Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра мережевих та інтернет технологій

Лабораторна робота №2

З дисципліни “Системне програмування”

Студента Фуркала Олександра

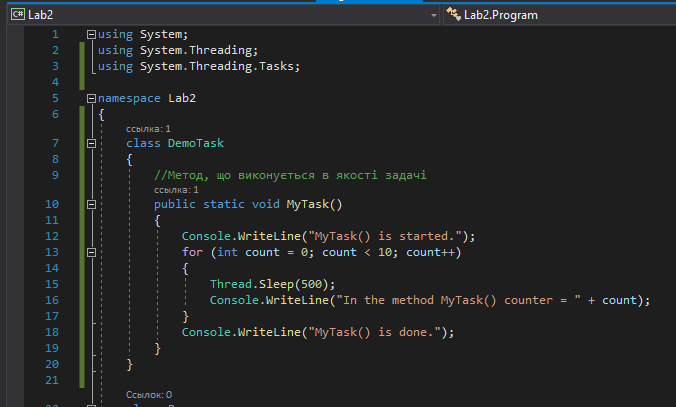
За темою:

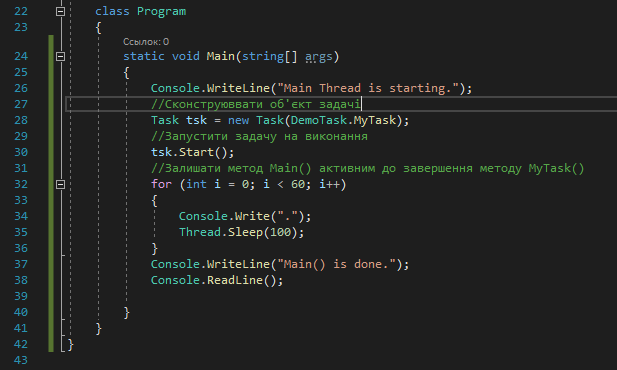
“Паралелізм даних та паралелізм задач (Ч.1)”

