Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра мережевих та інтернет технологій

Лабораторна робота №3

З дисципліни “Системне програмування”

Студента Фуркала Олександра

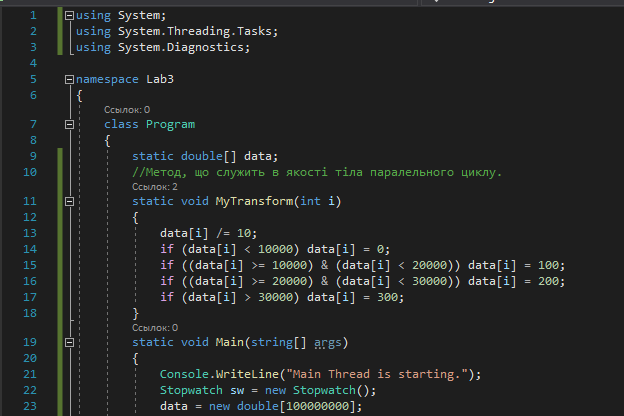
За темою:

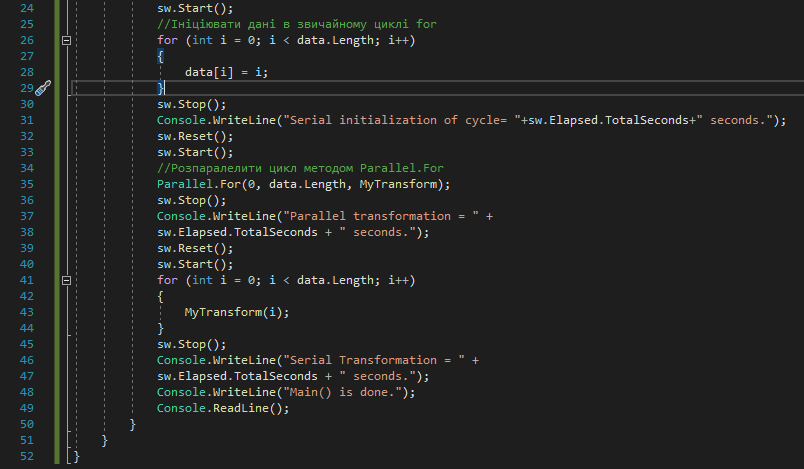
“Паралелізм даних та паралелізм задач (Ч.2)”

