Лабораторна робота № 2. "Організація багатопоточного виконання програм .NET".

Метою роботи є вивчення технології багатопоточного програмування засобами .NET Framework та мови C# на прикладі реалізації задачі інформаційного пошуку.

Робота виконується бригадою з 2-х студентів. Варіант визначається старостою академічної групи. Протокол до лабораторної роботи має містити титульний аркуш, загальне завдання та завдання за варіантом, фрагмент коду програму (2-4 сторінки) та 2-3 копії екранних форм програми.

1. **Завдання.** Використовуючи середовище розробки Microsoft Visual Studio .NET і мову програмування C#, реалізувати функцію пошуку фрази серед набору документів, представлених текстовими файлами. Під фразою розуміти набір **двох або більше** слів, розділених пробілом. При цьому врахувати булевську функцію за варіантом. Наприклад, якщо запитом є «СЛОВО1 СЛОВО2», а у варіанті визначено «**a OR b OR c**», то результатом пошуку буде список документів, де зустрічається слово СЛОВО1 або СЛОВО2.

   Результатом пошуку вважати ознаку «*знайдено*» чи «*не знайдено*» пошукову фразу у документі, не враховуючи позицію слів фрази у документі.

   Пошук виконати двома способами:
   - за допомогою колекції .NET за варіантом;
   - за допомогою булевської моделі пошуку та колекції BitArray (див.п.3).

   Задачі потоків:
   а) *Головний потік* – обробка подій від користувача, виведення результатів пошуку;
   б) *Потік 1* – підготовка колекцій до пошуку: створення індексів документів
   в) *Потік 2* – пошук у колекції за варіантом
   г) *Потік 3* – пошук у колекції BitArray

   Потоки мають виконуватися паралельно, при цьому швидкість пошуку має підраховуватися для подальшого порівняння.

Перед створенням програми підготувати вхідний набір документів. Сформувати 20-30 текстових документів українською мовою, в яких буде здійснюватися пошук. Розмір документу > 50Кб.

2. **Основні функції**:

- Розробити спосіб зберігання індексів документів за допомогою колекції за варіантом.
- Пошук у колекції забезпечити за допомогою методів IndexOf або Contains.
- Результати пошуку мають містити: назву документу, ознаку (знайдено чи ні), час пошуку у колекції за варіантом та завдяки використанню BitArray.
- Надати можливість «ручного» додавання документа у індекс у будь-який момент часу (зокрема, при виконанні пошуку).
- Забезпечити синхронізацію потоків у випадку «ручного» додавання документів та отриманні результатів пошуку головним потоком.
- Програма повинна мати віконний інтерфейс.

3. Булевська модель пошуку, індексація документів

Для всіх документів d з набору D сформувати спільний список слів $S=\{s_1,s_2..s_n\}$, де n – кількість слів, які входять до всіх документів (без повторів). Побудувати для кожного документу свій вектор $V_1=\{v_1,v_2,...v_n\}$, де $v_k \in \{0,1\}$, $k = 1..n$. Значення «1» компоненти v_k означає наявність слова s_k у документі d_i. «0» – його відсутність у даному документі.

Задача пошуку при такій моделі зводиться до перетворення пошукового запиту до векторного виду і виконання побітової операції AND, OR, над запитом та з кожним вектором з набору документу. Вихід – новий бітовий вектор, який показує виконання чи невиконання умови запиту.

Примітка. Колекція BitArray є реалізацією бітового масиву.

4. **Варіанти завдання**:

№	Пошук у колекції	Функція об'єднання слів у запиті	Об'єкт синхронізації потоків
1.	**HashTable**	**a AND b**	**Mutex**
2.	**ArrayList**	**a OR b**	**AutoResetEvent**
3.	**SortedList**	**NOT (a OR b)**	**ManualResetEvent**
4.	**ArrayList**	**a OR b**	**Monitor**
5.	**HashTable**	**a AND b**	**lock {…}**
6.	**Array**	**NOT (a AND b)**	**ReaderWriterLock**
7.	**Queue**	**a AND b**	**AutoResetEvent**
8.	**Stack**	**a OR b**	**ReaderWriterLock**
9.	**ArrayList**	**NOT (a OR b)**	**Monitor**
10.	**HashTable**	**a AND b**	**Mutex**
11.	**ArrayList**	**NOT (a OR b)**	**AutoResetEvent**
12.	**Stack**	**a OR b**	**ReaderWriterLock**

13.	**ArrayList**	**NOT (a AND b)**	**Mutex**
14.	**HashTable**	**a AND b**	**Monitor**
15.	**Array**	**a AND b**	**lock {…}**

5. Творче завдання

При виконанні даної лабораторної роботи студенти можуть отримати 2 додаткові бали за творчий підхід до вирішення задачі: реалізувати пошук будь-яким способом, зазначеним у завданні так, щоб можна було отримати позицію першого слова знайденої фрази у документі.

6. Контрольні запитання

6.1. Пояснити розбіжності у часі виконання алгоритму бінарного пошуку та пошуку в колекції.

6.2. Які недоліки та переваги має клас ArrayList у порівнянні з іншими видами колекцій?

6.3. Як впливає операція боксінгу на швидкість внесення та пошуку даних у колекціях. Пояснити відповідь.

6.4. Яким чином забезпечити обмін даними між потоками в .NET-додатку?

6.5. Яку роль відіграють інтерфейси IList, ICollection, IDictionary у реалізації колекцій.