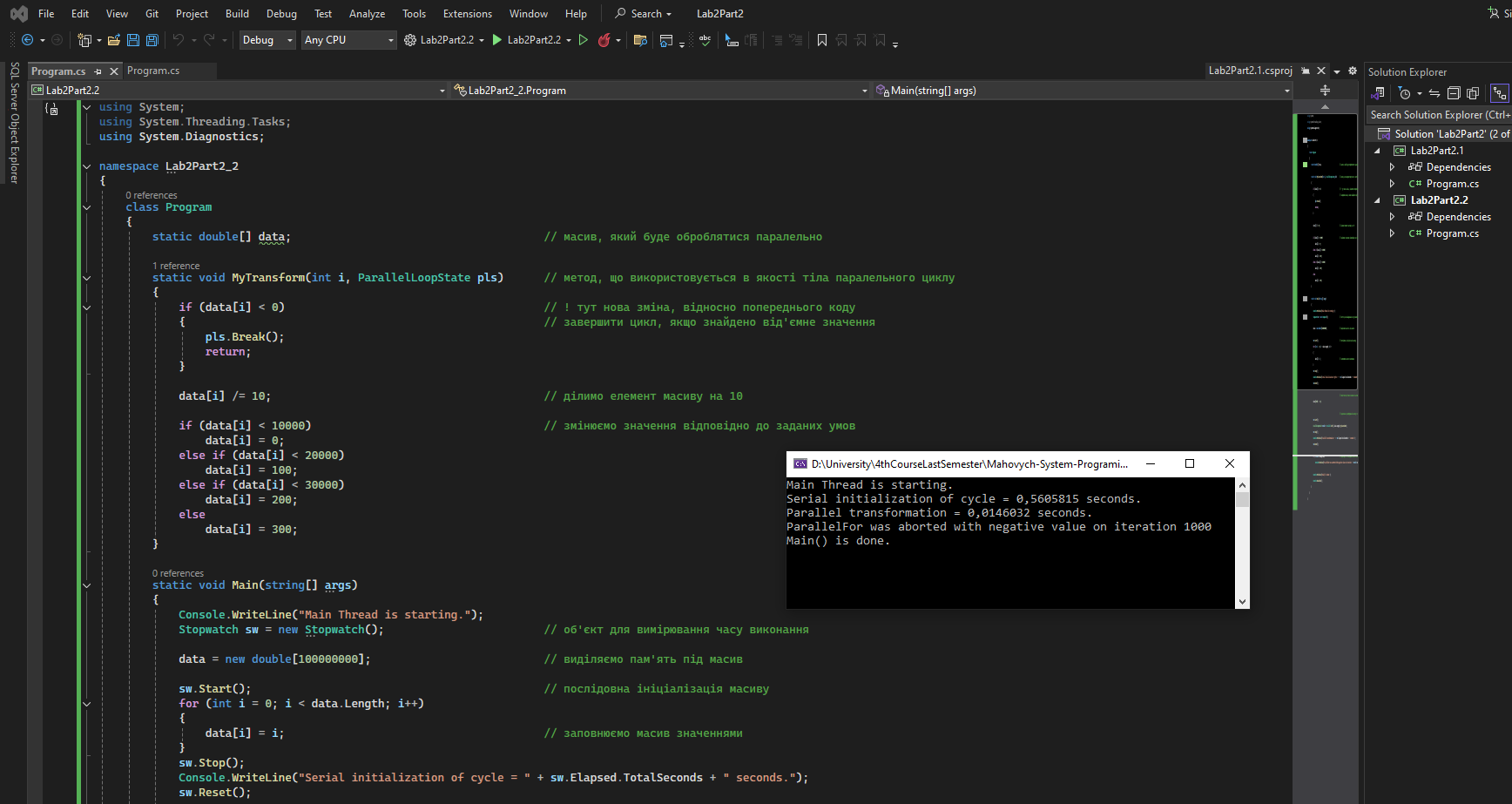
Виміряти час, що витрачається паралельним способом обробки та послідовним. Результати експериментів оформити у вигляді таблиці. Зробити висновки.

