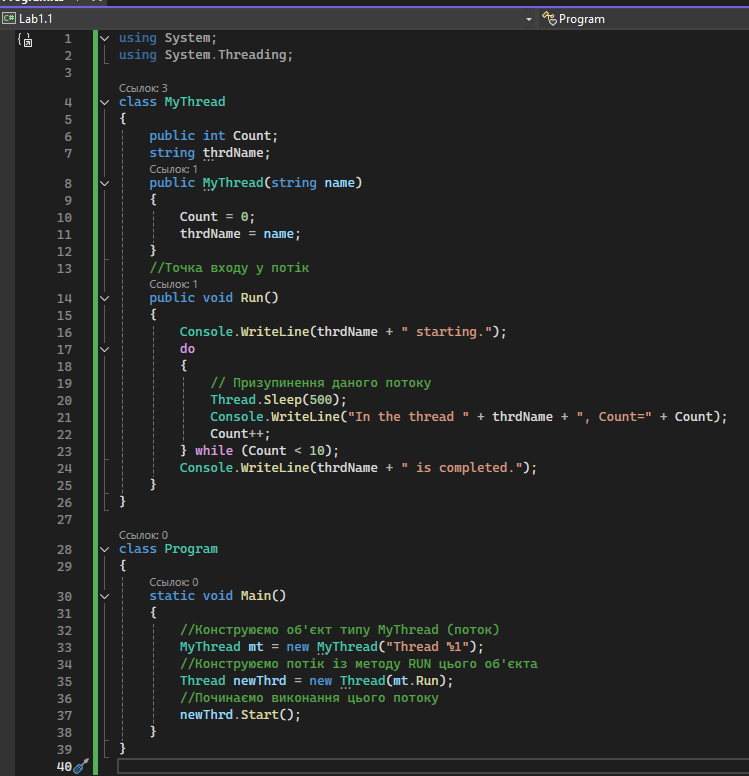
Лабораторна робота №1

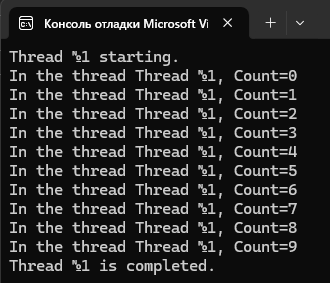
СТВОРЕННЯ ПОТОКІВ. КЛАС THREAD. ПРІОРИТЕТНІ ТА

ФОНОВІ ПОТОКИ. ПРІОРИТЕТИ ПОТОКІВ

1. Повторити задачі, розв’язані у тексті.
   1. Cтворення і запуск потоку.