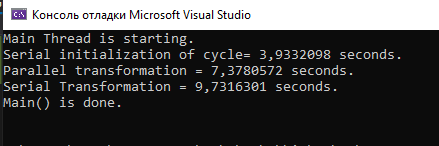
Варіант-3

Хід роботи

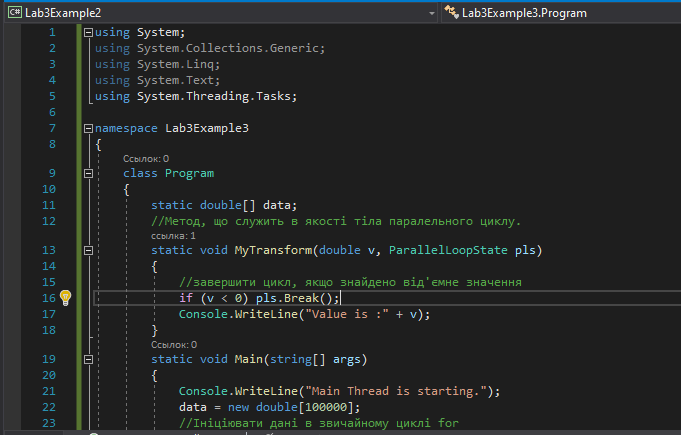
1. Повторити задачі

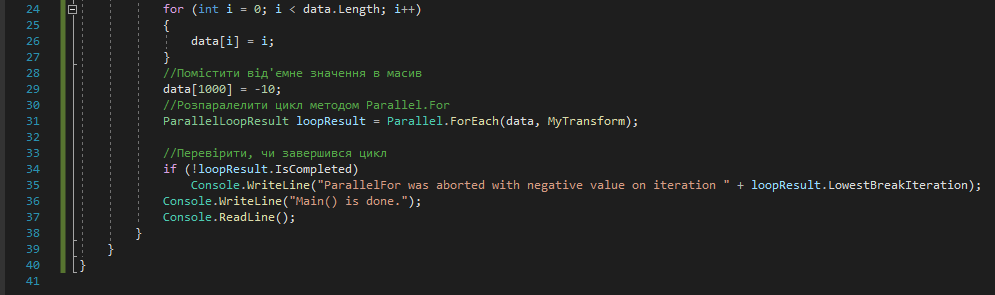


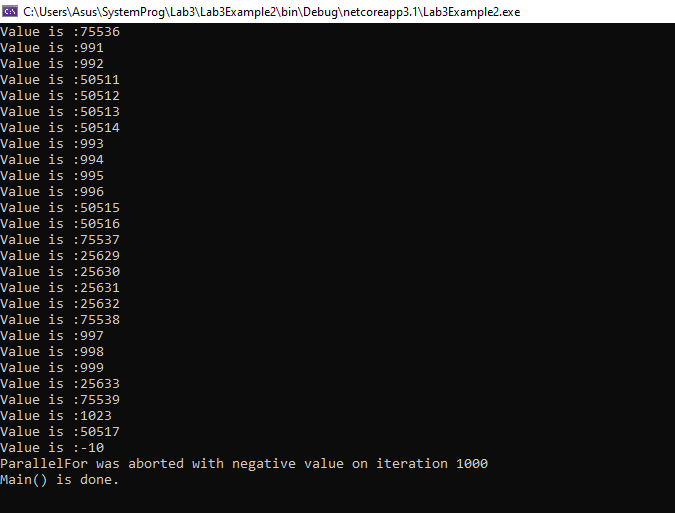




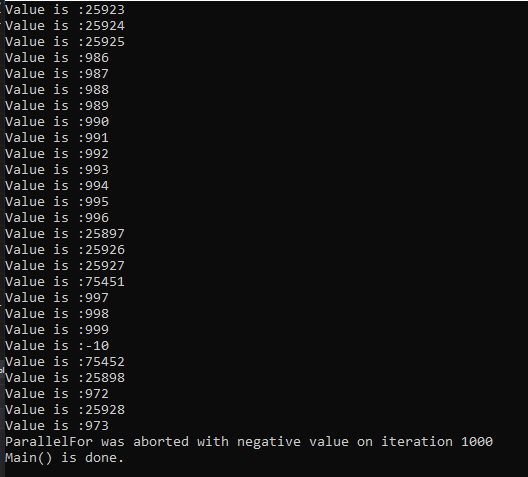
Ініціалізуємо дані data як масив із 100,000,000 елементів (використовуємо Stopwatch для вимірювання часу, цикл for для створення). Потім ми порівнюємо паралельну трансформацію (Parallel.For(0, data.Length, MyTransform), розподілення роботи між кількома ядрами ЦП) з послідовною (використання стандартного циклу). ReadLine гарантує, що вікно консолі не закриється одразу після виконання.