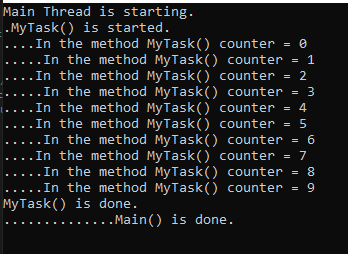
Варіант-3

Хід роботи

1. Повторити задачі

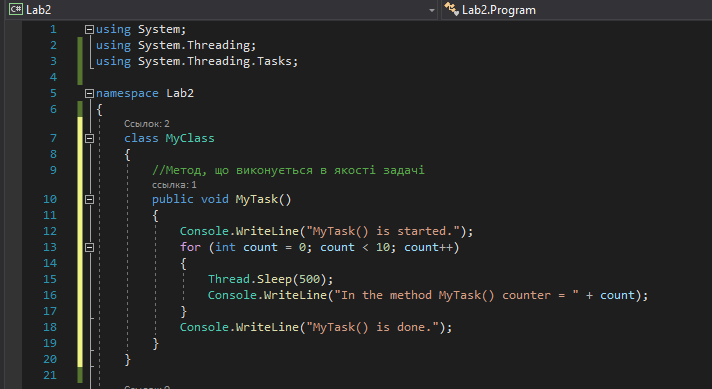


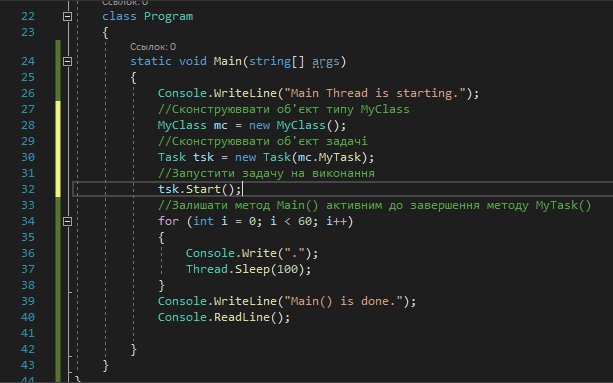


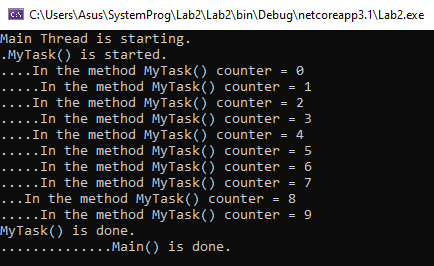


Клас створює об’єкт Task.tsl, який запускає DemoTask.MyTask(), але не запускає його негайно. Завдання запускається за допомогою tsk.Start(), MyTask() виконується асинхронно. Також виконується цикл (60 разів), який друкує . кожні 100 мс (імітація роботи основного потоку).

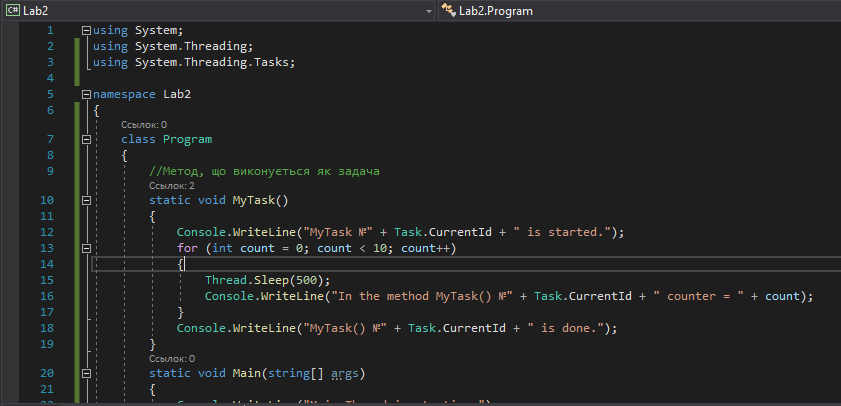
Основний потік виконується незалежно, друкуючи крапки – завдання tsk виконується паралельно, виконуючи MyTask(), та виклики Thread.Sleep() імітують затримки для демонстрації того що потоки виконуються одночасно.