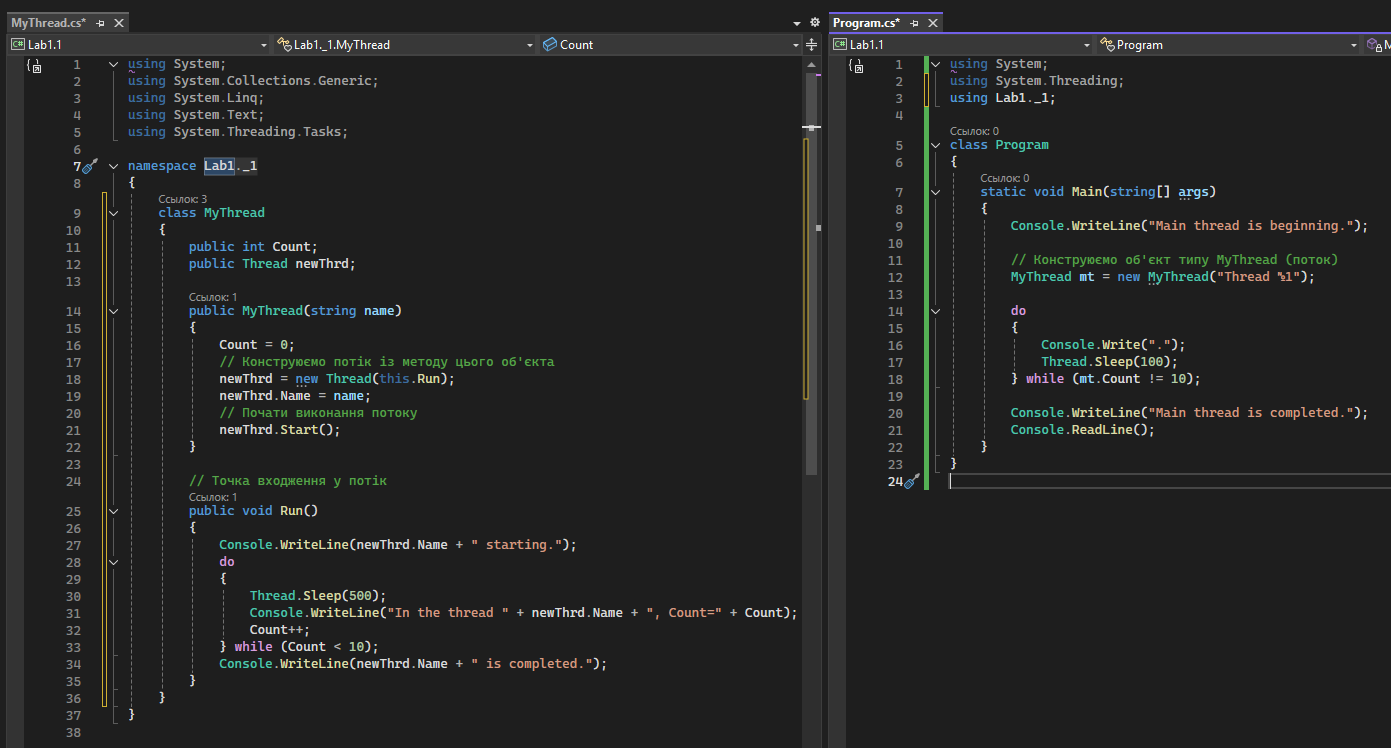


Результат виконання:

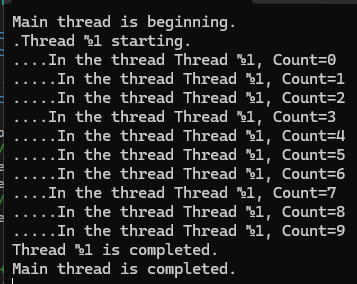


* 1. Удосконалення багатопотокової програми.

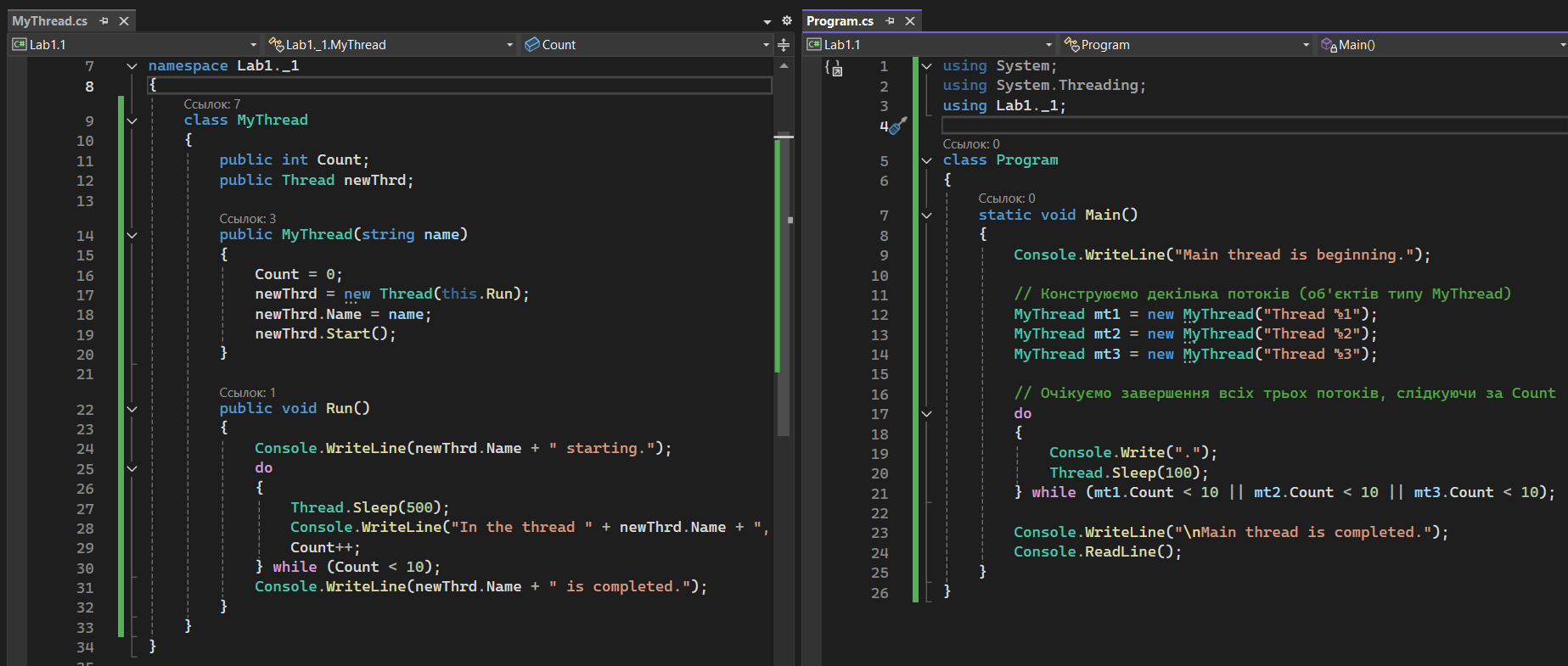
Для зручності винесемо класс MyThread в окремий файл.



