Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра мережевих та інтернет технологій

Лабораторна робота №1

З дисципліни “Системне програмування”

Студента Фуркала Олександра

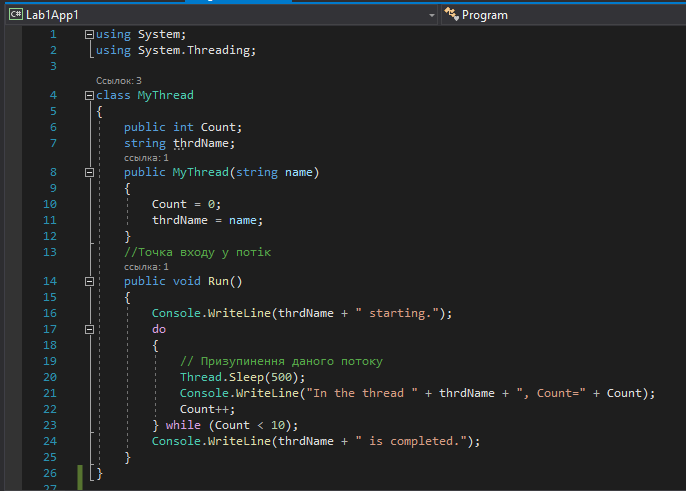
За темою:

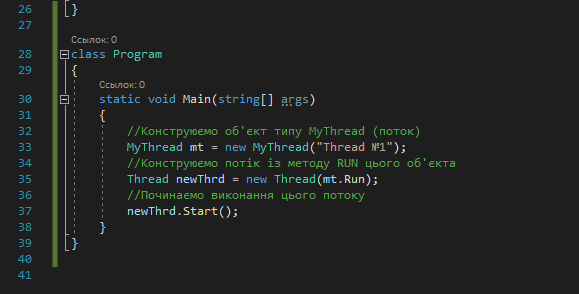
“Створення потоків. Клас Thread. Пріоритетні та фонові потоки. Пріоритети потоків”