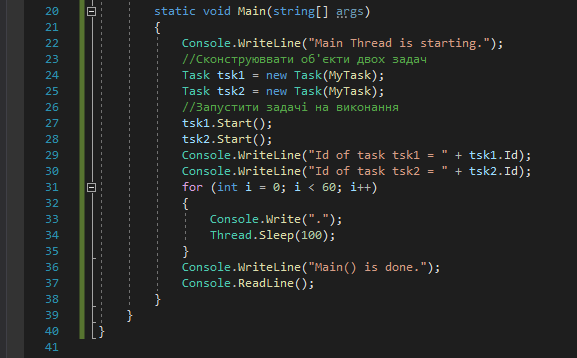


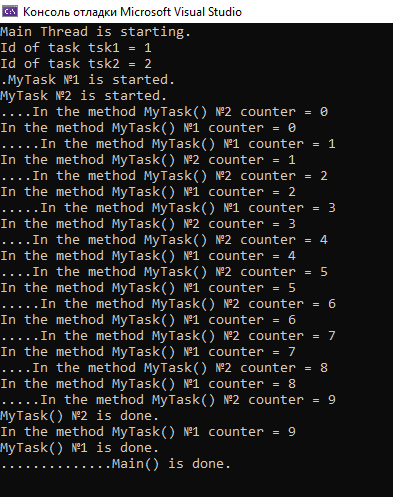




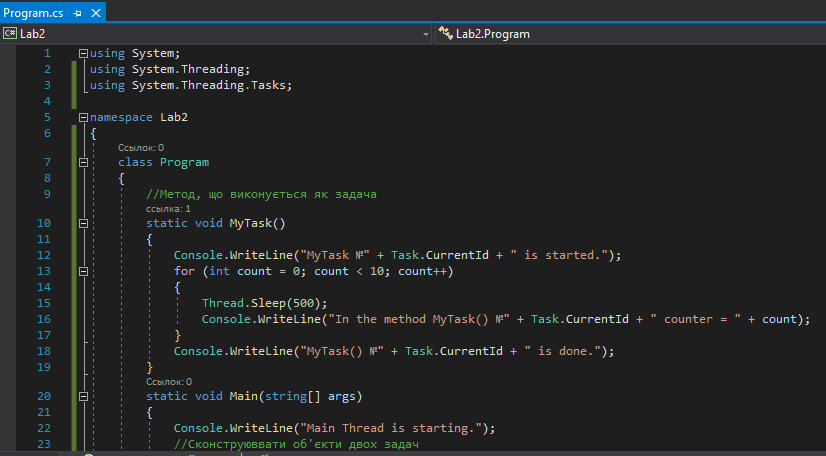
Приклад програми, коли задача інкапсулюється всередині класу.

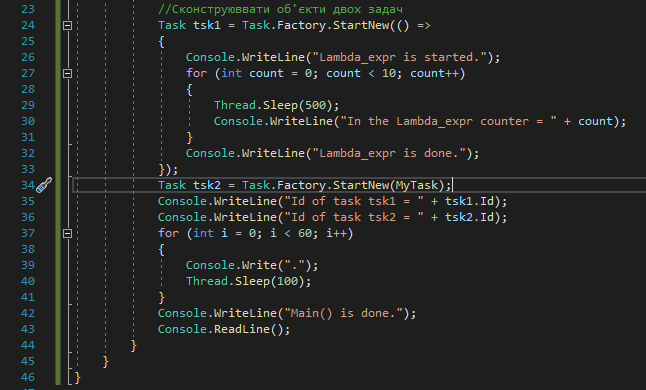


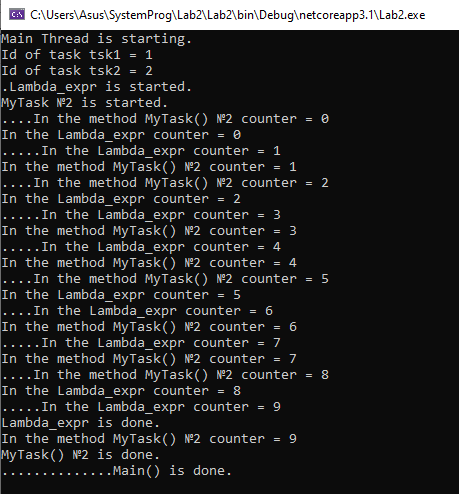




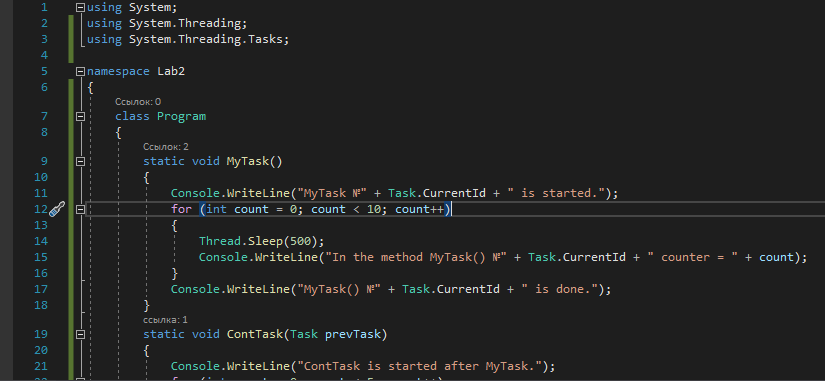
Task.CurrentId допомагає визначити, яке завдання виконується в будь-який момент часу. Кожне завдання отримує унікальний ідентифікатор, який автоматично призначається середовищем виконання. tsk1 і tsk2 створюються окремо за допомогою new Task(MyTask), обидва завдання запускаються за допомогою tsk1.Start() і tsk2.Start().