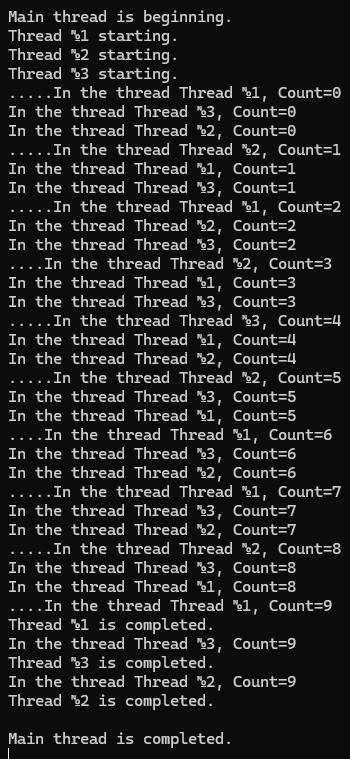
Результат виконання:



* 1. Створення декількох потоків.

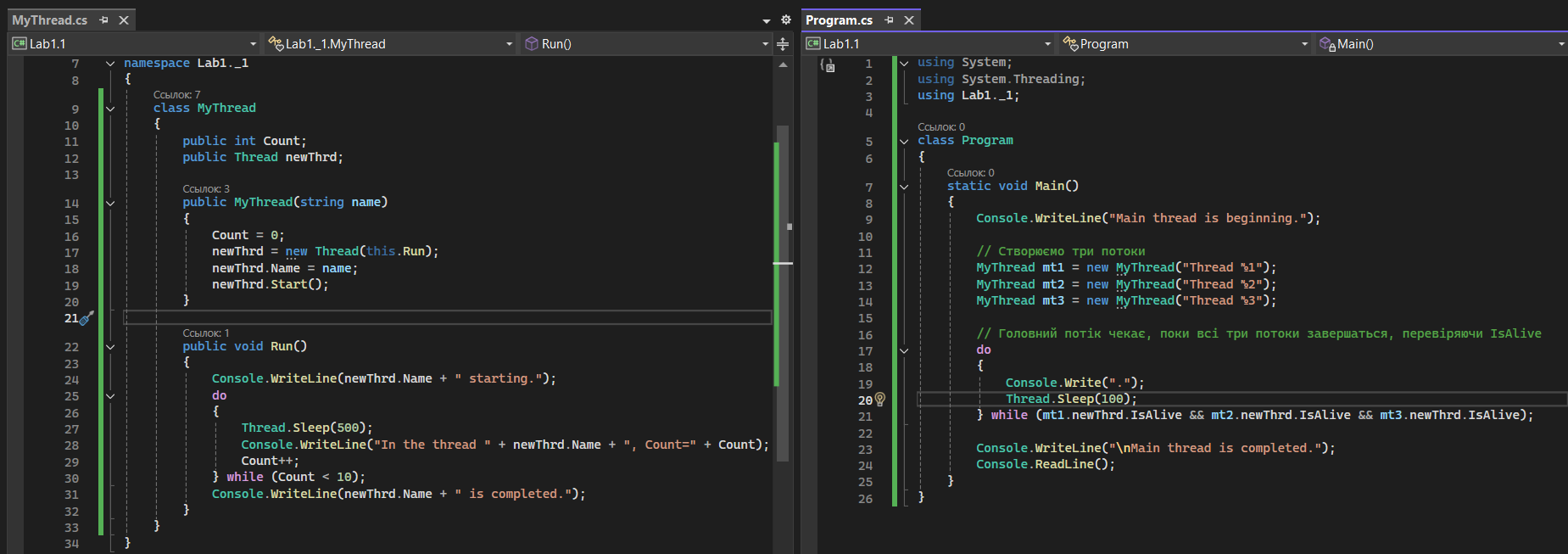


Результат виконання:



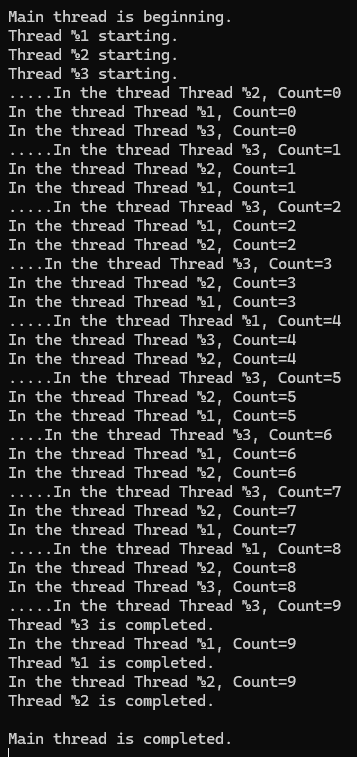
Три потоки (Thread №1, Thread №2, Thread №3) стартують одразу. Кожен з них виводить по 10 рядків Count = 0...9, з паузами по 500 мс. Головний потік у цей час виводить крапки . з частотою 100 мс. Всі потоки завершуються, після чого головний пише Main thread is completed.

* 1. Визначення моменту завершення потоку.
     1. Властивість IsAlive.