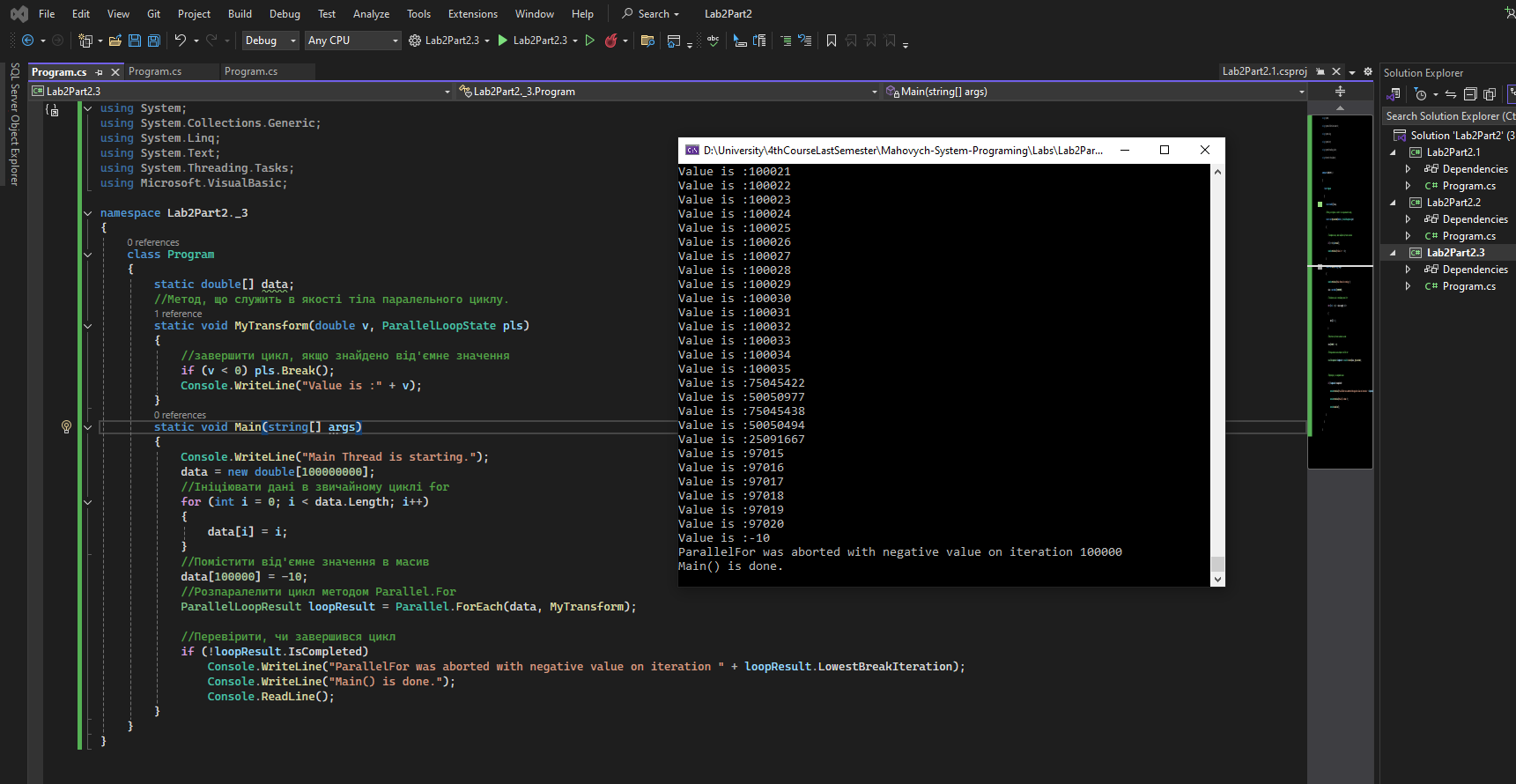
1. Модифікувати попередню програму (створивши новий проект у рішенні) таким чином, щоб відбувався вихід з паралельного циклу за умови входження значення елемента у деякий окіл деякого числа (число та відхилення задаються константами).
2. Повторити приклади розпаралелювання за допомогою ForEach().
3. Модифікувати попередню програму (створивши новий проект у рішенні) таким чином, щоб тіло паралельного циклу задавалося лямбда-виразом.