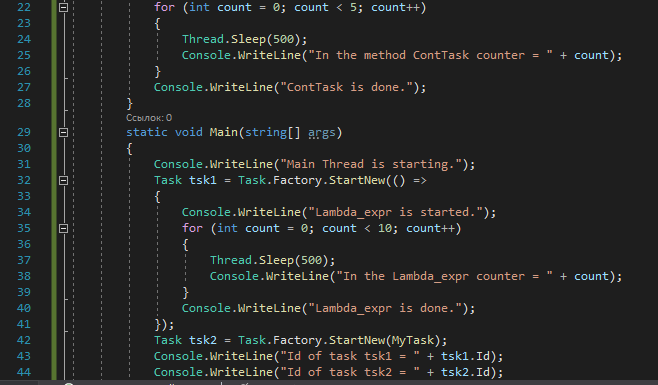


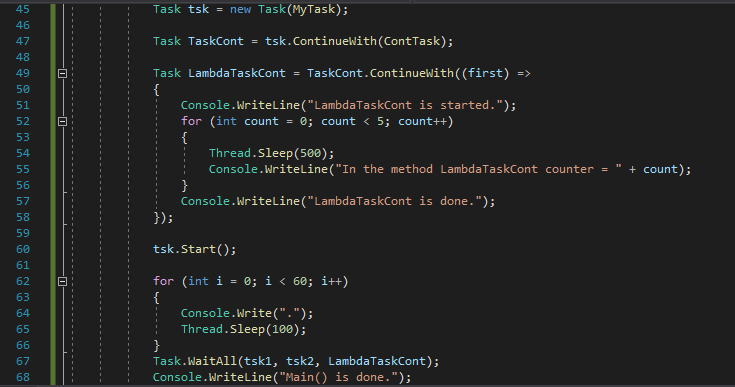


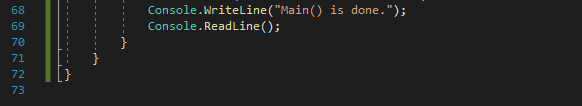


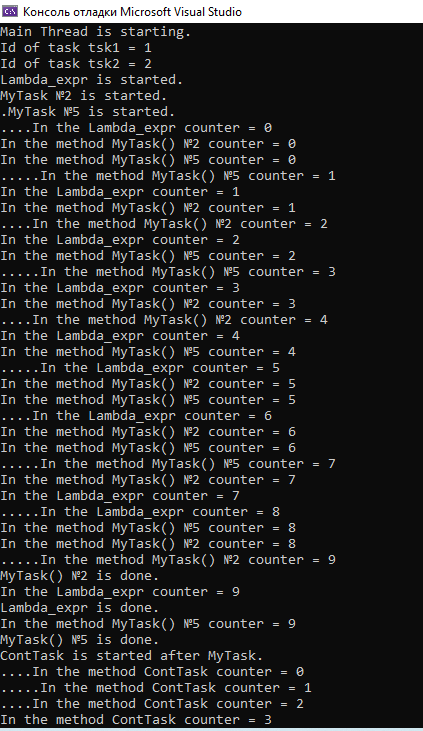
Використовується Task.Factory.StartNew() для створення та запуску задачі. Також використовується лямбда-вираз (форма анонімних функцій) як окремо вирішувана задача.