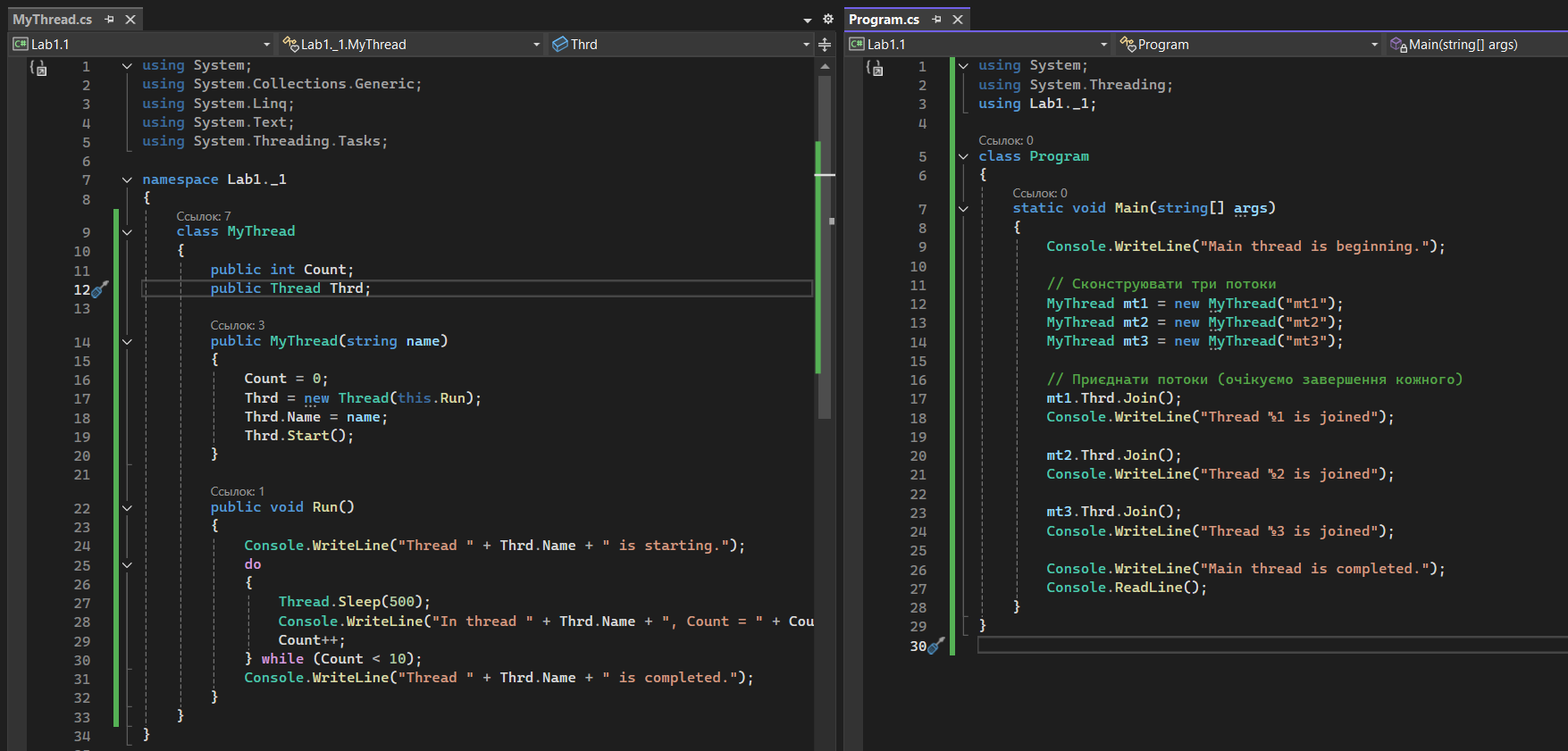


Головний потік перевіряє IsAlive у while, і виводить крапки, поки всі три потоки активні.

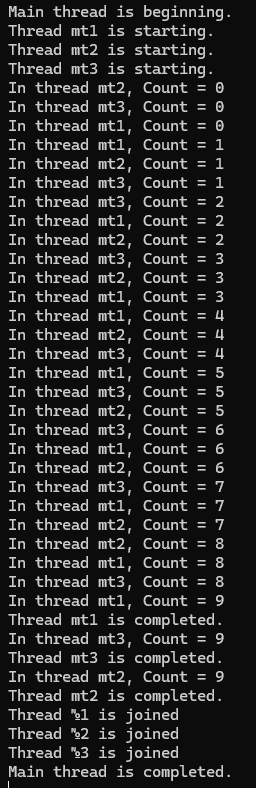
Результат виконання аналогічний до попереднього:



* + 1. Метод Join().

