Лабораторна робота №3а
Паралельне мультипоточне програмування

Реалізація алгоритмів пошуку підрядка та підрахунку частоти слів у тексті

Ця лабораторна робота реалізує алгоритм пошуку підрядка та підрахунку частоти слів у тексті. Ці алгоритми дозволяють продемонструвати відмінності продуктивності послідовного та паралельного виконань.

Структура:

У коді використовувалась стандартна бібліотека C++ для створення та керування потоками, а всі класи та функції було реалізовано з дотриманням принципів об'єктно-орієнтованого програмування та інкапсуляції.

Файли проекту:

- *MainForm.h*
- *MainForm.cpp*

Ці файли містять код для створення головного вікна програми, налаштування інтерфейсу користувача та обробки подій. Інтерфейс дозволяє вводити текст, виконувати пошук підрядка та підрахунок частоти слів, а також відображати результати.

- *ITextProcessor.h*

Цей файл містить інтерфейс *ITextProcessor*, який визначає метод process для обробки тексту.

- *SequentialSubstringSearch.h*
- *SequentialSubstringSearch.cpp*

Містять реалізацію послідовного алгоритму пошуку підрядка у тексті.

- *ParallelSubstringSearch.h*
- *ParallelSubstringSearch.cpp*

Містять реалізацію паралельного алгоритму пошуку підрядка у тексті.

- *SequentialWordFrequencyCounter.h*
- *SequentialWordFrequencyCounter.cpp*

Містять реалізацію послідовного алгоритму підрахунку частоти слів у тексті.

- *ParallelWordFrequencyCounter.h*
- *ParallelWordFrequencyCounter.cpp*

Містять реалізацію паралельного алгоритму підрахунку частоти слів у тексті.

- *ProcessorFactory.h*

Цей файл містить фабрику *ProcessorFactory,* яка створює об'єкти для обробки тексту, визначаючи, чи буде використано послідовний чи паралельний алгоритм.

Інтерфейс користувача:

Для зручності роботи з програмою було створено головне вікно з текстовими полями для введення тексту та підрядка, а також кнопками для виконання пошуку підрядка та підрахунку частоти слів. Результати виводяться у окреме текстове поле.

Результати виконання алгоритмів:

Для дослідження продуктивності було виконано послідовний та паралельний підрахунок частоти слів у великому тексті та пошук підрядка.

Ось отримані результати:

Послідовний підрахунок та пошук:

Пошук підрядка : Виконується досить швидко, особливо на великих текстах, які мають мільйони символів.

Підрахунок частоти слів : Ефективний на великих обсягах тексту, займає значно менше часу порівняно з паралельною версією на малих текстах.

Паралельний підрахунок та пошук:

Пошук підрядка : Час виконання може бути трохи довшим через витрати на створення та керування потоками. Вигоди від паралельного виконання може бути помітною на великих обсягах даних.

Підрахунок частоти слів : Може займати більше часу на менших обсягах тексту через витрати на координацію потоків. На великих обсягах даних може принести значне прискорення.

Висновок:

Виконано дослідження ефективності послідовного та паралельного виконання алгоритмів пошуку підрядка та підрахунку частоти слів у тексті. Послідовні версії демонструють добру продуктивність на малих обсягах даних, тоді як паралельні версії можуть бути вигідні на великих обсягах, хоча не завжди. Об'єктно-орієнтований дизайн

був строго дотриманий, з використанням інтерфейсів класів та інкапсуляції функціональності.