**Лабораторна робота №1**

Шевченко Н. В.

**Тема:** Засоби створення багатопоточних програм.

**Мета:** Познайомитись із засобами побудови паралельних програм різними мовами програмування.

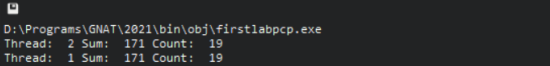
**Ada**

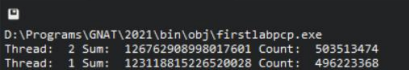
Цей код на мові програмування Ada демонструє використання многозадачності для створення паралельних потоків виконання. На початку програми підключений модуль Ada.Text\_IO, який дозволяє виводити текст у консоль. Оголошені змінні Can\_Stop (типу Boolean) та numOfThreads (типу Integer). Can\_Stop використовується для управління зупинкою виконання програми.

Оголошені два типи завдань (tasks): Break\_Thread: затримує виконання програми на 2 секунди, а потім встановлює значення Can\_Stop в True. Main\_Thread: виконує обчислення суми послідовних чисел до тих пір, поки Can\_Stop не стане True.

В головній частині програми створюється один екземпляр Break\_Thread та масив Threads з Main\_Thread. Кожен Main\_Thread створюється з унікальним ідентифікатором та кроком для обчислення. Потіки запускаються з методом Start з відповідними параметрами.

Після створення потоків вони починають виконувати свою роботу. Main\_Thread обчислює суму послідовних чисел з використанням заданого кроку до тих пір, поки не буде встановлено значення Can\_Stop в True. Коли виконання завершується, результати виводяться у консоль.





**Java**

**Клас Main**

У методі main спочатку створюється екземпляр класу BreakThread, який відповідає за зупинку виконання після певного часу. Визначаються кількість потоків numThreads та крок обчислення step. Створюється пул потоків (ExecutorService) з фіксованою кількістю потоків, які дорівнюють numThreads + 1. Створюється та запускається потік breakThread.

Створюються та запускаються об'єкти SequenceSumThread, які відповідають за обчислення суми послідовних чисел у різних потоках. Після цього пул потоків закривається методом shutdown(), що запобігає додаванню нових завдань в пул, та очікується завершення всіх потоків протягом 10 секунд методом awaitTermination().

Якщо час очікування закінчується, то потоки завершуються за допомогою shutdownNow().

**Клас BreakThread**

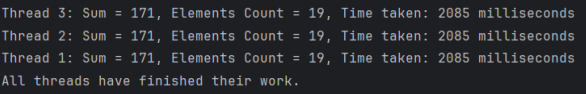
Цей клас реалізує інтерфейс Runnable та містить логіку для зупинки виконання після певного часу. Має поле canBreak, яке вказує, чи можна зупиняти виконання.

У методі run() встановлюється затримка на 2 секунди, після чого значення canBreak встановлюється в true.

**Клас SequenceSumThread**

Цей клас також реалізує інтерфейс Runnable та містить логіку для обчислення суми послідовних чисел. Має поля для зберігання номеру потоку, кроку, суми, кількості елементів та посилання на об'єкт BreakThread.

У методі run() обчислюється сума послідовних чисел до тих пір, поки не буде встановлено значення canBreak в true. Кожне число додається до суми з заданим кроком, після чого потік затримується на 100 мілісекунд. Після завершення обчислень виводяться результати у консоль.



**Висновок**

В даній лабораторній роботі було продемонстровано створення багатопоточних програм на мовах Ada та Java для виконання паралельних обчислень. В Ada використовувалися завдання (tasks), а в Java - ExecutorService та інтерфейс Runnable.

Обидві програми створювали потоки для обчислення суми послідовних чисел та потік для зупинки виконання після певного часу. Багатопоточність дозволяє виконувати завдання паралельно, що може підвищити продуктивність програм.