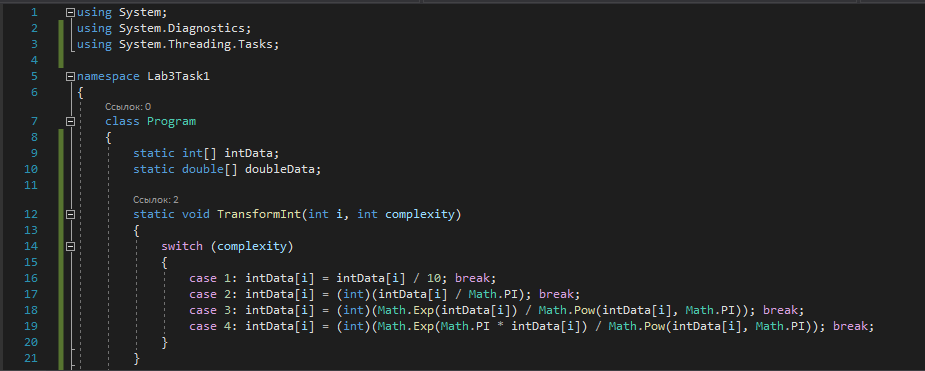


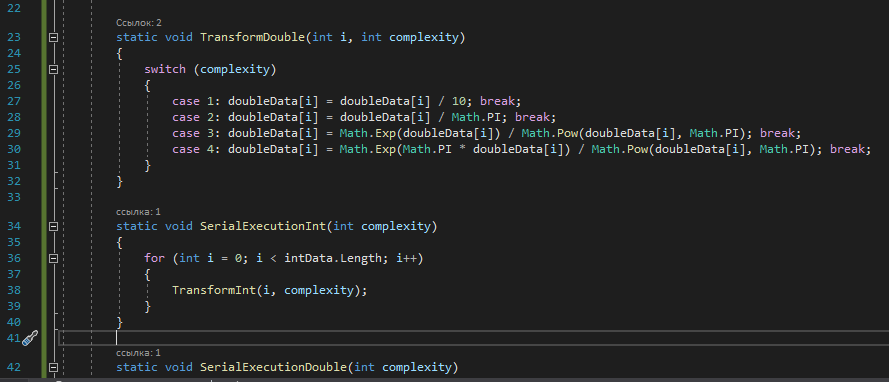
If (v<0) pls.Break перевіряє, чи знайдено від’ємне число, та якщо так, викликає pls.Break, сигналізуючи всім іншим потокам припинити виконання. pls.Break не відразу зупиняє всі ітерації, але дає вказівку припинити планування нових ітерацій поза поточною. Приклад:

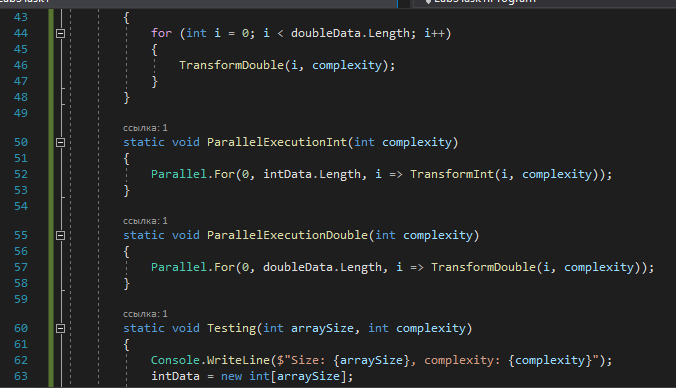


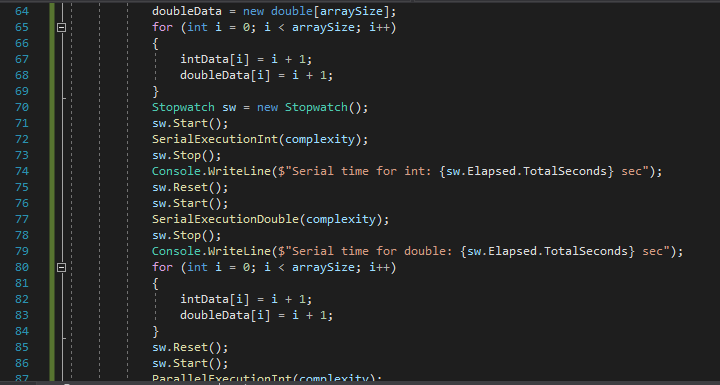
ParallelLoopResult loopResult = Parallel.ForEach(data, MyTransform) обробляє масив даних паралельно, викликаючи MyTransform для кожного елемента. Parallel.ForEach ділить масив даних на менші частини, і кожне ядро ЦП отримує окремий фрагмент для одночасної обробки, та кілька потоків працюють одночасно, перетворюючи різні елементи через MyTransform. Коли (або якщо) цикл перерветься завчасно (ParallelLoopState.Break), LowestBreakIteration буде містити найменше значення змінної управління циклом (найраніший індекс, коли було запущено Break)

