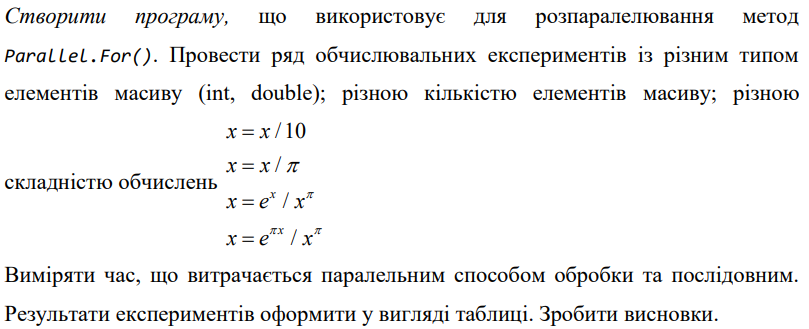
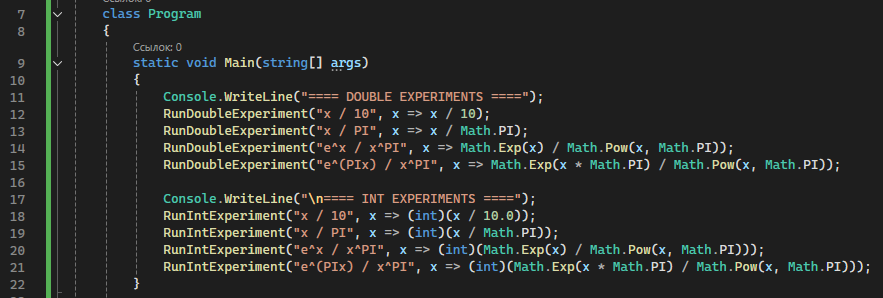
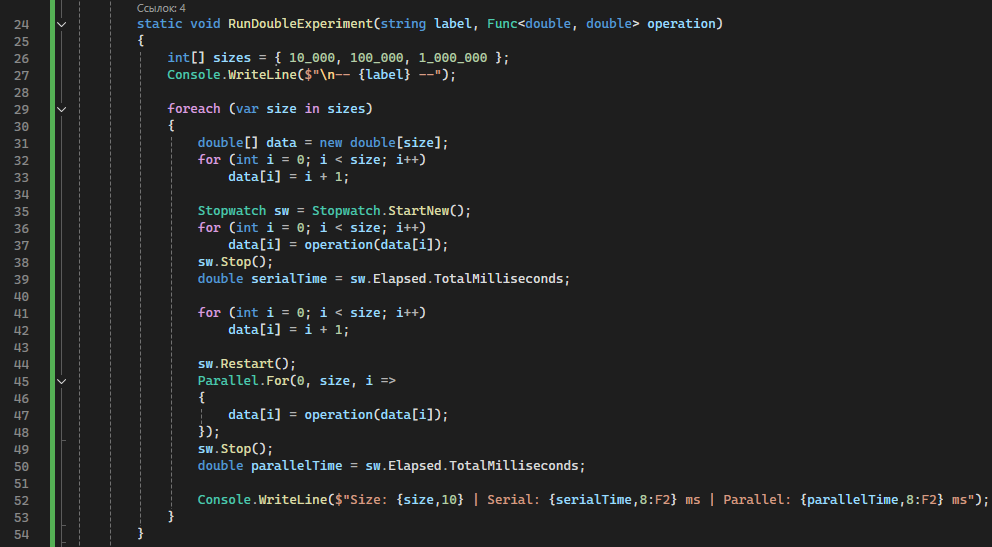
Лабораторна робота №2