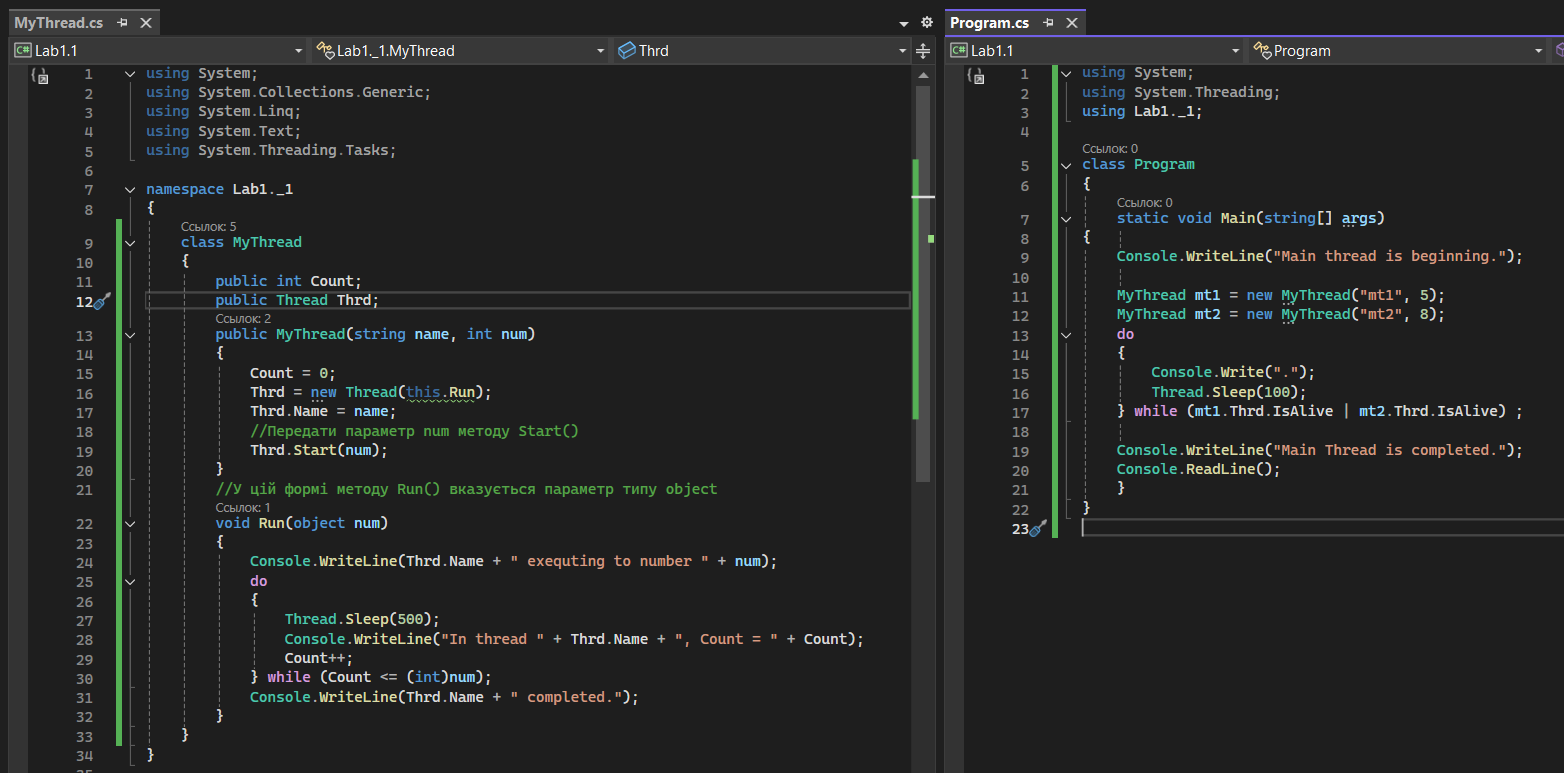


Результат виконання:

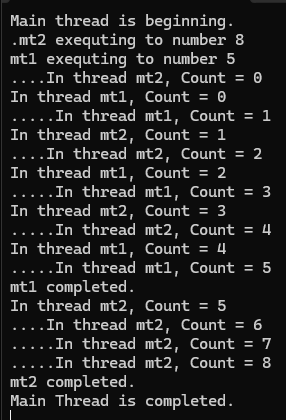


Кожен потік стартує одразу при створенні. Метод Join() блокує виконання головного потоку, доки відповідний потік не завершиться. Головний потік приєднує потоки послідовно: спершу mt1, потім mt2, потім mt3. Виводяться повідомлення: Thread №1 is joined, Thread №2 is joined тощо — після кожного завершення.

* 1. Передача аргументу потоку.