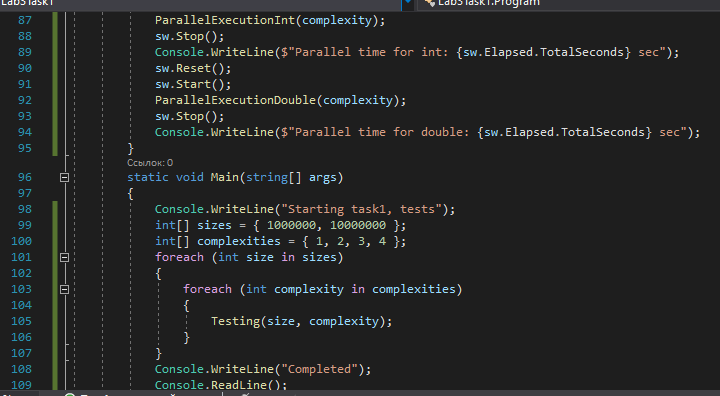
1. Створити програму, що використовує для розпаралелювання метод Parallel.For(). Провести ряд обчислювальних експериментів із різним типом масиву (int, double); різною кількістю елементів масиву; різною складністю обчислень ( , , ,). Виміряти час, що витрачається паралельним способом обробки та послідовним.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

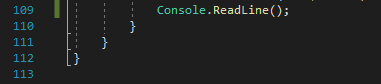


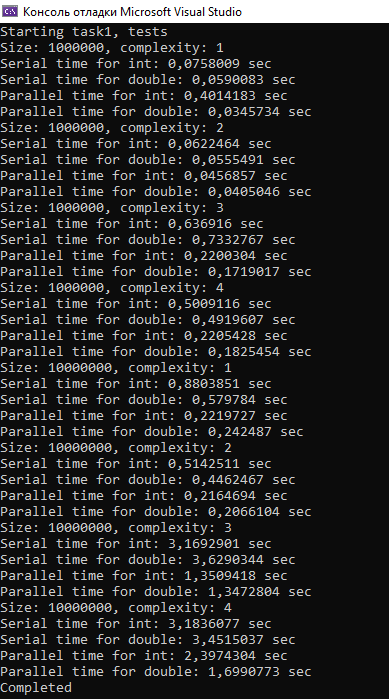












Для 1,000,000 елементів (в секундах)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Complexity 1 | Complexity 2 | Complexity 3 | Complexity 4 |
| Serial(int) | 0,0758009 | 0,0622464 | 0,636916 | 0,5009116 |
| Serial(double) | 0,0590083 | 0,0555491 | 0,7332767 | 0,4919607 |
| Parallel(int) | 0,4014183 | 0,0456857 | 0,2200304 | 0,2205428 |
| Parallel(double) | 0,0345734 | 0,0405046 | 0,1719017 | 0,1825454 |

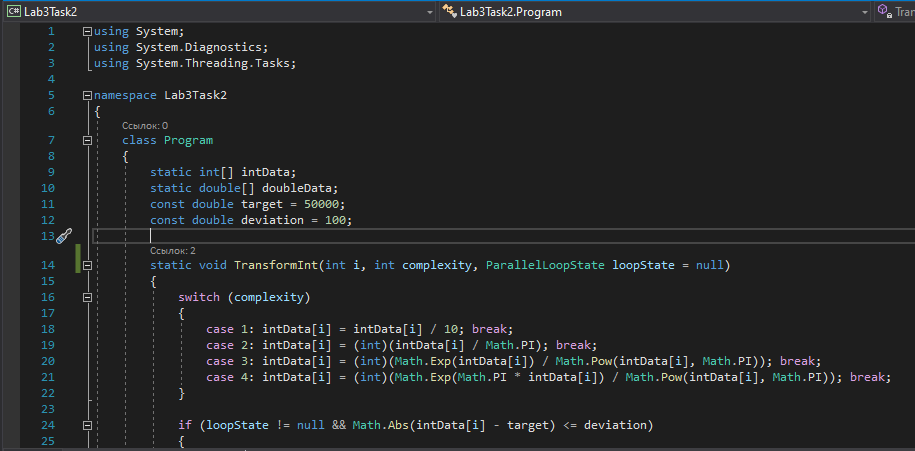
Для 10,000,000 елементів (в секундах)