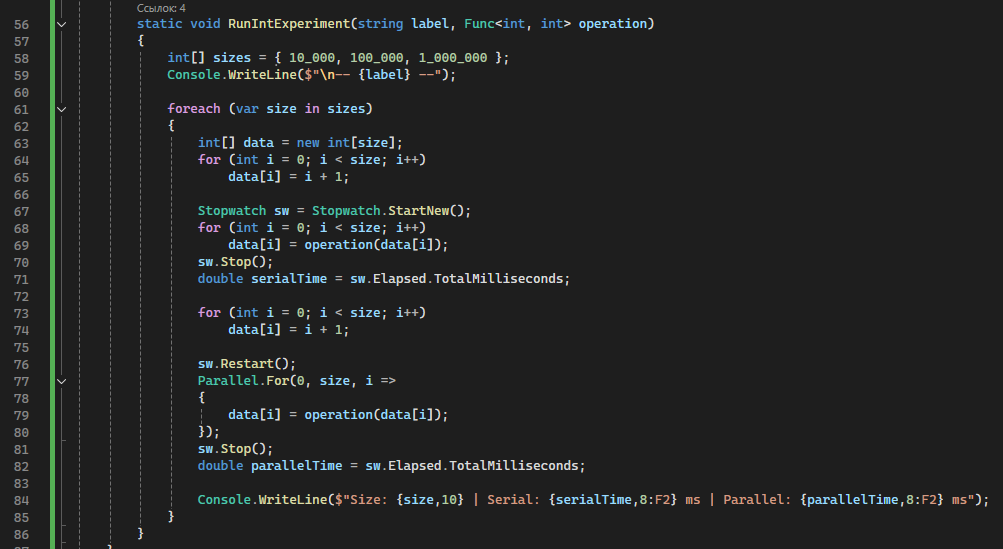
ПАРАЛЕЛІЗМ ДАНИХ ТА ПАРАЛЕЛІЗМ ЗАДАЧ (Ч. 2)

1. 

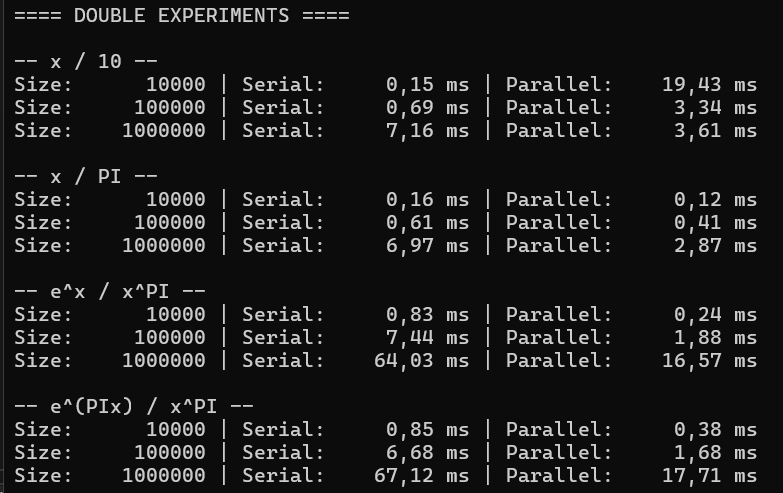
Код програми:

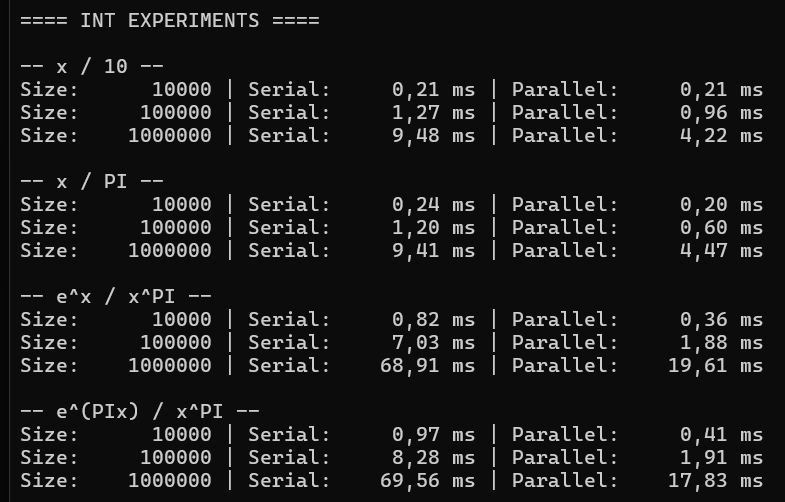




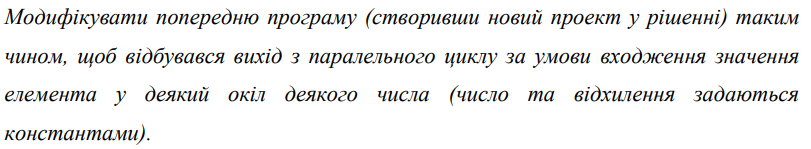


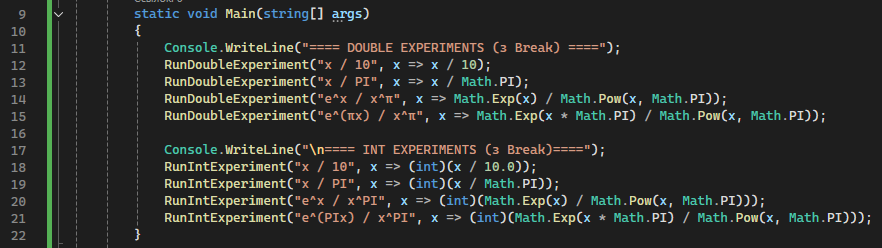
Результат виконання:

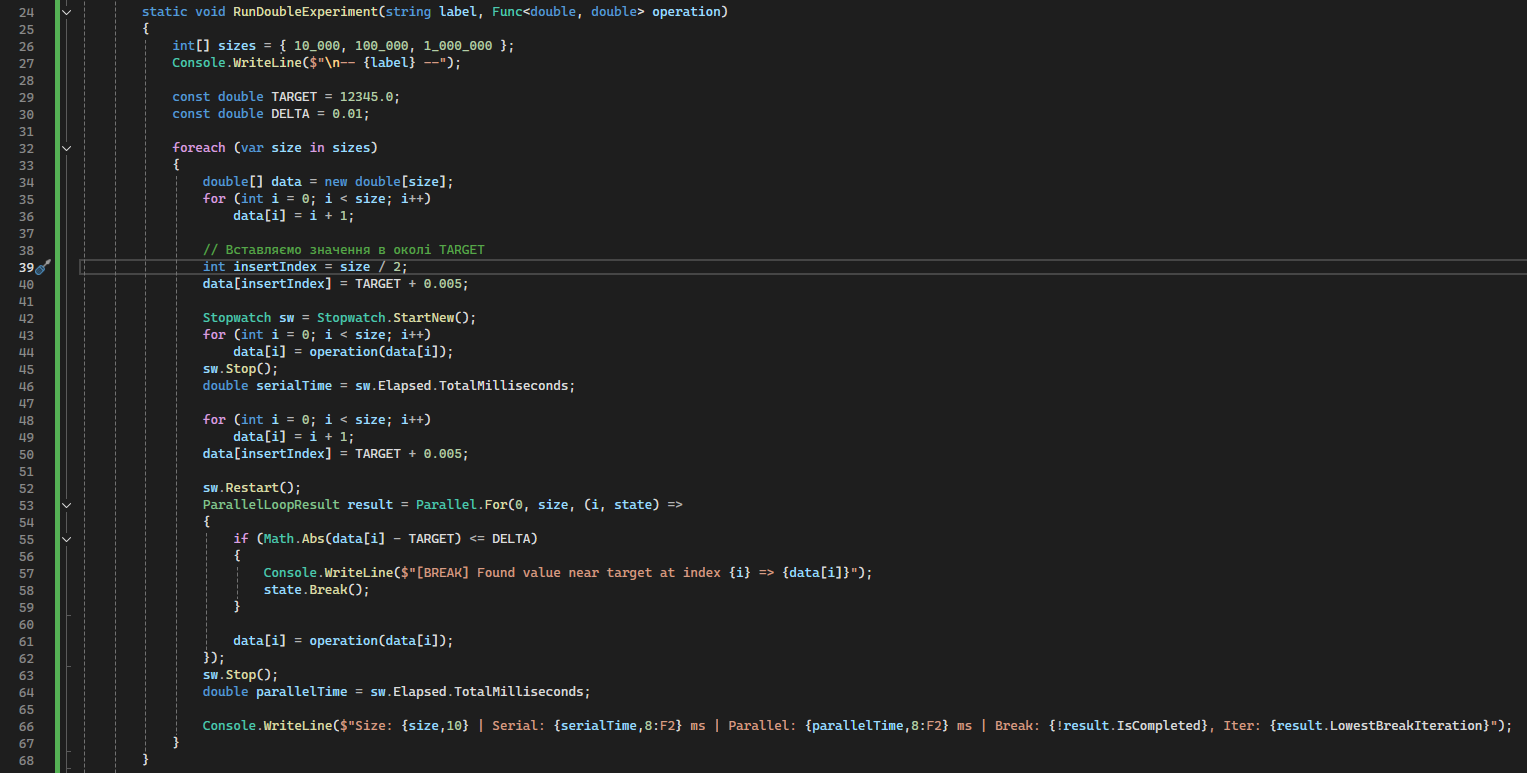


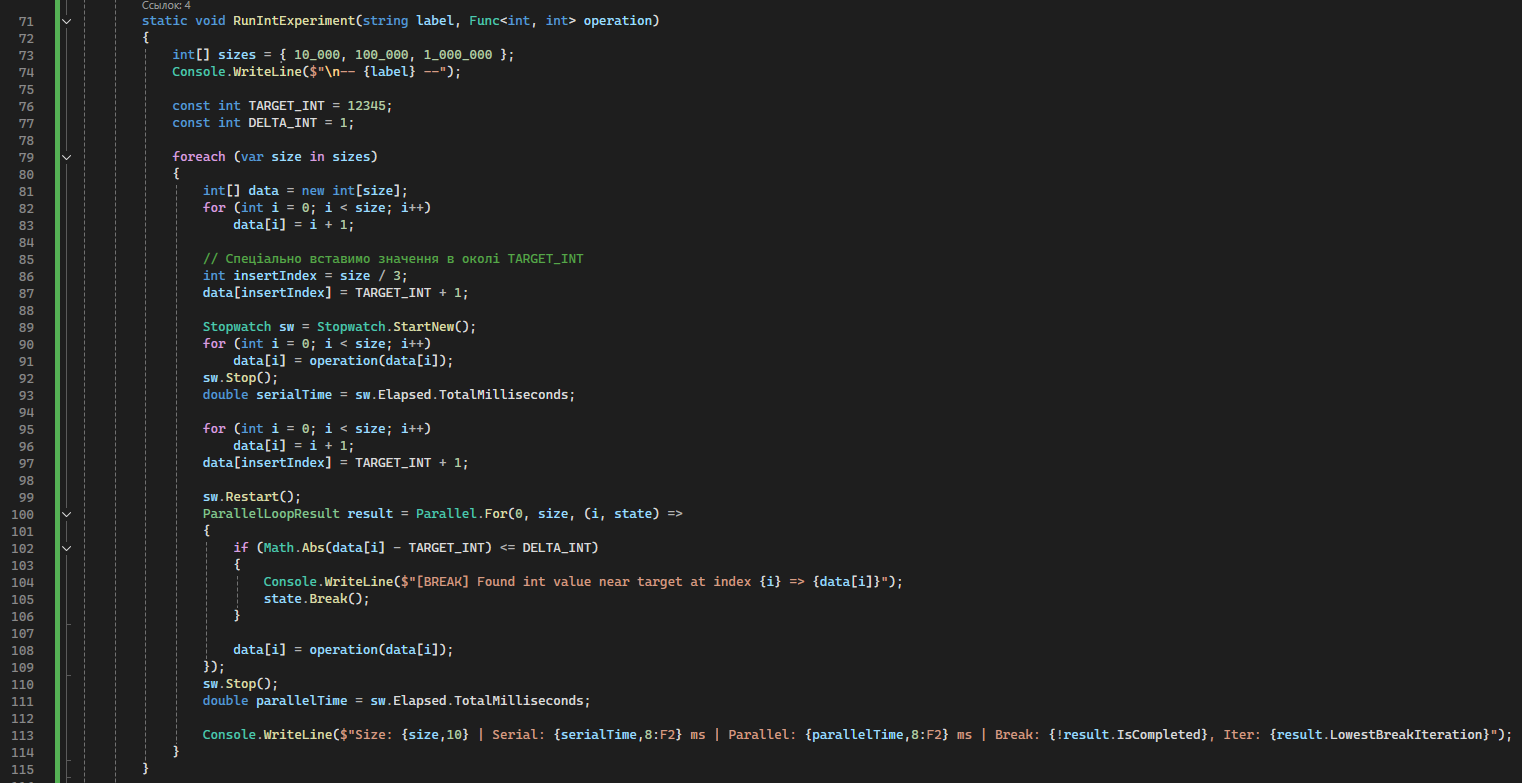


Висновок: паралелізм дає суттєвий приріст продуктивності при великих об’ємах даних і складних