

Лабораторна робота №3

Знайомство з потоками в C# (System.Threading).

Увага!

- Першим ділом розробіть код програми.
- Спочатку перевіряйте роботу на масивах невеликих довжин.
- Посилання на документацію є в цьому файлі знизу, але можете гуглити самостійно

Завдання до лабораторної роботи

Крок 1. Підготовка до роботи

1. Перевірте, чи ваша машина готова до виконання роботи. Скористайтесь файлом "Підготовка робочого місця"
2. Ознайомтеся з **прикладом**, який потрібен для початку роботи.
<https://github.com/d-borozenets/KS-2025/tree/main/Lab2Threads>
3. Можна або скопіювати весь код з [Program.cs](#) або лише фрагмент з обрахунками
4. Визначте кількість ядер вашого процесора N будь-яким способом, але краще це зробити програмно (Environment Class)

Крок 2. Дослідження використання потоків.

1. Визначте, скільки часу виконуватимуться **обрахунки** (код функції `HeavyCalculations()`) з прикладу
 - Для цього скористайтесь `Stopwatch` або `DateTime`
 - підберіть такі значення `const int count`, щоб обрахунки тривали:
 - i. 0.5-1 сек для тестування коду (далі в інструкції **count_для_коду**)
 - ii. 10-20 сек для звіту (далі в інструкції **count_для_звіту**)
 - занотуйте собі ці значення (наприклад в коментарях).
2. Визначте, скільки часу виконуватиметься програма, що виконуватиме **послідовно** ці ж **обрахунки** 2 рази, 3 рази і так далі до N+1 разів
3. Для пункту 2 подивіться, у якій послідовності видавалися результати виконання **обрахунків** в консоль. (для цього передавайте в `HeavyCalculations()` унікальний `label` для кожного виклику)
4. Запишіть результати в таблицю, якщо зробили дослідження з **count_для_звіту**.
 - рядки - кількість викликів обрахунків (від 1 до N+1)
 - час - в мілісекундах для обраної кількості викликів
 - порядок - порядок у якому завершилися **обрахунки** (наприклад: 00,01,02,03 - для 4 запусків)
5. Визначте, скільки часу виконуватиметься програма, що виконуватиме **багатопоточно** ці ж **обрахунки** 2 рази, 3 рази і так далі до N+1 разів.

- Потоки при цьому створюйте з використанням лямбда-виразів(*Lambda expression*).
 - Тобто брати сам код функції *HeavyCalculations()* з **прикладу**
 - Не забудьте заблокувати виконання основного потоку до завершення роботи всіх потоків.
6. Запишіть час і порядок виконання у відповідні комірки таблиці(аналогічно з послідовною обробкою) Стовпець **Lambda вираз**
 7. Виконайте пункт 5 знову але:
 - при цьому сам фрагмент коду **обрахунків** оформіть у вигляді **статичної функції(Static Method)**(виконати якщо ви з **прикладу** НЕ копіювали функцію *HeavyCalculations()*)
 - при створенні потоків задавайте вашу функцію або *HeavyCalculations()* .
 - порядковий номер буде передаватися через метод потоку *Start(label)*
 - ваша функція має приймати його
 8. Запишіть час і порядок виконання у відповідні комірки таблиці(аналогічно з пунктом 6). Стовпець **Static Method**
 9. Виконайте пункт 7 знову але:
 - при цьому для створення потоків використовуйте пул.
 - не забудьте використовувати документацію
 - <https://learn.microsoft.com/en-us/dotnet/api/system.threading.threadpool.queueuserworkitem?view=net-9.0>
 10. Запишіть час і порядок виконання у відповідні комірки таблиці(аналогічно з пунктом 6). Стовпець **ThreadPool**
 11. Запишіть результати у стовпець (8) таблиці.
 12. Дослідіть як впливає пріоритет потоків. Виконайте пункт 7 знову але:
 - Надайте кожному потоку різний пріоритет
 - порівняйте результати:
 - i. отримані при кількості потоків менше кількості ядер процесора, при кількості потоків
 - ii. рівному кількості ядер, при кількості потоків більше кількості ядер.
 - Зробіть висновок:
 - i. у яких ситуація пріоритет вплинув порядок виведення результату на екран, у яких немає.

Крок 3. Внесення даних у звіт.

1. Внесіть дані у звіт при використанні **count_для_звіту**
2. Внесіть решту інформації та свої висновки у звіт

<https://learn.microsoft.com/en-us/dotnet/api/system.environment?view=net-9.0>

<https://learn.microsoft.com/en-us/dotnet/api/system.diagnostics.stopwatch?view=net-9.0>

<https://learn.microsoft.com/en-us/dotnet/api/system.datetime?view=net-9.0>

<https://learn.microsoft.com/en-us/dotnet/api/system.threading?view=net-9.0>