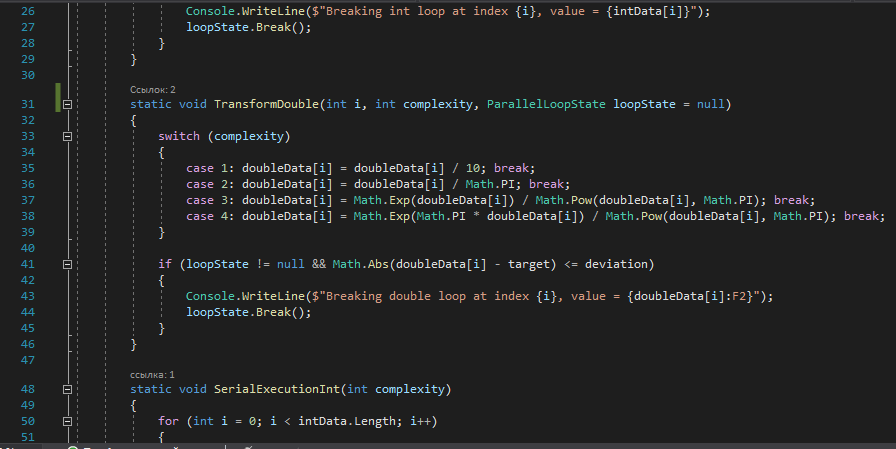
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Complexity 1 | Complexity 2 | Complexity 3 | Complexity 4 |
| Serial(int) | 0,8803851 | 0,5142511 | 3,1692901 | 3,1836077 |
| Serial(double) | 0,579784 | 0,4462467 | 3,6290344 | 3,4515037 |
| Parallel(int) | 0,2219727 | 0,2164694 | 1,3509418 | 2,3974304 |
| Parallel(double) | 0,242487 | 0,2066104 | 1,3472804 | 1,6990773 |

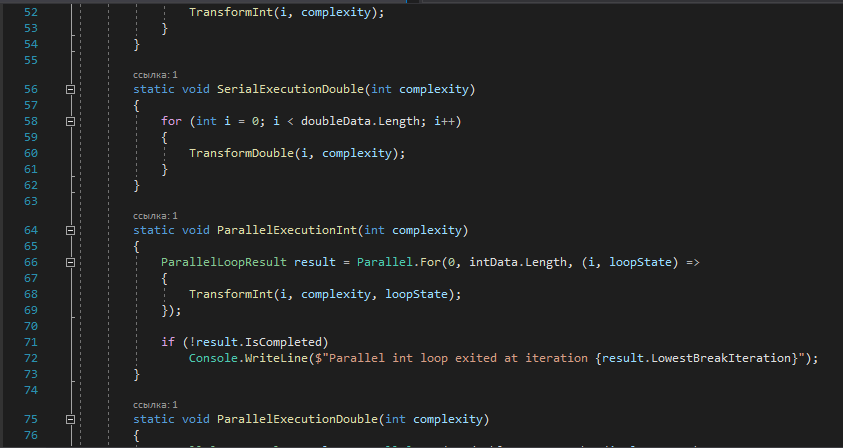
Цікаво, що за підсумковими результатами з цим обсягом даних розрахунки з double зазвичай швидші ніж розрахунки з int. Можливо, це через те, що цілочисельне ділення вимагає обчислення залишку, або обчислення int надто прості, і накладні витрати на синхронізацію більші, ніж обчислення.

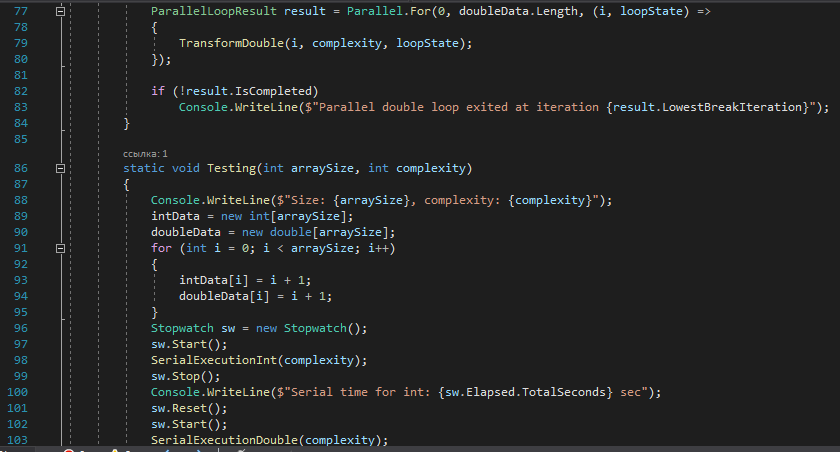
Через Main запускаємо Testing для обраних sizes (розмір) та complexities (складність обчислень), Testing робить розрахунки з SerialInt, SerialDouble, ParallelInt та ParallelDouble.