обчисленнях. Для простих операцій і малих масивів `Parallel.For()` часто повільніший через накладні витрати. Найбільше прискорення спостерігається при використанні складних формул (e^x / x^π тощо) на масивах від 100 000 елементів. Тому паралелізацію варто застосовувати лише тоді, коли вона справді виправдана.

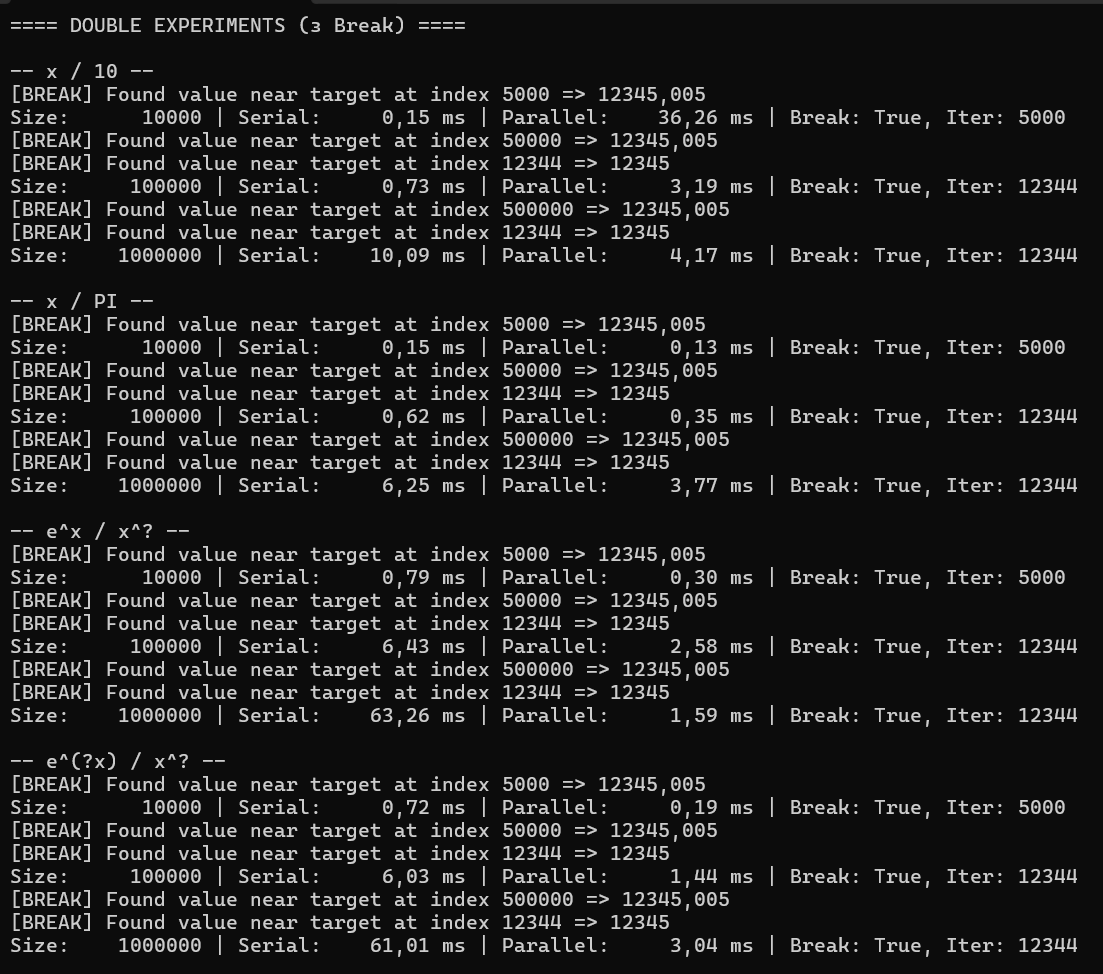
1. 