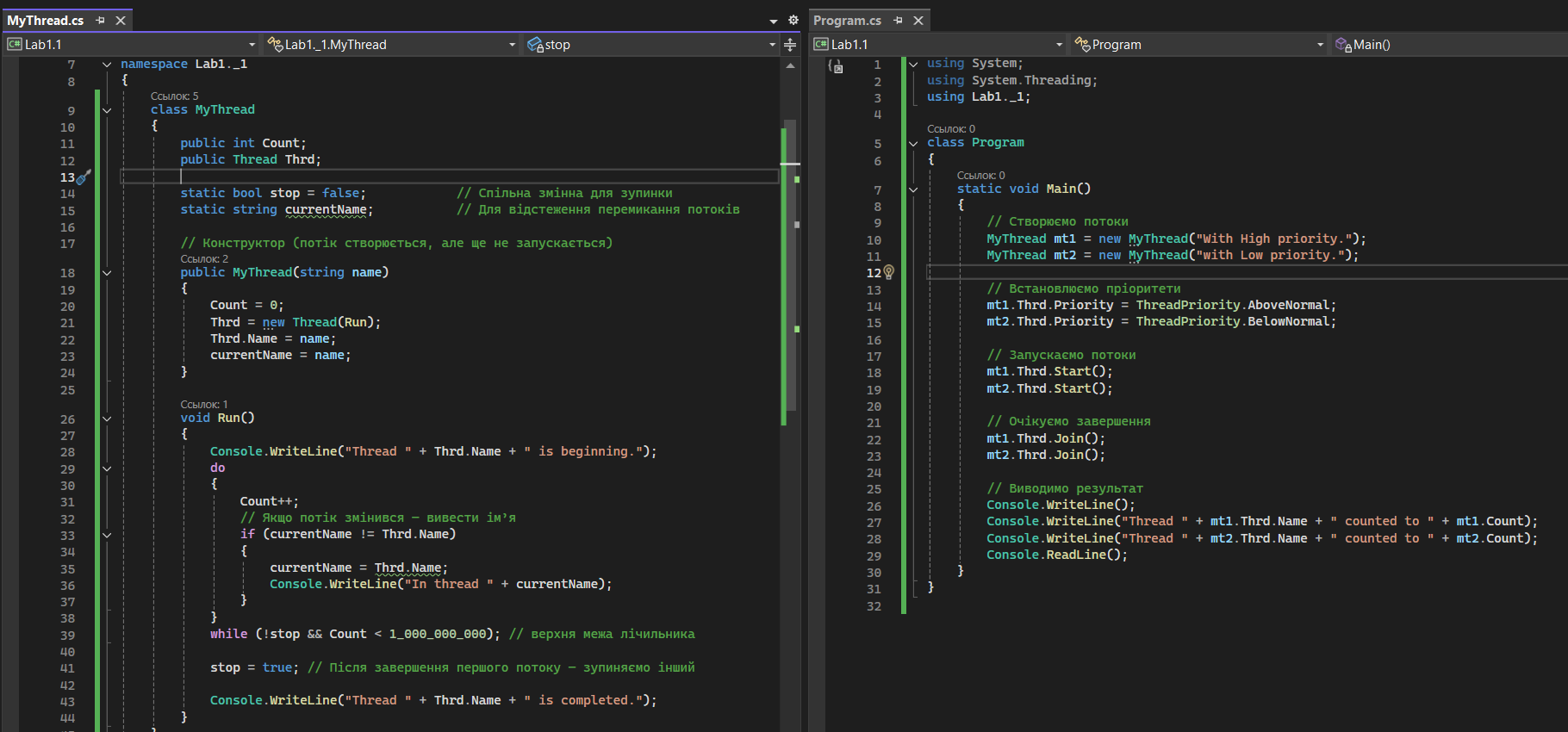


Результат виконання:

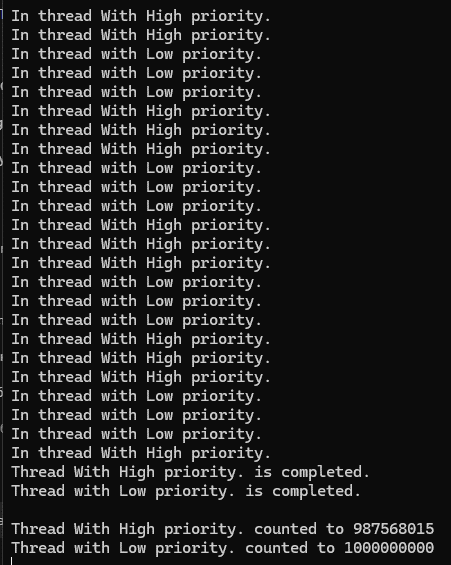


Потік mt1 виконує 5 ітерацій, тобто рахує від 0 до 5. Потік mt2 виконує 8 ітерацій — від 0 до 8. Виведення з потоків відбувається впереміш із головним потоком (.), що демонструє паралельність виконання. Це підтверджує правильну роботу передачі параметрів у потік: кожен потік отримав своє обмеження (5 і 8) через Thread.Start(object). Також видно, що завершення потоків відбувається у різний час, залежно від кількості ітерацій.

* 1. Пріоритети потоків.