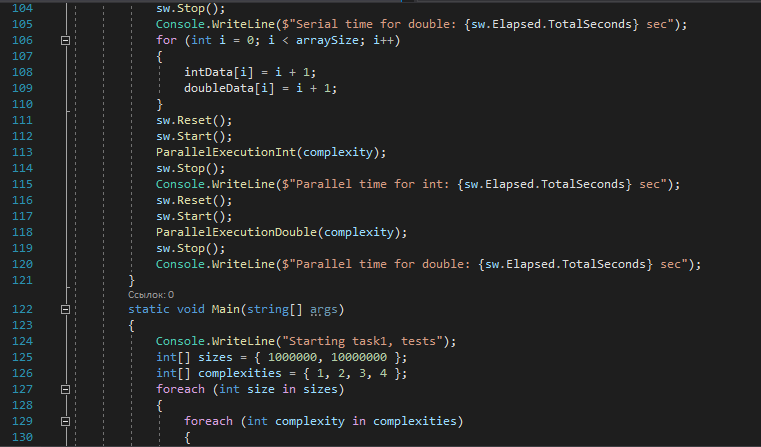
1. Модифікувати попередню програму (створивши новий проект у рішенні) таким чином, щоб відбувався вихід з паралельного циклу за умови входження значення елемента у деякий окіл деякого числа (число та відхилення задаються константами).