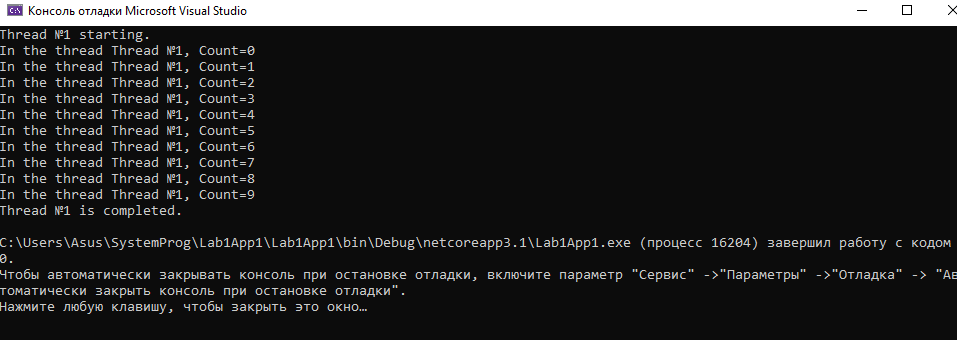
Варіант-3

Хід роботи

1. Повторити задачі

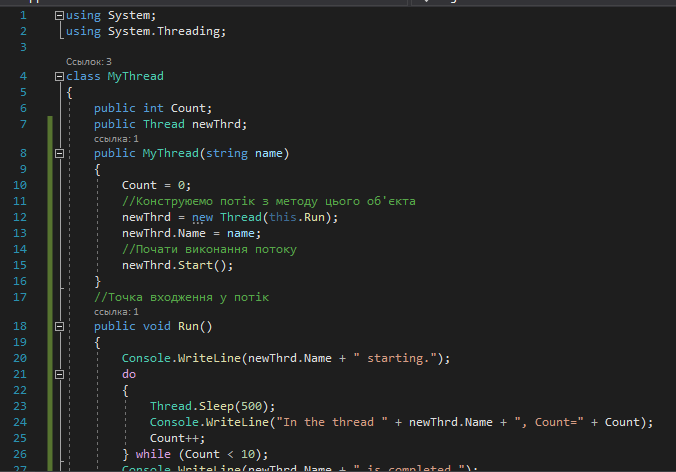


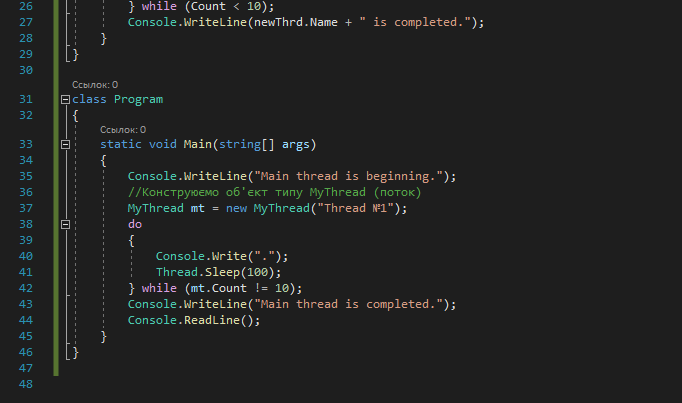


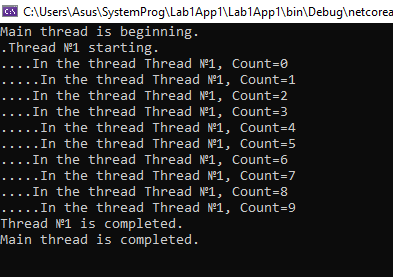


public int Count – лічильник, який починається з 0 і збільшується до 10 у методі Run. string thrdName зберігає назву потоку для ідентифікації під час виведення повідомлень. Public MyThread – конструктор, який ініціалізує лічильник до 0 і призначає ім’я потоку за допомогою переданого параметра імені. Public void Run() – метод, який виконується під час запуску потоку, друкує повідомлення про початок ланцюга, та входить у цикл, де спить протягом 500 мілісекунд (thread.sleep(500)), друкує поточний підрахунок, збільшую рахунок на 1, та коли count досягає 10, виходить із циклу та відправляє, що потік завершено.

MyThread mt = new MyThread(“Thread №1”) – створює екземпляр класу MyThread і називає його. Thread newThrd = new Thread(mt.Run) створює новий потік, і йому вказується запустити метод Run з об’єкта mt. newThrd.Start() запускає потік.







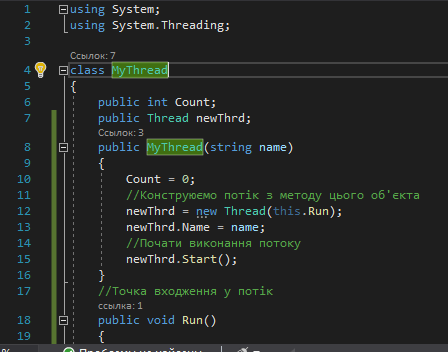
Зміни в коді:

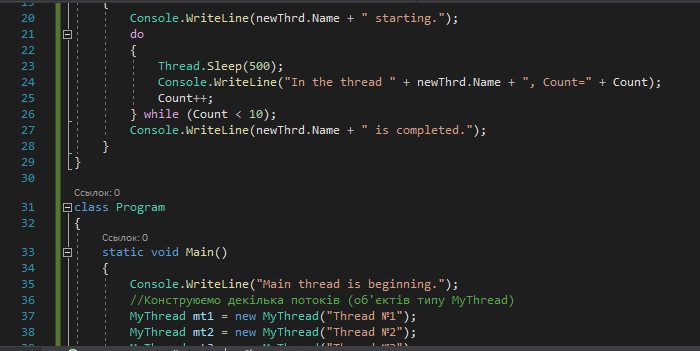
У попередній версії потік створювався та запускався поза класом MyThread, у методі Main, тоді як у новій версії створення та запуску потоку відбувається всередині конструктора MyThread (newThrd = newThread(this.Run), newThrd.Start()). Щойно створюється екземпляр MyThread, потік автоматично запускається, його не потрібно обробляти у методі Main.

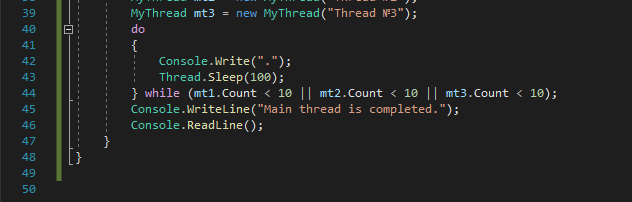
Додано Thread newThrd як члена класу MyThread – це дозволяє методу Run отримати доступ до таких властивостей, як newThrd.Name, що відобразити назву потоку.