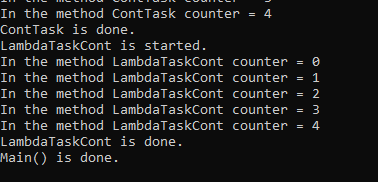






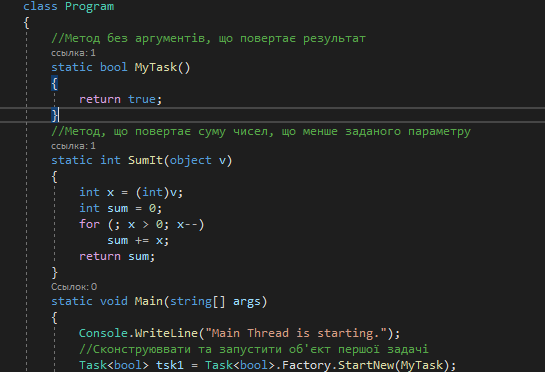


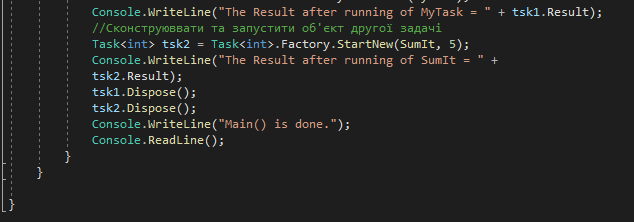


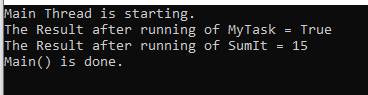


Tsk1 та tsk2 запускаються негайно, tsk(MyTask) також створюється, але запускається пізніше. Коли tsk завершує роботу, запускається TaskCount(ContTask), після кого запускається LambdaTaskCount, в той час як основний потік відправляє крапки.

Продовження – це задачі, які автоматично починаються після завершення другої задачі. Їх можна створити через метод ContinueWith().

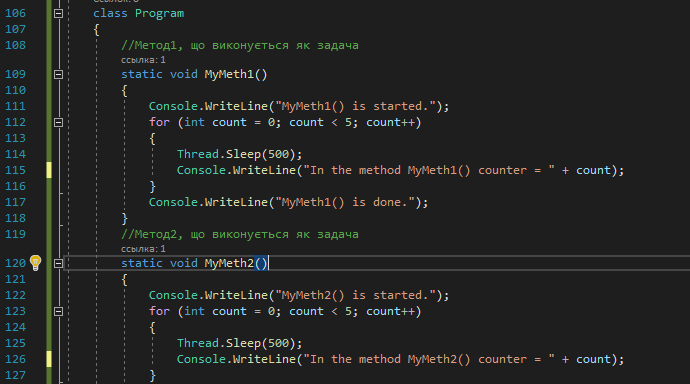


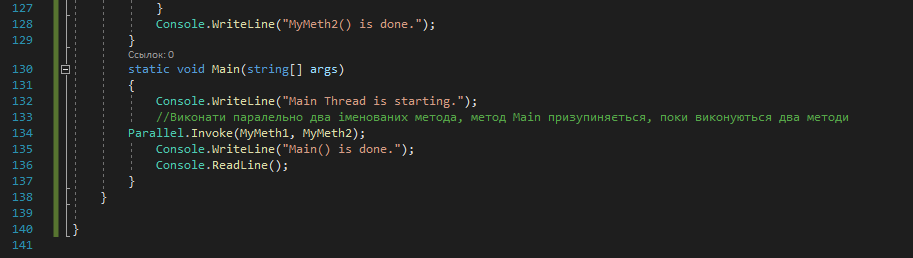


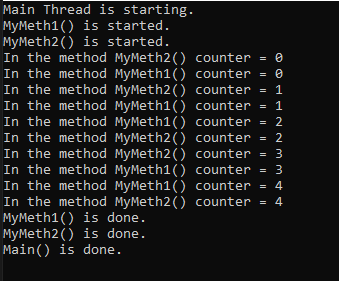


Метод MyTask() повертає true, tsk1 створюється за допомогою Task<bool>.Factory.StartNew(MyTask), та доступ до результату здійснюється за допомогою tsk1.Result.