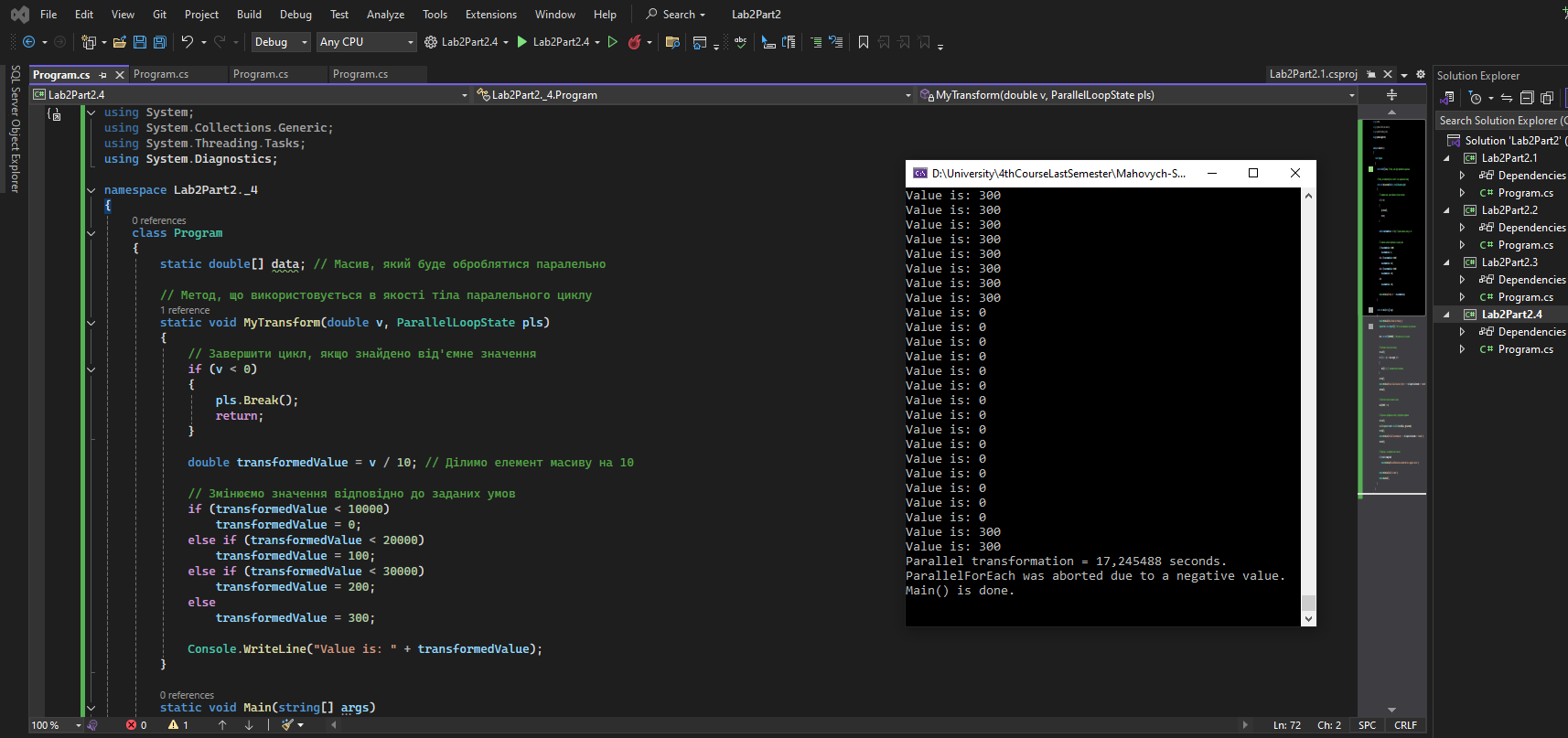
**Хід роботи**

**1 ПОВТОРИТИ ЗАДАЧІ**

**1.1 Застосування методу Parallel.For():**

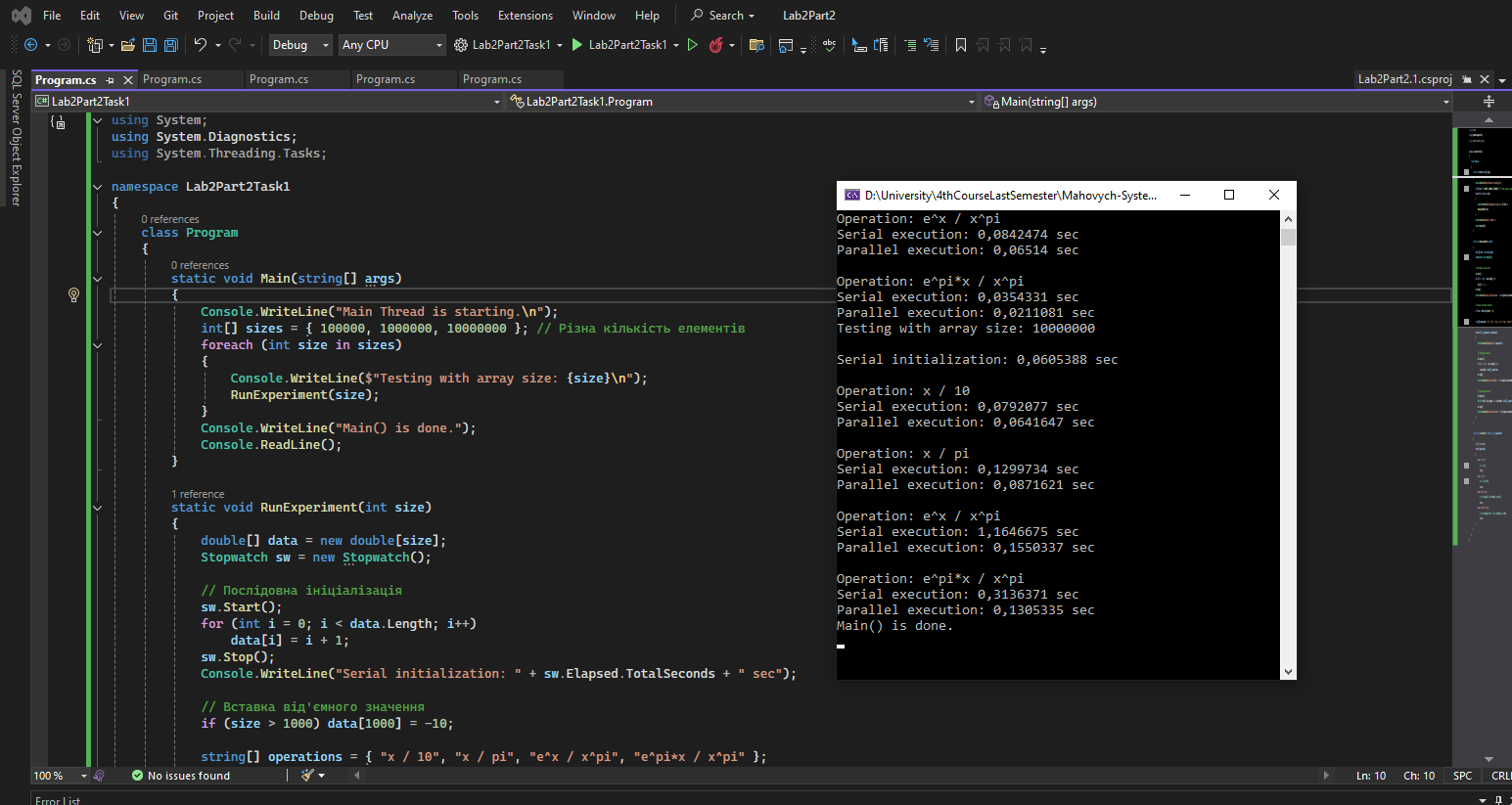
**1.2 Break() для передчасного завершення циклу:**

  
**1.3 Застосування методу ForEach() –**

**1.4 Використання Break() разом для передчасної зупинки ForEach():**

**2 СТВОРИТИ ПРОГРАМУ, ЩО ВИКОРИСТОВУЄ ДЛЯ РОЗПАРАЛЕЛЮВАННЯ МЕТОД PARALLEL.FOR().**

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

Виміряти час, що витрачається паралельним способом обробки та послідовним. Результати експериментів оформити у вигляді таблиці. Зробити висновки.

**Результат виконання:**

Main Thread is starting.

Testing with array size: **100000**

Serial initialization: 0,0007488 sec

Operation: x / 10

Serial execution: 0,0017628 sec

Parallel execution: 0,0183876 sec

Operation: x / pi

Serial execution: 0,0012815 sec

Parallel execution: 0,0008042 sec

Operation: e^x / x^pi

Serial execution: 0,0070353 sec

Parallel execution: 0,0052472 sec

Operation: e^pi\*x / x^pi

Serial execution: 0,0034422 sec

Parallel execution: 0,0016446 sec

Testing with array size: **1000000**

Serial initialization: 0,005829 sec

Operation: x / 10

Serial execution: 0,0084289 sec

Parallel execution: 0,0057748 sec

Operation: x / pi

Serial execution: 0,0137596 sec

Parallel execution: 0,0077986 sec

Operation: e^x / x^pi

Serial execution: 0,0803665 sec

Parallel execution: 0,0580602 sec

Operation: e^pi\*x / x^pi

Serial execution: 0,0247917 sec

Parallel execution: 0,0134043 sec

Testing with array size: **10000000**

Serial initialization: 0,0570273 sec

Operation: x / 10

Serial execution: 0,0843293 sec

Parallel execution: 0,0582576 sec

Operation: x / pi

Serial execution: 0,1770604 sec

Parallel execution: 0,0953337 sec

Operation: e^x / x^pi

Serial execution: 1,3239612 sec

Parallel execution: 0,1525796 sec

Operation: e^pi\*x / x^pi

Serial execution: 0,243635 sec

Parallel execution: 0,1317625 sec

Main() is done.