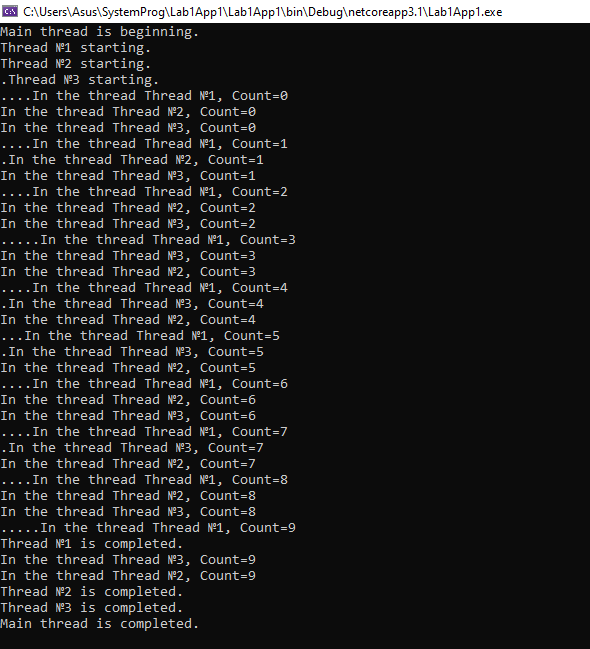
Замість збереження назви потоку в окремій змінній thrdName використовується вбудована властивість Thread.Name.

Крім того, тепер головний потік входить у цикл, друкуючи точки кожні 100 мілісекунд, чекаючи, поки фоновий потік збільшить лічильник до 10.

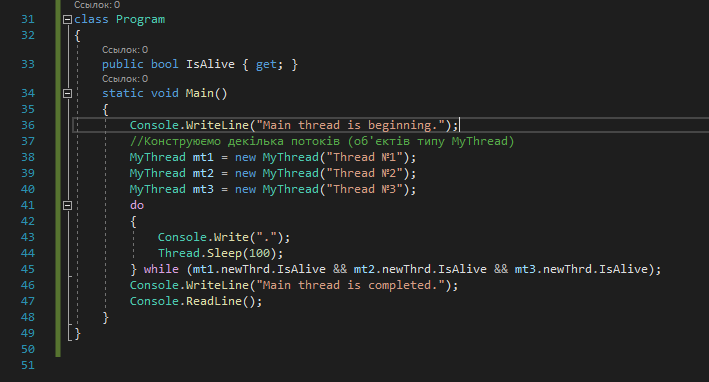


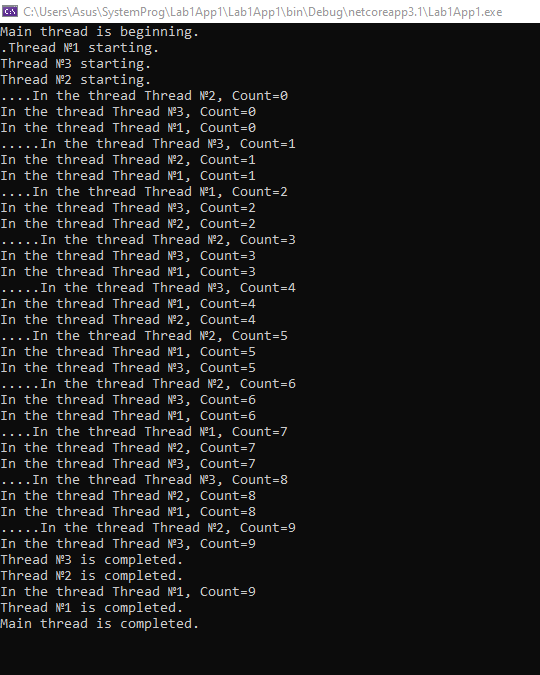




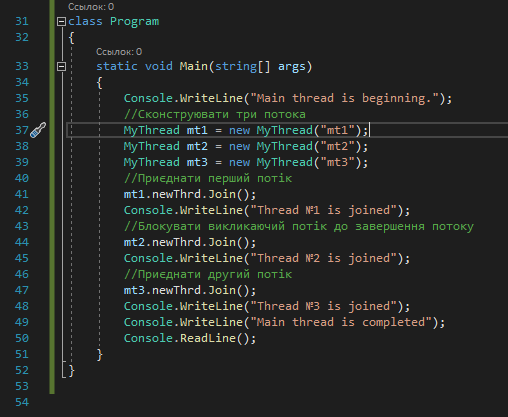


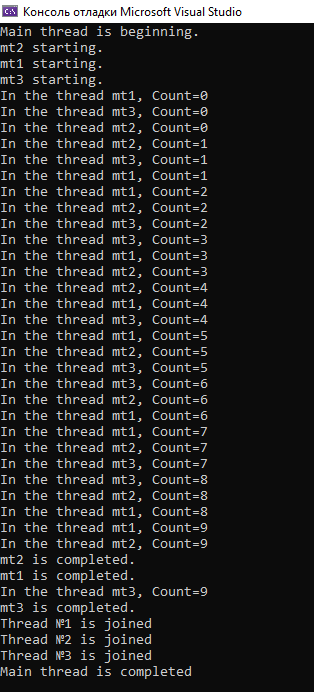
Цей код створює три окремі потоки (використовуючи екземпляри MyThread), кожен з яких працює незалежно та збільшує лічильник. Головний потік чекає, поки всі три потоки завершать свої завдання, перш ніж завершити.