Метод SumIt(object v) обчислює суму чисел від v до 1, tsk2 створюється за допомогою Task<int>.Factory.StartNew(SumIt, 5).  
Наприкінці звільнюємо ресурси, tsk.Dispose().

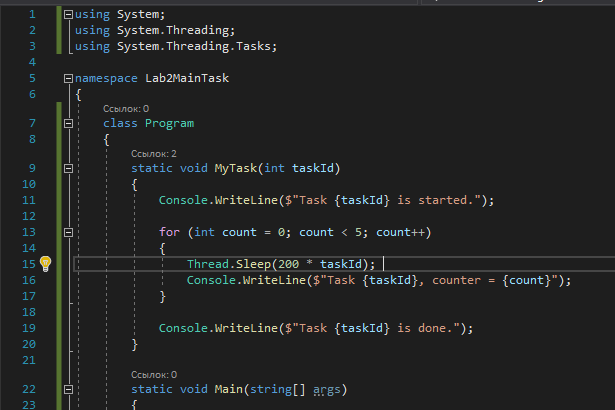


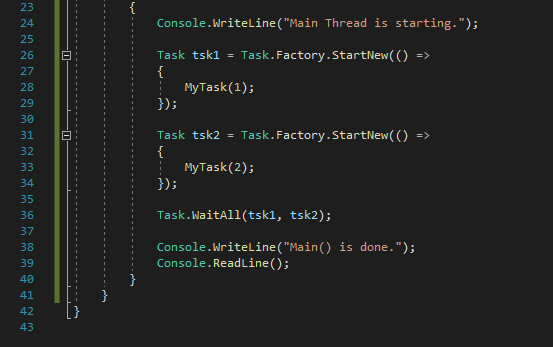


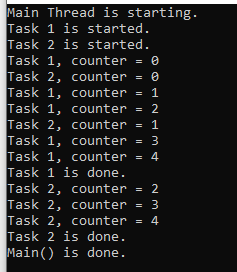


Invoke дозволяє виконувати один або кілька методів, що вказуються у вигляді його аргументів. Кожен метод, який передається цьому методу як аргумент, не повинен ні приймати параметрів, ні повертати значення (але виконувані методи можуть бути вказані у вигляді змінного списку аргументів). Метод Invoke() спочатку ініціює виконання, а потім очікує завершення всіх переданих методів.

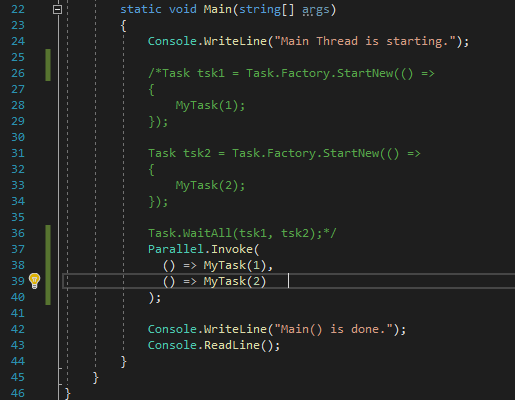
1. Створити програму, що створює дві задачі, які виконуються паралельно. Затримку методом Sleep() організувати на величину 200мс та пропорційно ідентифікатору задачі.
2. Організувати очікування виконання задач методом WaitAll()
3. Визначити задачу для виконання у вигляді лямбда-виразу

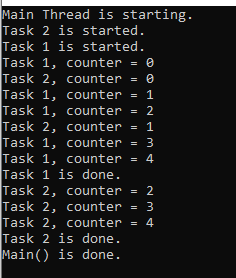






1. Створити програму паралельних обчислень за допомогою виклику методу Invoke(), де в якості аргументів застосовуються лямбда-вирази





1. Оформити звіт