**Результат виконання у таблицях:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| На масиві розміром 100000 (найменший) | | | | |
| Спосіб обробки / Виконуване обчислення | x / 10 | x / pi | e^x / x^pi | e^pi\*x / x^pi |
| Послідовний  (Serial) | 0,0017 sec | 0,0012 sec | 0,007  sec | 0,0034  sec |
| Паралельний (Parallel) | 0,0183 sec | 0,0008 sec | 0,0052  sec | 0,0016  sec |

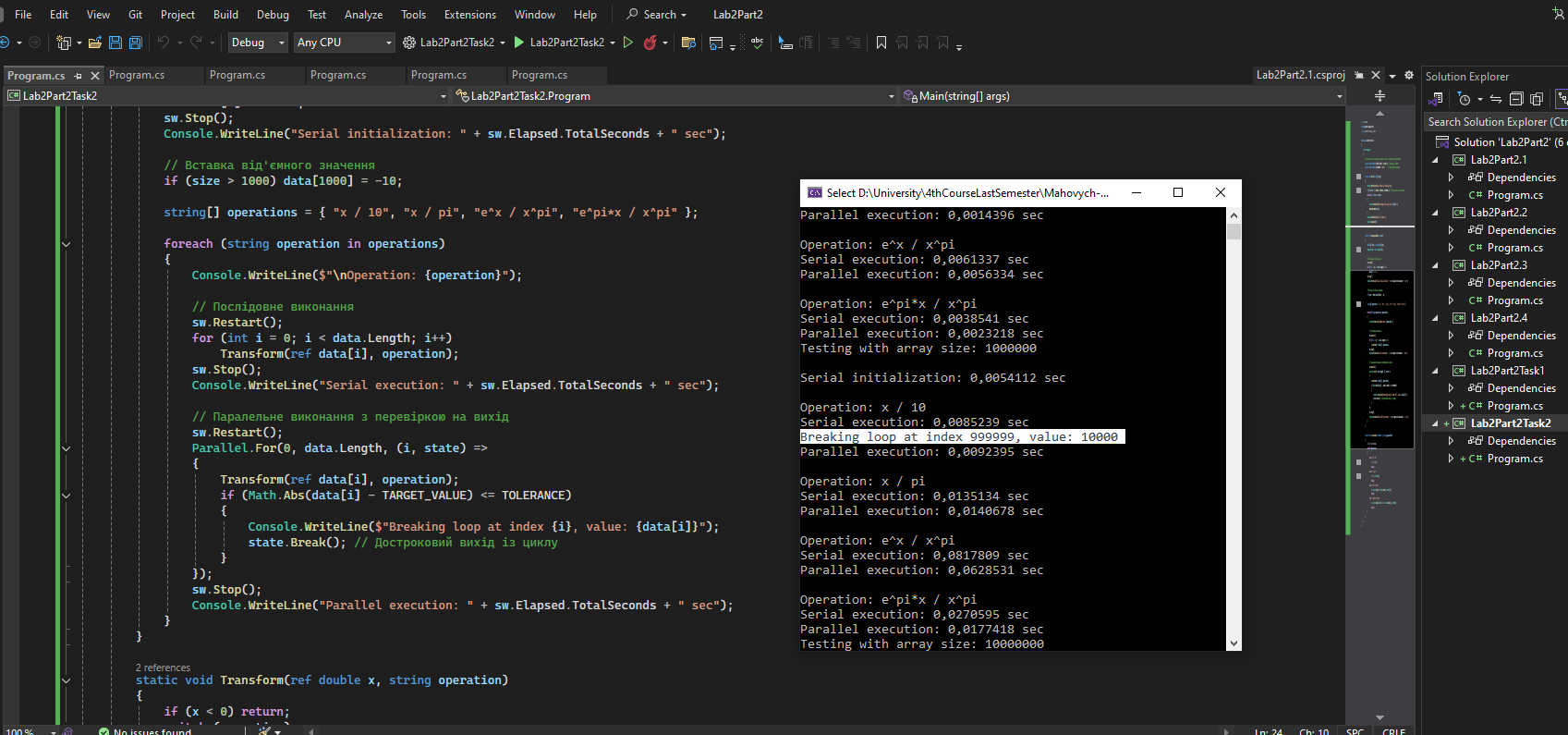
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| На масиві розміром 1000000 (середній) | | | | |
| Спосіб обробки / Виконуване обчислення | x / 10 | x / pi | e^x / x^pi | e^pi\*x / x^pi |
| Послідовний  (Serial) | 0,0084 sec | 0,0137 sec | 0,0803  sec | 0,0247  sec |
| Паралельний (Parallel) | 0,0057 sec | 0,0077 sec | 0,058  sec | 0,0134  sec |