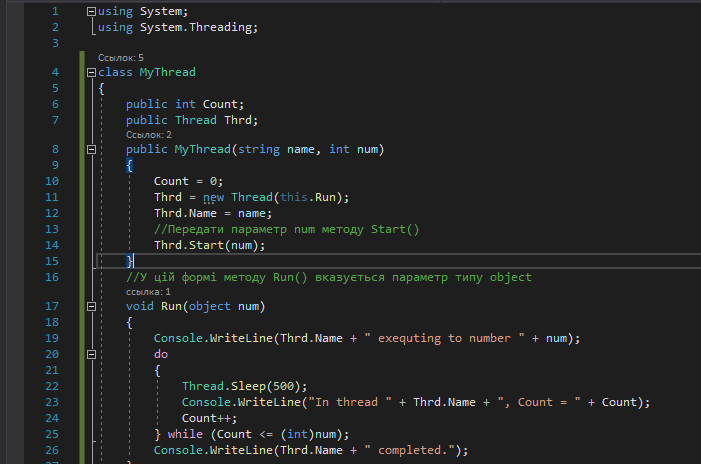


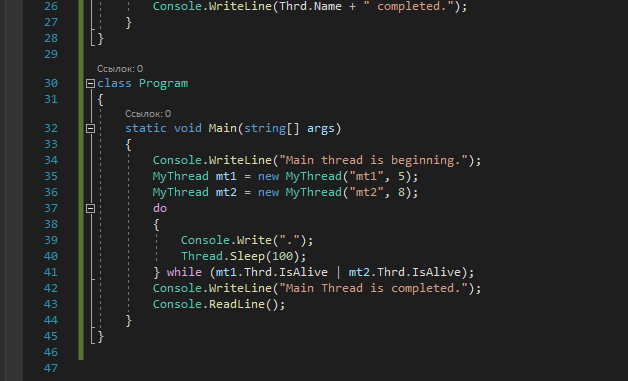
Використання властивості IsAlive – вона повертає логічне значення true, якщо потік, для якого вона викликається, продовжує виконується.

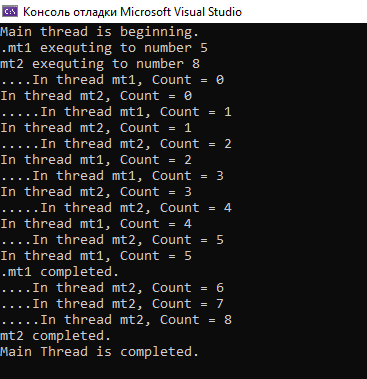




Інший спосіб відстеження моменту закінчення – викликається метод Join(), який очікує до тих пір, поки потік, для якого він був викликаний, не завершиться.





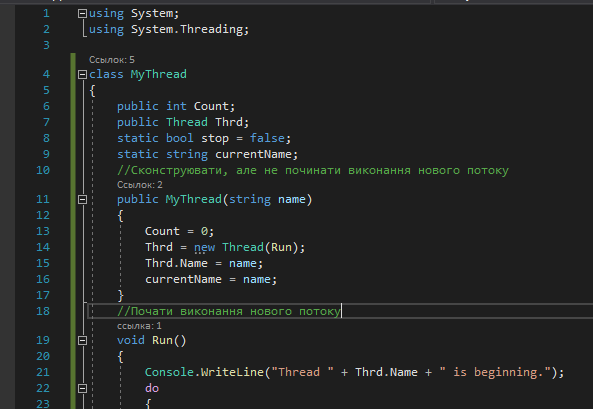


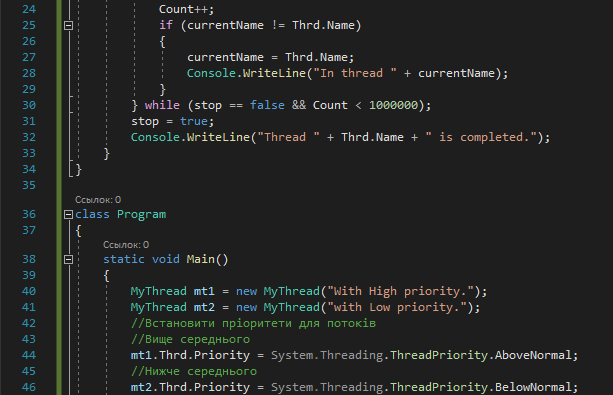
Клас MyThread тепер має параметризованими потоками. Конструктор приймає два аргументи: name, ім’я потоку, та num, параметр, який контролює тривалість виконання потоку. Потік створюється за допомогою new Thread(this.Run) і запускається з аргументом Thrd.Start(num).