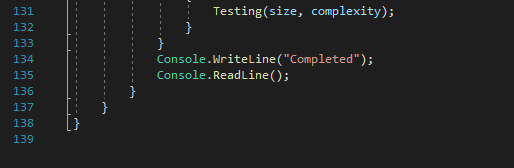


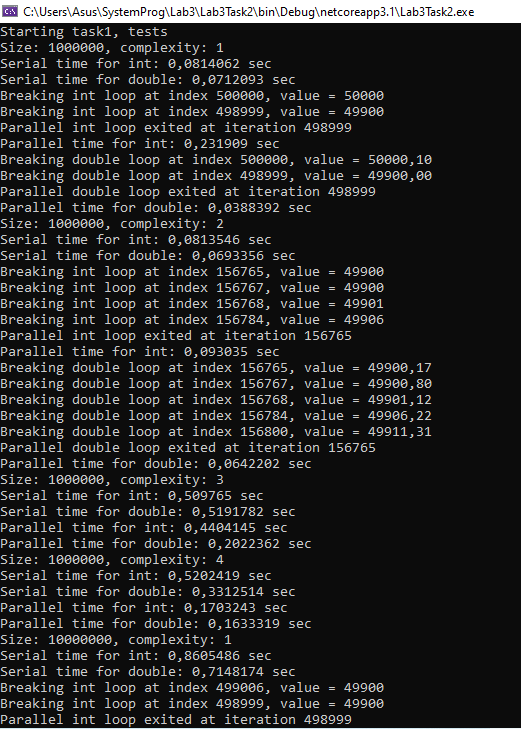


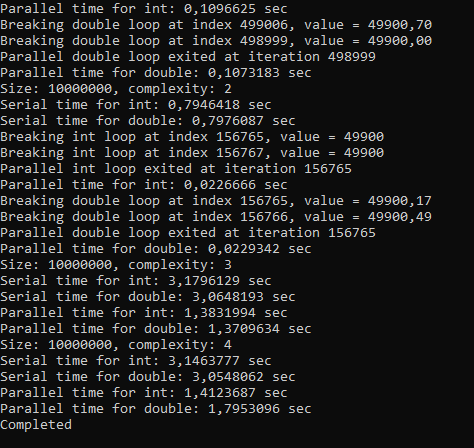










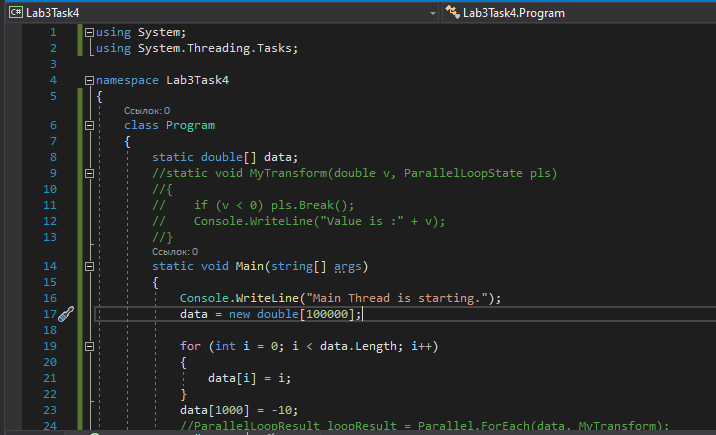


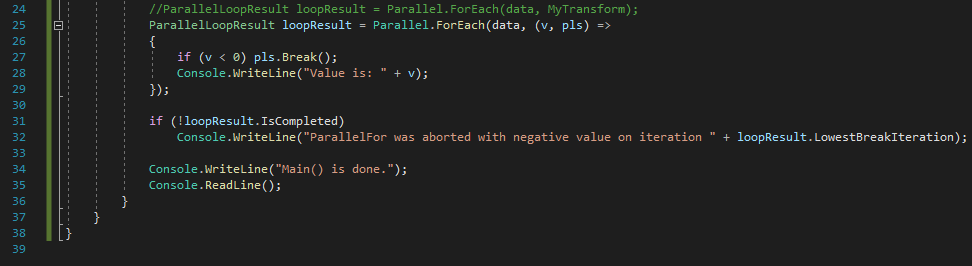
ParallelLoopState loopState буде null, коли ми використовуємо Transform для Serial, та для Parallel використовується частина з target та deviation (число та відхилення). Якщо data знаходиться між 50000-50=49950 або 50000+50=50050, цикл завершується раніше.

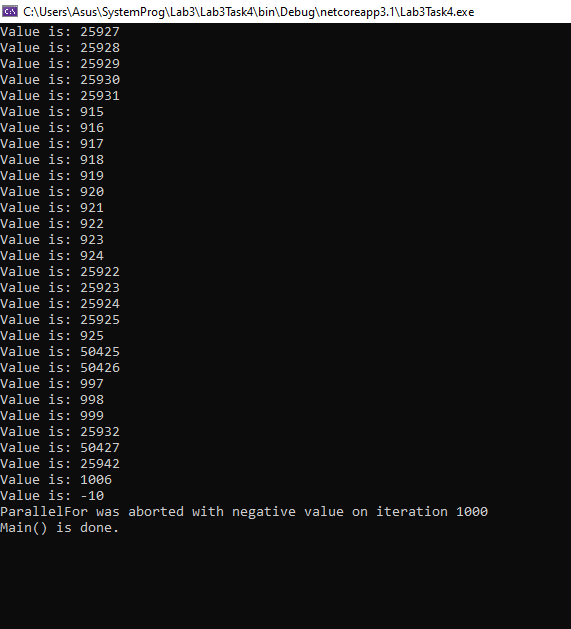
Break використовується для того, щоб паралельні завдання, які вже виконуються, були завершені, але нові ітерації після поточної мають бути зупинені. LowestBreakIteration допомагає відстежувати, де відбувся вихід.

Інколи можемо побачити декілька повідомлень про Breaking the loop – це відбувається тому, що Parallel.For виконує ітерації одночасно в кількох потоках, та оскільки кілька потоків одночасно працюють над різними частинами масиву, деякі потоки все ще можуть обробляти значення, близькі до умови зупинки.

1. Повторити приклади розпаралелювання за допомогою ForEach()
2. Модифікувати попередню програму (створивши новий проект у рішенні) таким чином, щоб тіло паралельного циклу задавалося лямбда-виразом







1. Оформити звіт