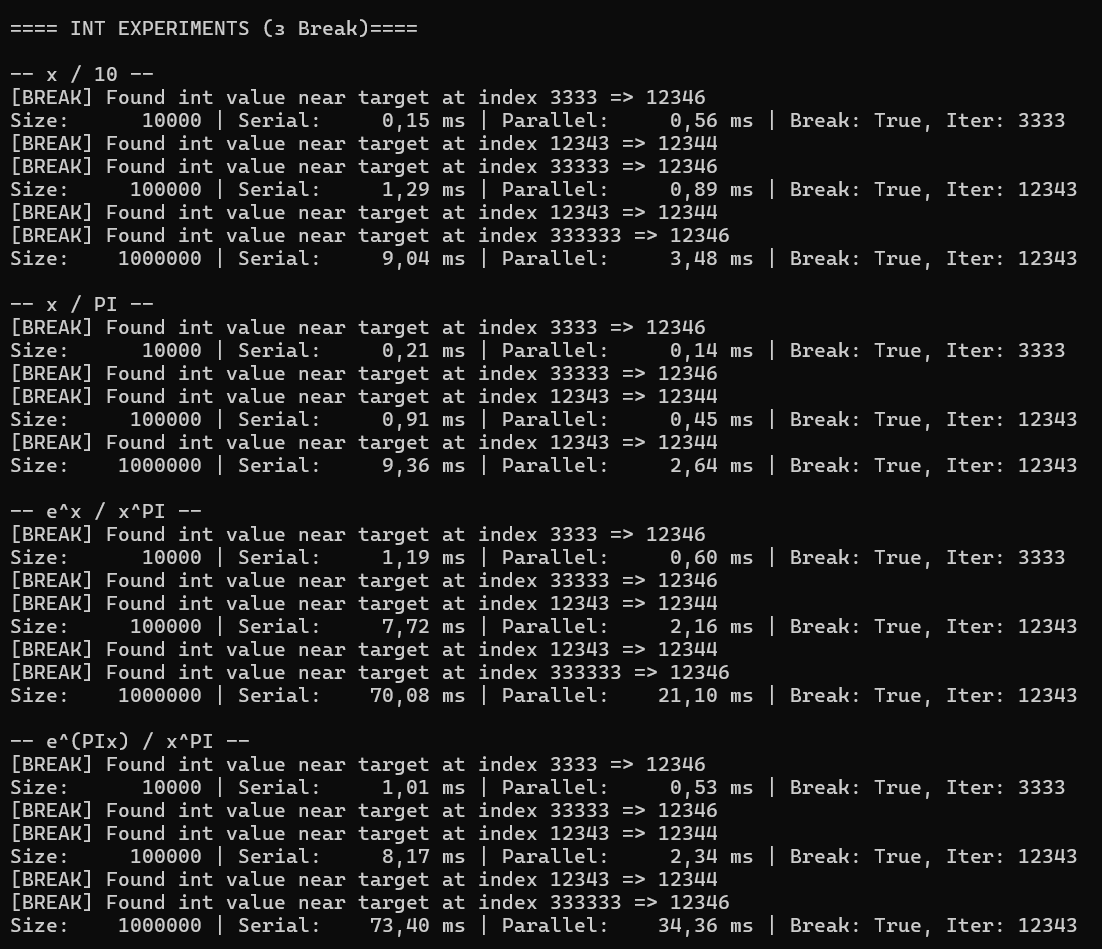






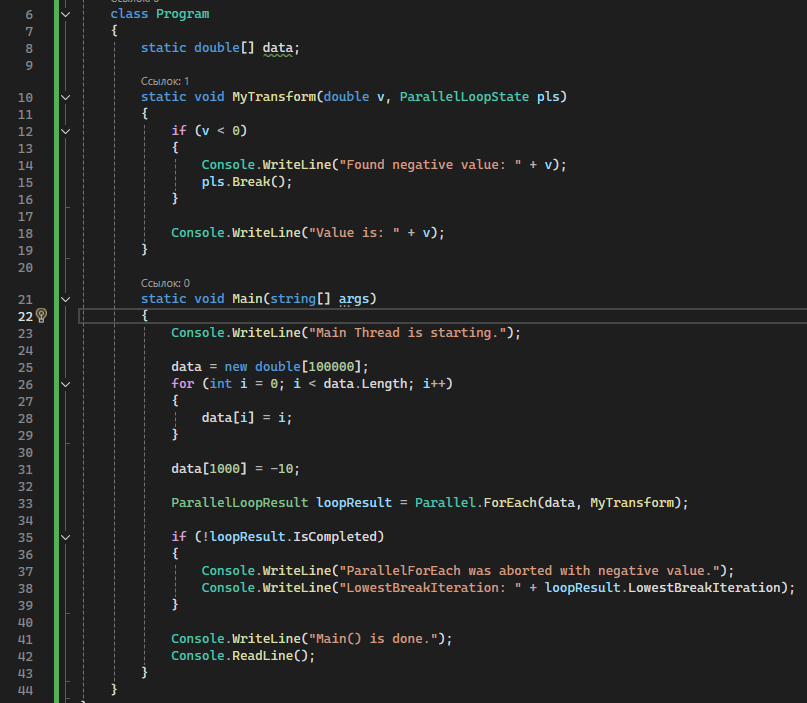
Результат виконання:



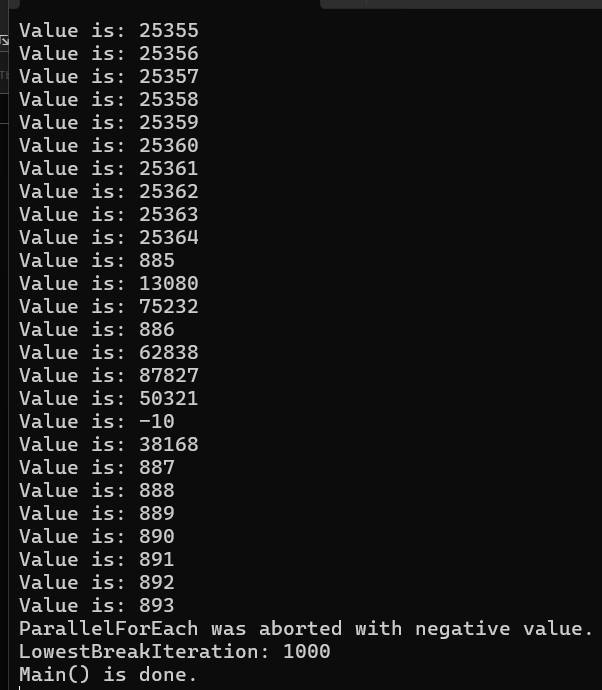


У програмі виконується обробка масивів типу double та int за допомогою методу Parallel.For, з умовним достроковим завершенням циклу. Для кожного елемента перевіряється, чи потрапляє його значення в околицю заданого числа з допустимим відхиленням. Якщо така умова виконується, цикл переривається через виклик state.Break(). Після завершення виводиться інформація про те, чи був цикл повністю виконаний, і на якій ітерації він був перерваний, використовуючи властивість LowestBreakIteration. Це демонструє можливість контролю виконання паралельного циклу на основі певної умови.

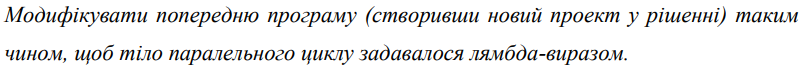
1. 



Результат виконання:

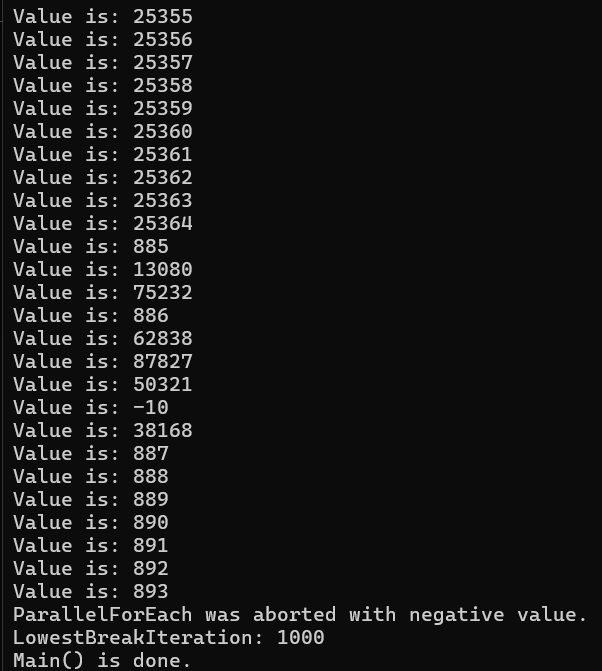


У третьому завданні ми реалізуємо паралельну обробку масиву за допомогою методу Parallel.ForEach, який проходить по елементах колекції. При знаходженні елемента, що задовольняє умову (наприклад, від’ємного значення), цикл достроково завершується за допомогою Break(), що демонструє можливість контролю виконання паралельного перебору.

1. 



Результат виконання ідентичний до попереднього:



У цьому завданні ми модифікуємо попередній приклад з `Parallel.ForEach`, щоб замість окремого методу використовувати лямбда-вираз як тіло паралельного циклу. Це дозволяє компактніше та зручніше записати логіку обробки елементів без створення додаткових методів.

Висновок: у цій лабораторній роботі ми дослідили можливості паралельної обробки даних у C# за допомогою методів Parallel.For і Parallel.ForEach. Ми порівняли час виконання послідовної та паралельної обробки для масивів різного типу та складності обчислень, продемонстрували дострокове завершення циклу при виконанні певної умови, а також використали лямбда-вирази як тіло паралельного циклу. Отримані результати показали, що паралелізм доцільно застосовувати при великих обсягах даних і складних обчисленнях, де він забезпечує суттєве прискорення обробки.