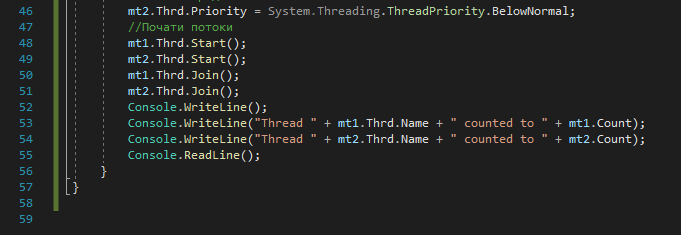
Метод Run тепер приймає параметр об’єкта під назвою num. Усередині циклу num перетворюється на ціле число за допомогою int(num). Це дозволяє потоку рахувати до динамічного обмеження на основі параметра, переданого під час запуску потоку.

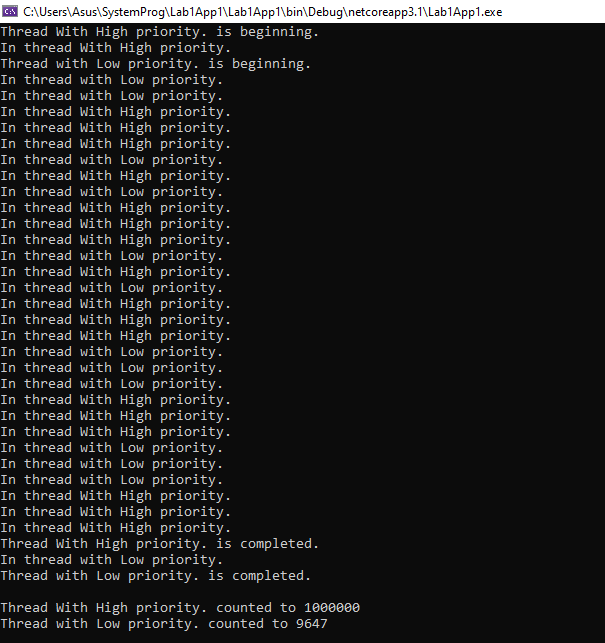
Клас Program запускає кілька параметризованих потоків, mt1 (рахує до 5) та mt2 (рахує до 8).

Основний потік тепер використовує Thrd.IsAlive, щоб перевірити, чи потоки все ще працюють. IsAlive повертає true, якщо потік все ще виконується, і false, коли він завершився – цикл продовжується до тих пір, поки живий хоча б один з потоків.