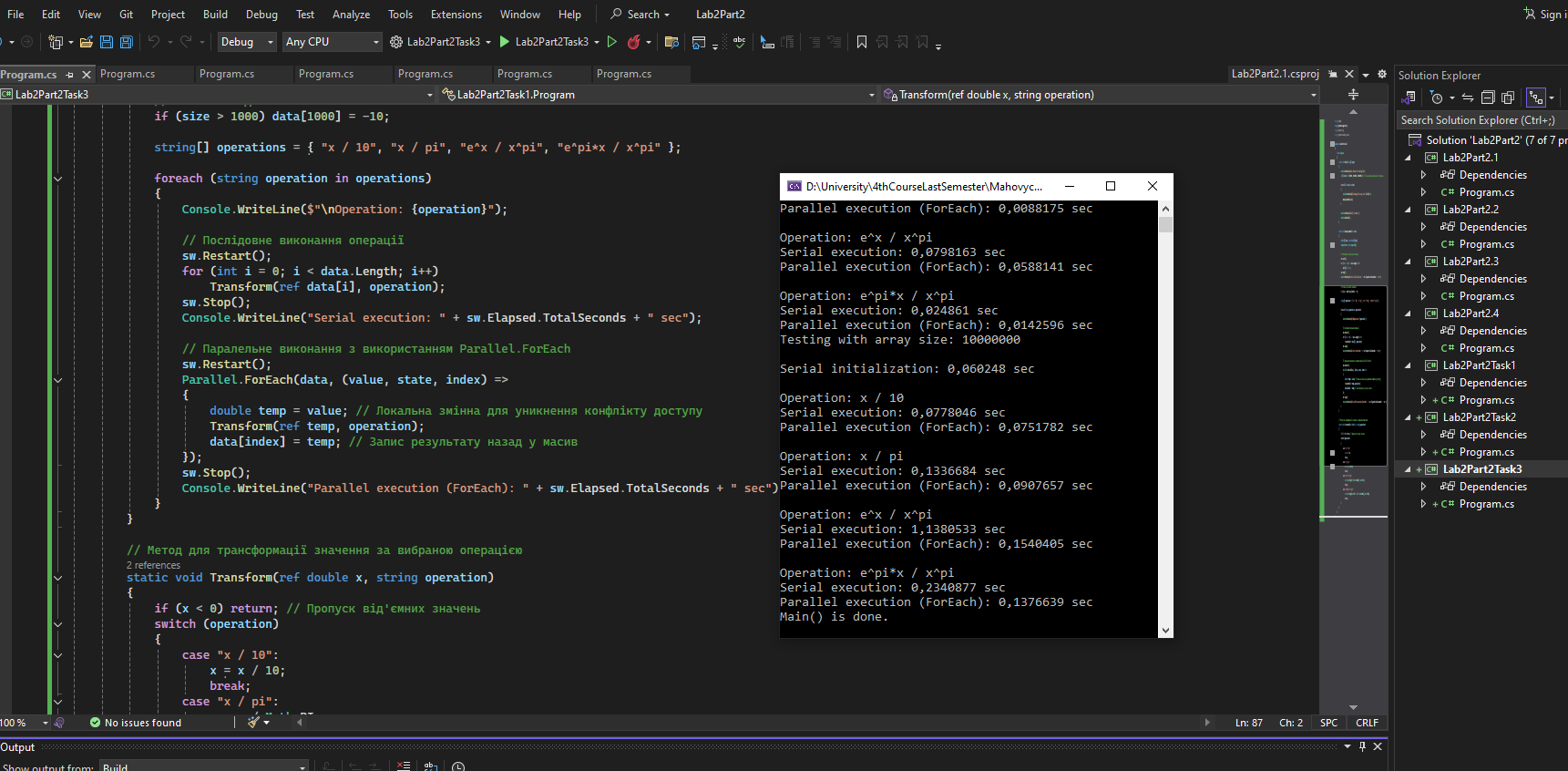
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| На масиві розміром 10000000 (найбільший) | | | | |
| Спосіб обробки / Виконуване обчислення | x / 10 | x / pi | e^x / x^pi | e^pi\*x / x^pi |
| Послідовний  (Serial) | 0,0843 sec | 0,177  sec | 1,3239  sec | 0,2436  sec |
| Паралельний (Parallel) | 0,0582 sec | 0,0953 sec | 0,1525  sec | 0,1317  sec |

Як можемо бачити, паралельна обробка даних відбувається швидше для усіх випадків, окрім найпростішого обчислення на масиві найменшого розміру, де послідовне обчислення виявилося ефективніше.