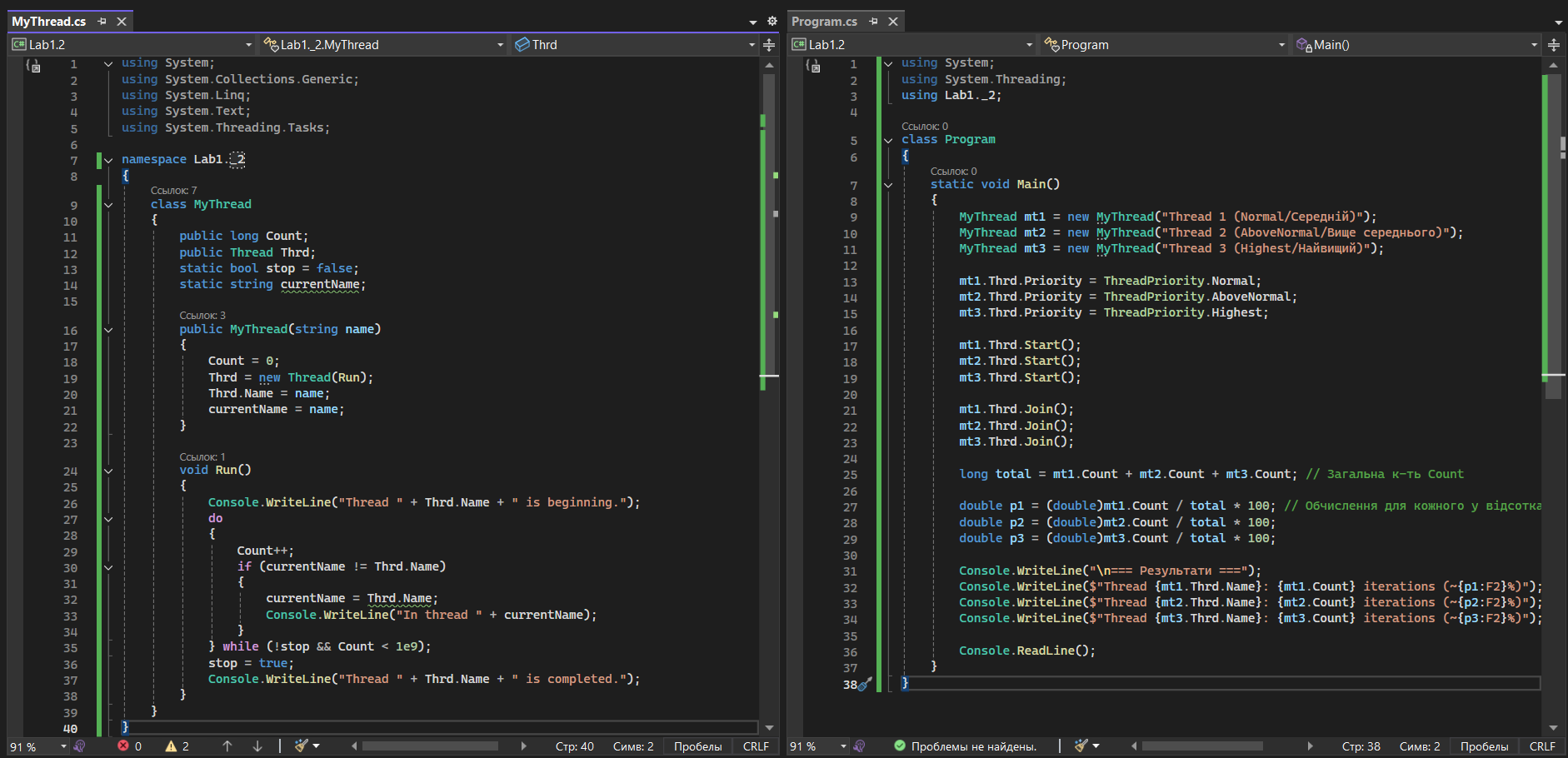


Результат виконання:

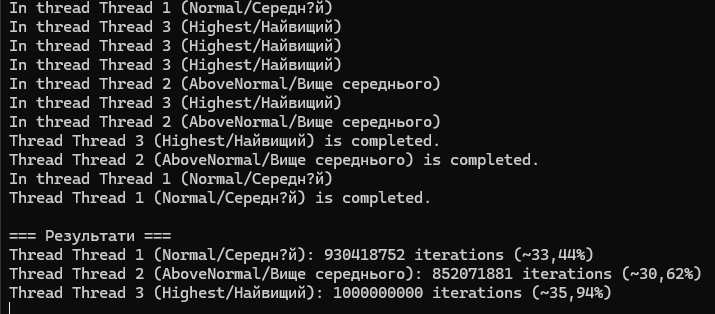


Обидва потоки рахують до 1 000 000 000, але перший, що завершує раніше, зупиняє обидва (stop = true). Потік з вищим пріоритетом (AboveNormal) виконується частіше, тому виконує значно більше ітерацій. Потік з нижчим пріоритетом (BelowNormal) отримує менше часу процесора й виконує менше ітерацій. Наприкінці ми бачимо, скільки лічильників встиг зробити кожен потік

1. Створити багатопотокову програму, задати різні пріоритети для потоків. Підрахувати розподіл процесорного часу між потоками у %. **Варіант 1**) Потоків – 3 шт. (пріоритети: середній, вище середнього, найвищий).



Результати виконання:



У результаті виконання програми найбільше ітерацій виконав потік з найвищим пріоритетом, що підтверджує вплив пріоритетів на розподіл процесорного часу. Водночас потік із середнім пріоритетом обійшов потік із пріоритетом вище середнього, що свідчить про те, що реальний розподіл CPU залежить не лише від пріоритету, а й від інших факторів, зокрема роботи планувальника операційної системи.

Висновок: у ході лабораторної роботи було вивчено створення потоків у C#, методи керування ними, передачу параметрів та вплив пріоритетів на розподіл процесорного часу.