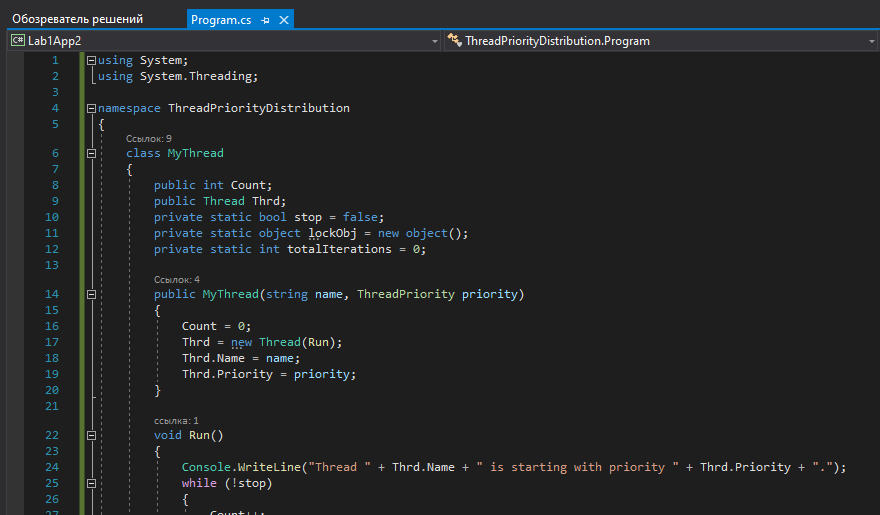


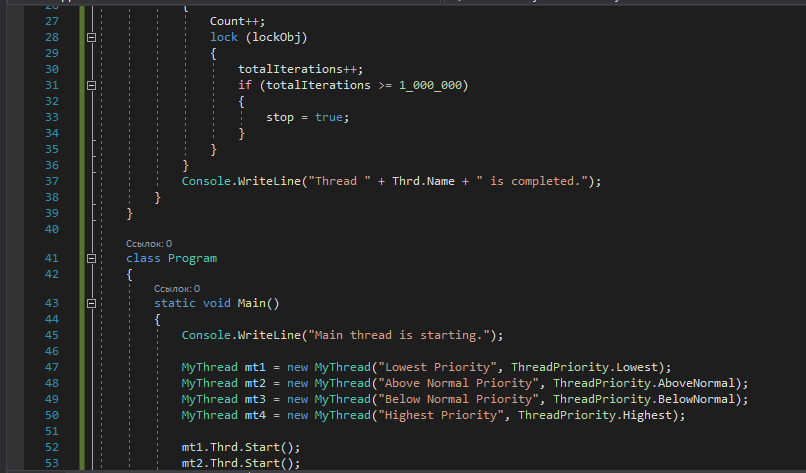
Static bool stop зупиняє всі потоки, коли один завершує своє завдання. currentName відстежує назву потоку, який зараз виконується, гарантуючи, що консоль реєструє, який потік запущено.

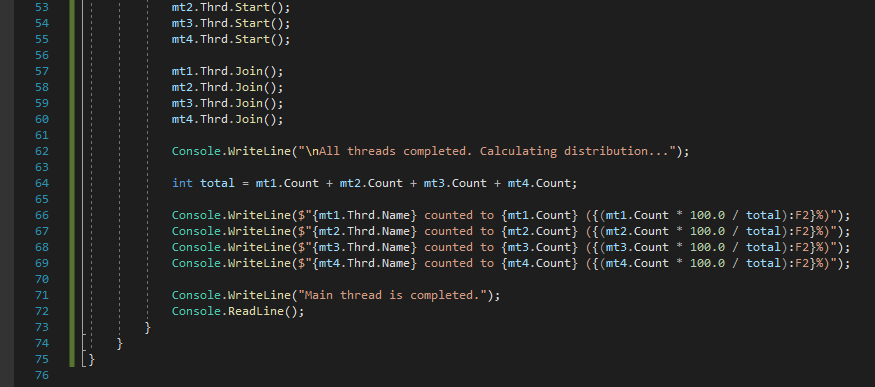
Конструктор ініціалізує потік, але не запускає негайно. Кожному потоку надається унікальне ім’я, а загальне поточне ім’я ініціалізується.

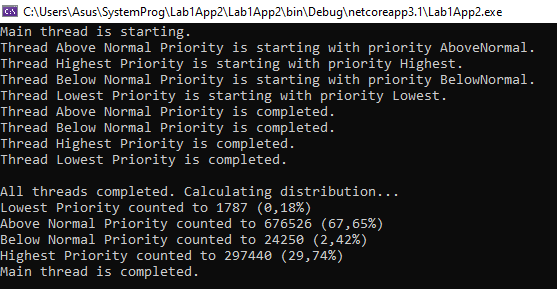
Кожен потік рахує значення від 0 до, як вказано, 1000000. Якщо інший потік починає виконуватися, це реєструється як “In thread [ThreadName]”. Коли потік досягає обмеження або триває достатньо довго, встановлюється stop = true, що зупиняє потоки. У коді створюється лише два потоки: mt1 та mt2. mt1 має пріоритет AboveNormal, тобто операційна система віддаватиме перевагу цьому потоку під час розподілу часу ЦП, тоді як mt2 має пріоритет BelowNormal, тобто він отримуватиме менше процесорного часу порівняно з mt1.

1. Створити багатопотокову програму, задати різні пріоритети для потоків. Підрахувати розподіл процесорного часу між потоками у %. Потоків – 4 шт. (пріоритети: найнижчий, вище середнього, нижче середнього, найвищий).









Клас MyThread визначає спеціальний клас потоку для інкапсуляції поведінки потоку та даних (наприклад, підрахунок). Public int Count відстежує кількість ітерацій, виконаних кожним потоком, public Thread Thrd запускає метод Run, private static bool stop = false є спільним прапором, який використовується для сигналу про зупинку всіх потоків після виконання умови, private static object lockObj = new object () використовується для синхронізації доступу до спільної змінної totalIterations, яка є спільним лічильником для відстеження загальної роботи, виконаної в усіх потоках.