Зазвичай паралельна обробка відбувається трохи швидше або приблизно у два рази швидше в деяких випадках.

**3 МОДИФІКУВАТИ ПОПЕРЕДНЮ ПРОГРАМУ (ДОДАТИ ВИХІД З ПАРАЛЕЛЬНОГО ЦИКЛУ ЗА УМОВИ…)**

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).

**4 ПОВТОРИТИ ПРИКЛАДИ РОЗПАРАЛЕЛЮВАННЯ ЗА ДОПОМОГОЮ FOREACH().** - **ПАРАЛЕЛЬНИЙ ВАРІАНТ ЦИКЛУ FOREACH():**

**Результат виконання:**

Main Thread is starting.

Testing with array size: **100000**

Serial initialization: 0,0007713 sec

Operation: x / 10

Serial execution: 0,001867 sec

Parallel execution (ForEach): 0,023017 sec

Operation: x / pi

Serial execution: 0,001371 sec

Parallel execution (ForEach): 0,000912 sec

Operation: e^x / x^pi

Serial execution: 0,0062466 sec

Parallel execution (ForEach): 0,0054496 sec

Operation: e^pi\*x / x^pi

Serial execution: 0,0034426 sec

Parallel execution (ForEach): 0,0017544 sec

Testing with array size: **1000000**

Serial initialization: 0,0059513 sec

Operation: x / 10

Serial execution: 0,0081185 sec

Parallel execution (ForEach): 0,0067809 sec

Operation: x / pi

Serial execution: 0,0138935 sec

Parallel execution (ForEach): 0,0088175 sec

Operation: e^x / x^pi

Serial execution: 0,0798163 sec

Parallel execution (ForEach): 0,0588141 sec

Operation: e^pi\*x / x^pi

Serial execution: 0,024861 sec

Parallel execution (ForEach): 0,0142596 sec

Testing with array size: **10000000**

Serial initialization: 0,060248 sec

Operation: x / 10

Serial execution: 0,0778046 sec

Parallel execution (ForEach): 0,0751782 sec

Operation: x / pi

Serial execution: 0,1336684 sec

Parallel execution (ForEach): 0,0907657 sec

Operation: e^x / x^pi

Serial execution: 1,1380533 sec

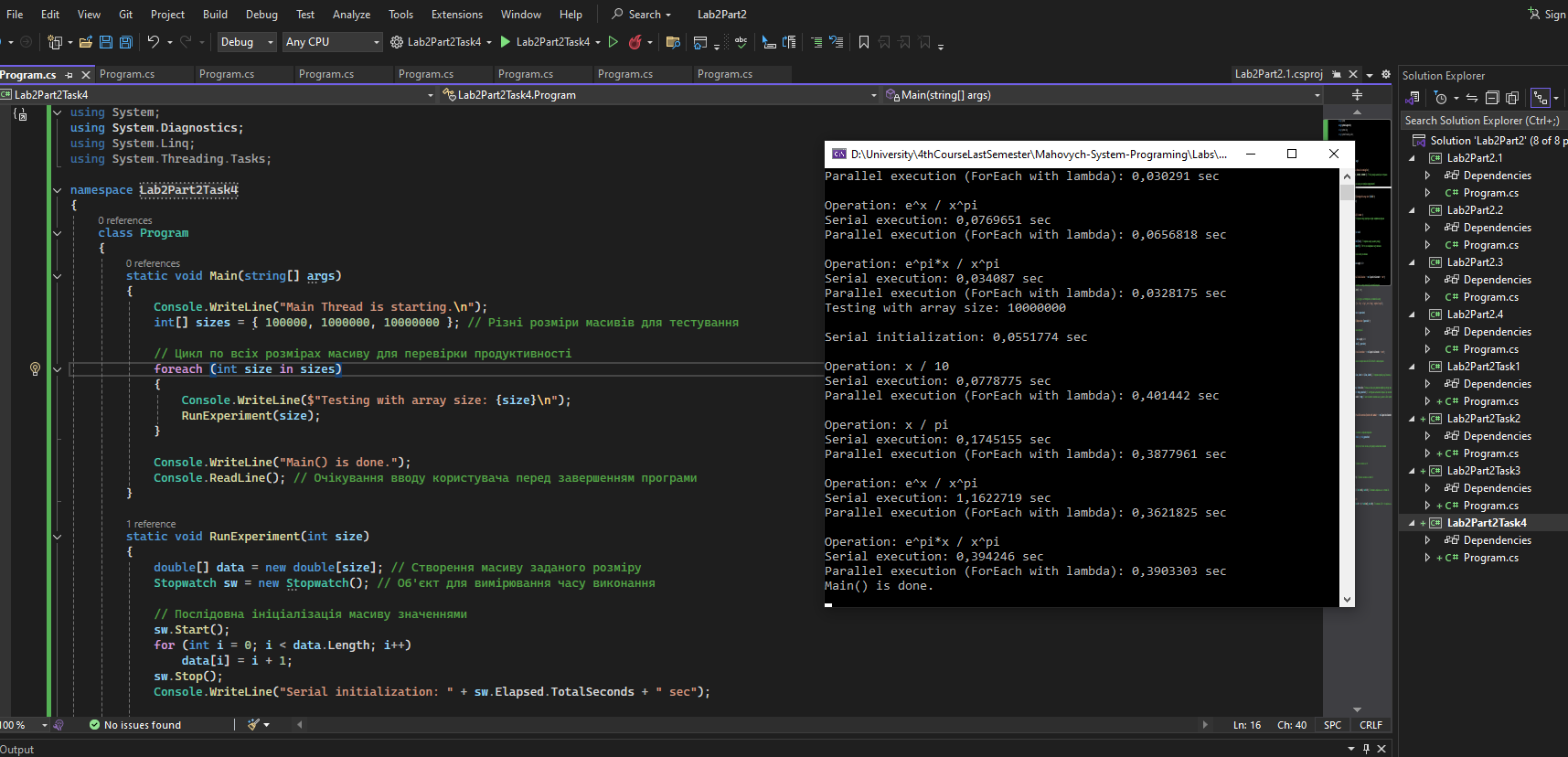
Parallel execution (ForEach): 0,1540405 sec

