Львівський національний університет імені Івана Франка

Факультет прикладної математики та інформатики

**Паралельні та розподілені обчислення**

**Лабораторна робота №1**

Виконав:

Студент групи ПМі-33

Тимчишин Ярема

Львів 2023

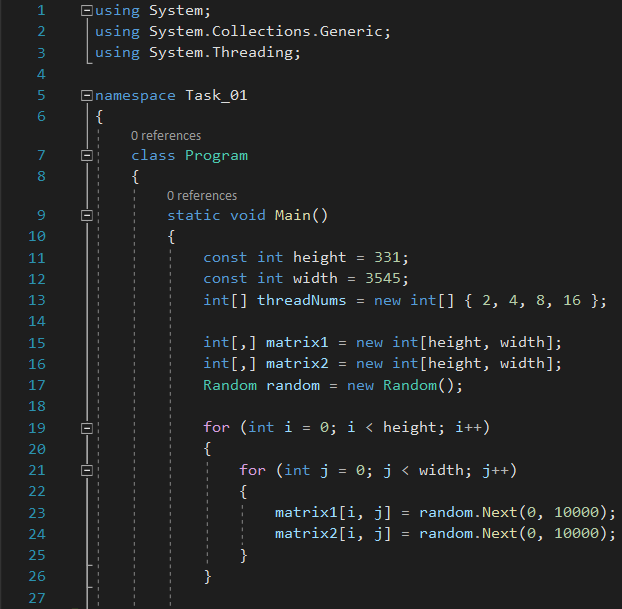
**Тема:** Додавання(віднімання) матриць.

**Завдання:** Написати програму обчислення суми(різниці) двох матриць (послідовний та паралельний алгоритми). Порахувати час роботи кожної з програм, обчислити прискорення та ефективність роботи паралельного алгоритму.

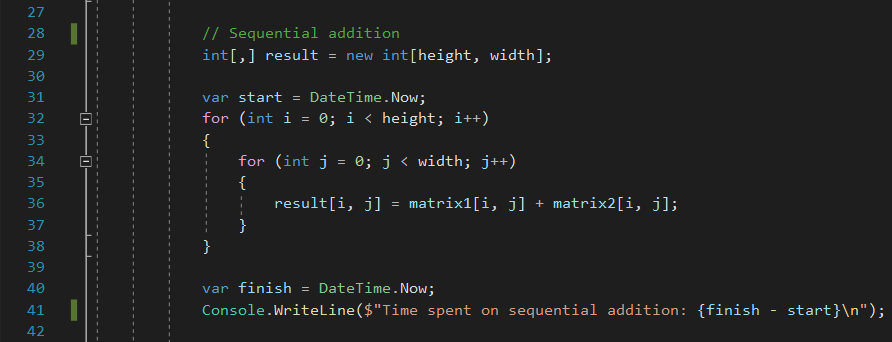
**Хід роботи**

Використовуючи мову програмування C# написав програму для обчислення суми двох матриць та порахував час роботи кожного із методів.

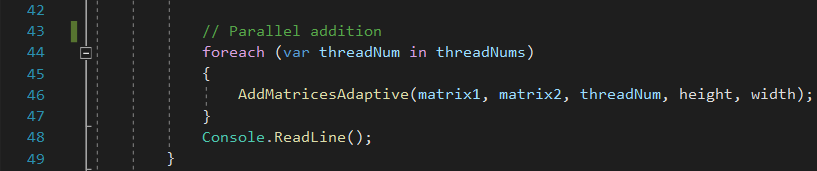
Почав із оголошення констант для висоти та ширини матриць, а також масиву “threadNums”, який зберігає кількість потоків для паралельного методу. Два двовимірні масиви “matrix1” та “matrix2” для зберігання випадкових цілих значень. А також створив об’єкт “Random” для генерації випадкових значень матриці. Після чого написав два вкладених цикли “for”, які заповнюють матриці “matrix1” та “matrix2” випадковими цілими числами від 0 до 10000.



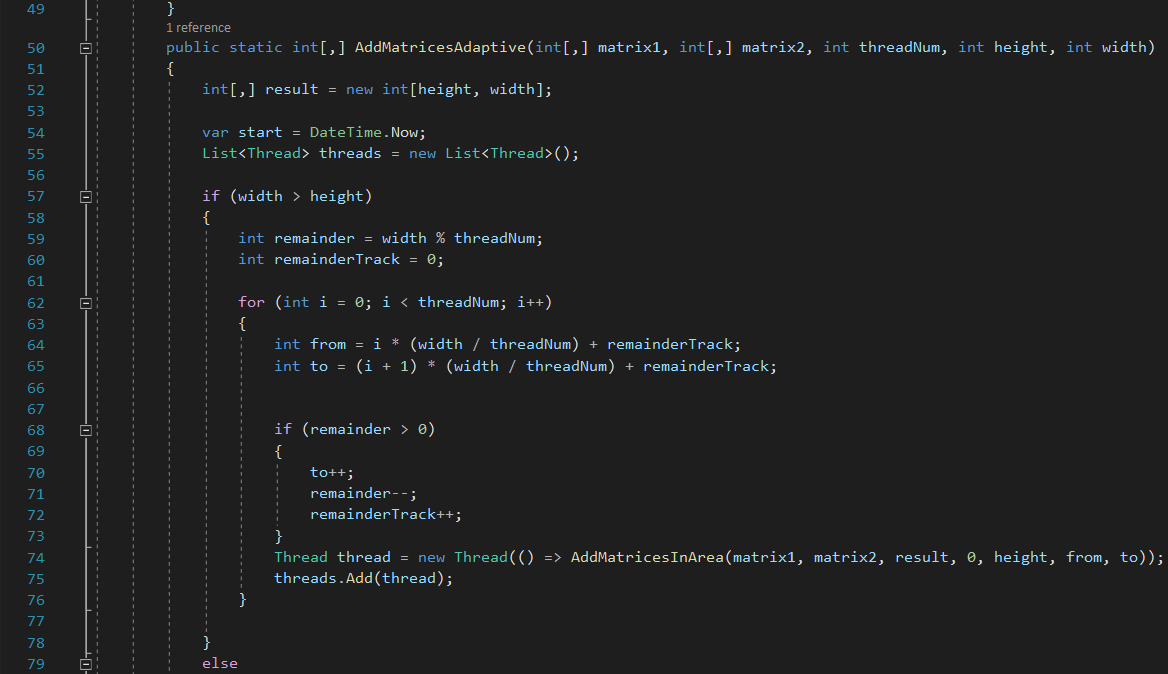
Для послідовного методу, почав із оголошення нового двовимірного масиву “result” в якому буде зберігатися результат додавання матриць. Після чого за допомогою “DateTime.Now” записав в змінну “start” початок додавання матриць. Потім за допомогою ще двох вкладених циклів “for” відбувається послідовне додавання елементів двох матриць. Знову за допомогою “DateTime.Now” присвоїв в змінну “finish” кінець додавання двох матриць. Після чого, обчислив час витрачений на паралельне додавання матриць та вивів його на консоль.