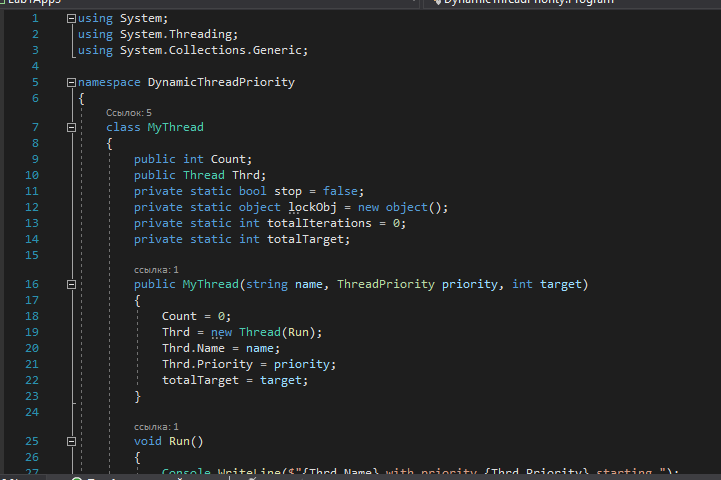
Конструктор ініціалізує Count до 0 і налаштовує Thread за допомогою його методу Run, та встановлює назву та пріоритет потоку, які передаються як аргументи.

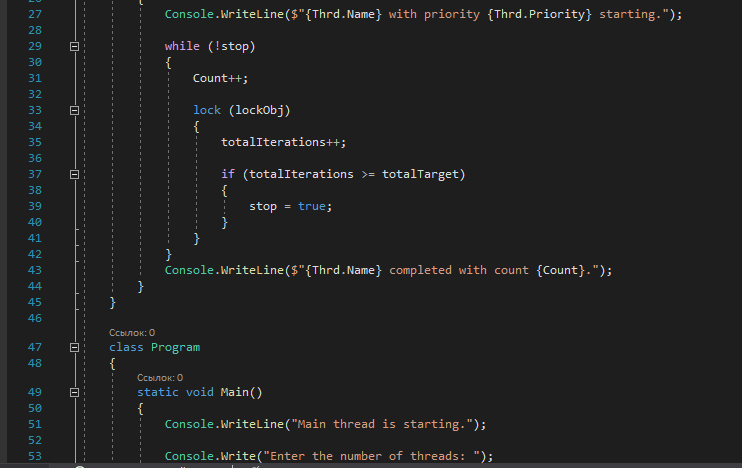
Метод Run виводить початкове повідомлення з назвою та пріоритетом потоку. Цикл збільшує Count, доки спільна totalIterations не досягне 1 000 000, тоді як lock (lockObj) гарантує, що лише один потік оновлює totalIterations за раз. Як тільки умова виконується, stop встановлюється на true, що змушує всі потоки виходити з циклу.

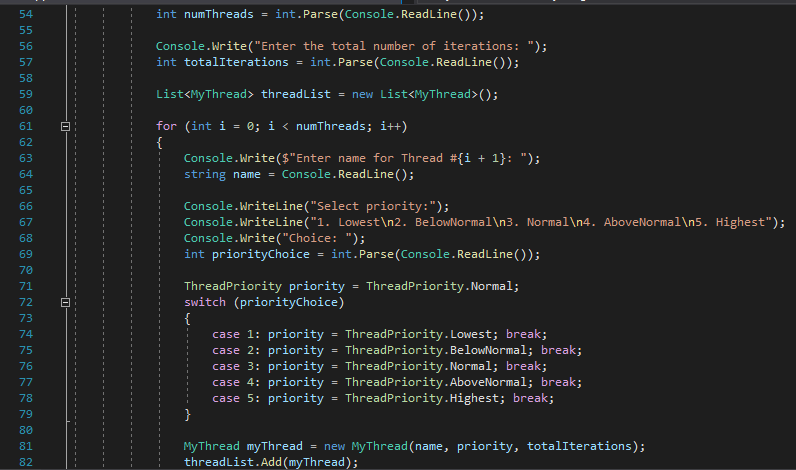
Клас Program створює чотири об’єкти MyThread з різними пріоритетами, запускає всі потоки за допомогою Start() і чекає на їх завершення за допомогою Join(). Після завершення всіх потоків загальна кількість підсумовується для обчислення внеску кожного потоку – результати відображаються як у абсолютних значеннях, так і у відсотках від загальної виконаної роботи.