Operation: e^pi\*x / x^pi

Serial execution: 0,2340877 sec

Parallel execution (ForEach): 0,1376639 sec

Main() is done.

**5 МОДИФІКУВАТИ ПОПЕРЕДНЮ ПРОГРАМУ, ЩОБ ТІЛО ПАРАЛЕЛЬНОГО ЦИКЛУ ЗАДАВАЛОСЯ ЛЯМБДА-ВИРАЗОМ**

**Результат виконання:**

Main Thread is starting.

Testing with array size: **100000**

Serial initialization: 0,0007326 sec

Operation: x / 10

Serial execution: 0,0009791 sec

Parallel execution (ForEach with lambda): 0,0302054 sec

Operation: x / pi

Serial execution: 0,0013118 sec

Parallel execution (ForEach with lambda): 0,0155398 sec

Operation: e^x / x^pi

Serial execution: 0,0061388 sec

Parallel execution (ForEach with lambda): 0,0173987 sec

Operation: e^pi\*x / x^pi

Serial execution: 0,0035665 sec

Parallel execution (ForEach with lambda): 0,0110788 sec

Testing with array size: **1000000**

Serial initialization: 0,0069664 sec

Operation: x / 10

Serial execution: 0,0084871 sec

Parallel execution (ForEach with lambda): 0,1036672 sec

Operation: x / pi

Serial execution: 0,0131823 sec

Parallel execution (ForEach with lambda): 0,030291 sec

Operation: e^x / x^pi

Serial execution: 0,0769651 sec

Parallel execution (ForEach with lambda): 0,0656818 sec

Operation: e^pi\*x / x^pi

Serial execution: 0,034087 sec

Parallel execution (ForEach with lambda): 0,0328175 sec

Testing with array size: **10000000**

Serial initialization: 0,0551774 sec

Operation: x / 10

Serial execution: 0,0778775 sec

Parallel execution (ForEach with lambda): 0,401442 sec

Operation: x / pi

Serial execution: 0,1745155 sec

Parallel execution (ForEach with lambda): 0,3877961 sec

Operation: e^x / x^pi

Serial execution: 1,1622719 sec

Parallel execution (ForEach with lambda): 0,3621825 sec

Operation: e^pi\*x / x^pi

Serial execution: 0,394246 sec

Parallel execution (ForEach with lambda): 0,3903303 sec

Main() is done.

**Висновок:** у ході виконання цієї лабораторної роботи я працювала з паралельним виконанням задач. Я використовувала For() та ForEach(), останній – це паралельний варіант foreach(). Також я провела невеликий експеримент, в якому вимірювала час виконання операцій різної складності у масивах трьох розмірів викорисвовуючи послідовну та паралельну обробку. Паралельна обробка виявилася швидшою ніж послідовна у всіх випадках окрім одного. Далі я модифікувала цей код, щоб продемонструвати роботу Break(), ForEach() та як можливо задати тіло паралельного циклу лямбда-виразом.