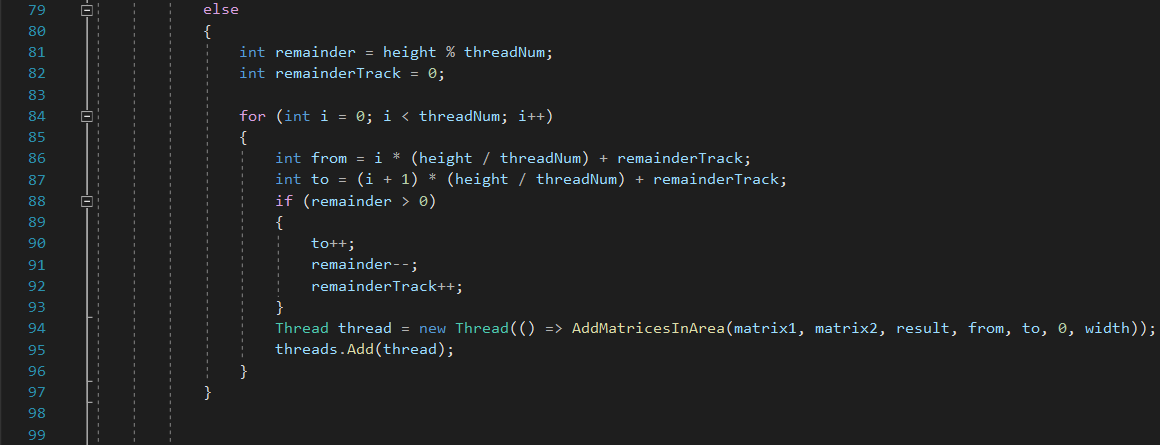


Для паралельного методу, почав із циклу “foreach” який перебирає значення масиву “threadNums” та викликає функцію “AddMatricesAdaptive”, яка кожного разу бере різну кількість потоків і про яку я розкажу далі.

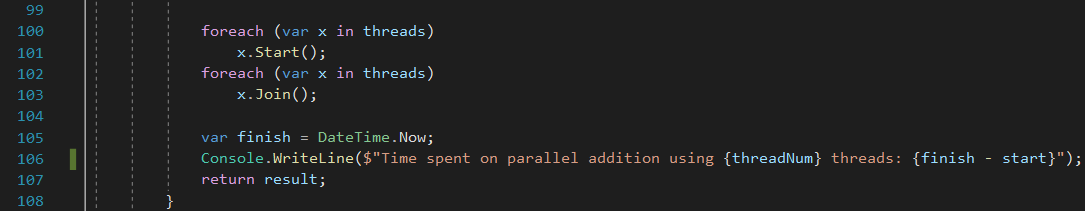


Функція “AddMatricesAdaptive” відповідає за виконання паралельного додавання матриць. Двовимірний масив “result” для зберігання додавання двох матриць та “DateTime.Now” для засікання часу роботи алгоритму, усе аналогічно до того як це було в послідовному додаванні. Список “<Thread>” під назвою “threads” для зберігання об’єктів потоку для паралельної обробки. Після чого, використовую “if” для того щоб перевірити чи є ширина або висота більшою, щоб рівномірно розподілити роботу між потоками. Якщо ширина є більшою за висоту, програма ділить ширину на сегменти, гарантуючи, що кожна нитка(“thread”) отримала справедливу частку роботи. Якщо ж висота є більшою, то програма дотримується подібної логіки, але натомість ділить висоту на сегменти. Самі потоки створюються за допомогою об’єктів “Thread” і додаються до списку потоків.





Після створення усіх потоків, програма запускає їх за допомогою методу “Start()”, потім чекає щоб всі потоки завершили свою роботу та з’єднює їх результати за допомогою методу “Join()”. Програма записує час закінчення, тоді коли всі потоки вже завершили свою роботу. Після чого програма обчислює час, витрачений на паралельне додавання, та виводить його на консоль разом з кількістю використаних потоків.



Додаткова функція “AddMatricesInArea”. Ця функція відповідає за додавання матриць у певній області, визначеній “fromRow”, “toRow”, “fromColumn” і “toColumn”. Вона використовується паралельними потоками для оновлення зміненої матриці результатами додавання в межах призначеної їм області.