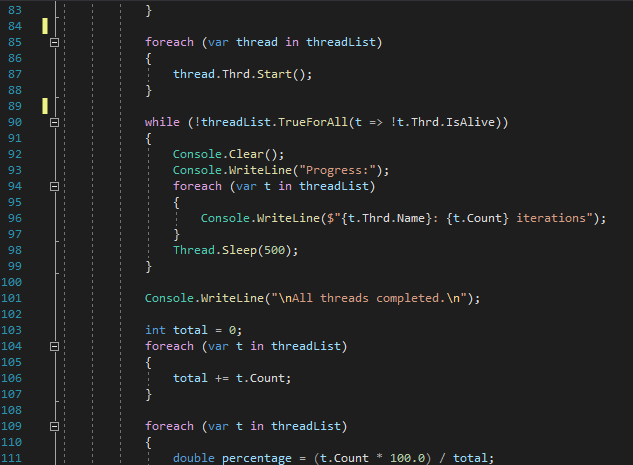
Програма друкує кількість ітерацій, виконаних кожним потоком, і відсоток загальної роботи, показуючи, як пріоритет потоку вплинув на розподіл часу ЦП.

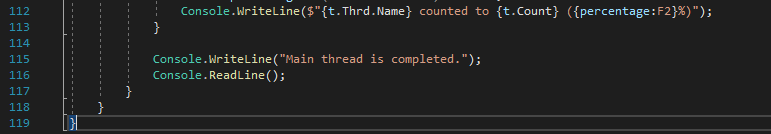
1. Високий рівень: Реалізувати програму із попереднього завдання з використанням графічного інтерфейсу та можливістю динамічного задання кількості та пріоритетності потоків; відображення ходу обчислень та виведення результатів

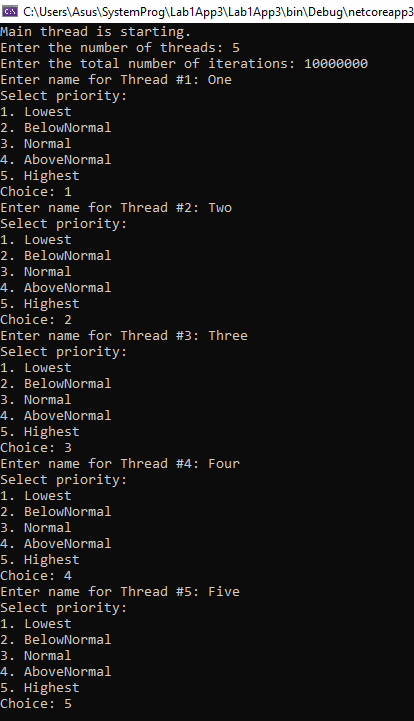


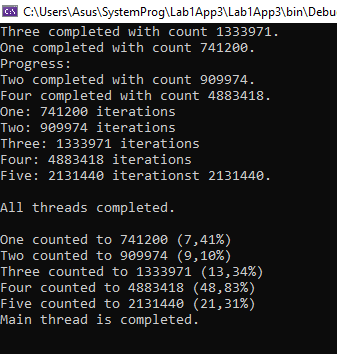




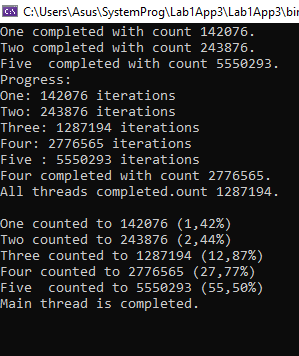


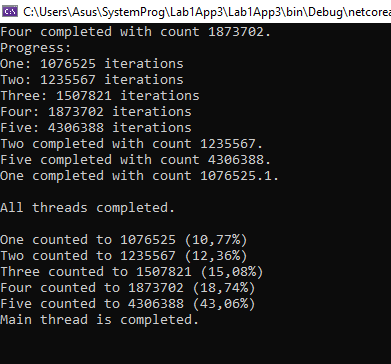






Такі ж самі налаштування:





List<MyThread> threadList = new List<MyThread>() створює список (динамічний масив) для зберігання екземплярів класу MyThread – використовується, щоб динамічно відстежувати всі потоки. Через for (int i=0; i<numThread;i++) динамічно створюємо потоки, використовуючи надані імена та пріоритети, також додаючи їх до списку через threadList.Add.

threadList.TrueForAll(t=>!t.Thrd.IsAlive) перевіряє, чи всі потоки завершено – вираз в скобці говорить про те, що потік не живий.

total підсумовує загальну кількість ітерацій, виконаних усіма об’єднаними потоками. t.Count містить кількість ітерацій, виконаних певним потоком.

Double percentage рахує % загальної роботи, виконаної кожним потоком.

1. Оформити звіт