**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу  
Кафедра системного проектування**

**Звіт**

про виконання лабораторної роботи №2

«Дослідження атомарних змінних та атомарних операцій»

Виконала:  
студентка IІІ курсу,

групи ДА-21  
Щербинко Єлизавета Миколаївна

Київ – 2025

**Мета роботи:** розглянути поняття атомарності, навчитися працювати з атомарними змінними, а також ознайомитися з підходом написання паралельного коду без блокування.

**Завдання:**

1. Ознайомитися з визначенням: атомарна змінна, атомарна операція, неблокуючий алгоритм. Ознайомитися з деталями атомарності в обраній мові програмування.

2. Надати в протоколі роботи опис того, як саме досягається справжня атомарність операцій в обраній студентом мові програмування.

3. Виконати завдання за варіантом без використанням паралелізації. Заміряти час виконання завдання.

4. Виконати завдання за варіантом з використанням блокуючих примітивів синхронізації. Заміряти час виконання завдання.

5. Виконати завдання за варіантом з використанням атомарних змінних та CAS\CMPXCHNG операцій (більш високорівнені функції, що абстрагують дані операції, не приймаються для використання в даній роботі). Заміряти час виконання завдання.

6. Повторити пункти 2 – 4 з використанням різної розмірності даних та фіксованою кількістю потоків виконання.

7. В протокол занести отримані для пункту 5 результати у вигляді графіків залежності часу від кількості даних, надати порівняльний аналіз блокуючого та неблокуючого алгоритму з використання атомарних операцій.

8. Надати висновок, що повинен містити аналіз отриманих результатів.

**Варіант 23:**

Знайти кількість елементів, кратних 17, і найменший такий елемент.

**Хід роботи:**

1. **Атомарність** – це властивість, що визначена для операцій, і говорить про те, що операція може мати лише один з двох станів – виконана, або не виконана, не допускаючи втручання інших процесів або потоків у середині виконання операції.

**Атомарні змінні** – це спеціальні типи даних, призначені для коректного використання в паралельних середовищах. Вони гарантують атомарність операцій, тобто виконання окремих операцій без переривань, забезпечуючи коректність результатів, навіть при одночасному доступі декількох потоків.

**Атомарність операції** – набір інструкцій з властивістю неперервності цілої операції. Атомарна операція виконується повністю (або відбувається відмова у виконанні), без переривань. Атомарність має особливе значення в багатопроцесорних комп'ютерах. Доступ до загальних для декількох потоків ресурсів повинен бути обов'язково атомарним.

Атомарність буває апаратна (коли безперервність забезпечується апаратурою) і програмна, коли використовуються спеціальні засоби для взаємодії (м'ютекс, семафор). За своєю суттю програмні засоби забезпечення атомарності представляють собою два етапи: блокування ресурсу і виконання самої операції. Блокування являє собою атомарну операцію, яка або успішна, або повертає повідомлення про зайнятість.

**Неблокуючий алгоритм** – це алгоритм, що забезпечує синхронізацію між потоками без використання блокуючих примітивів, таких як м’ютекс

2. У С++ справжня атомарність операцій досягається за допомогою механізмів, що гарантують виконання операцій без можливості їх переривання іншими потоками.

1) <atomic> – забезпечує атомарні операції над змінними.

2) Вбудовані інструкції процесора (CAS, LOCK, XCHG) – ці механізми і використовує <atomic> під капотом.

3. Виконаємо завдання без паралелізації. Перевіримо правильність виконання програми:

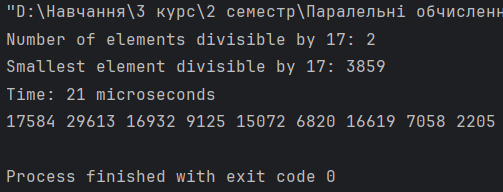


Рис. 1 – Перевірка правильності роботи послідовної програми

Розмірність масиву = 1000:

|  |  |
| --- | --- |
|  |  |
| (а) | (б) |
| Рис. 2 – Результати послідовного виконання | |

4. Виконаємо завдання з використанням блокуючих примітивів синхронізації. Перевіримо правильність роботи програми:

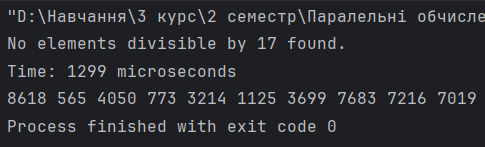


Рис. 3 – Перевірка правильності роботи блокуючого алгоритму

Розмірність масиву = 1000, кількість потоків = 6.

|  |  |
| --- | --- |
|  |  |
| (а) | (б) |
| Рис. 4 – Результати роботи блокуючого алгоритму | |

5. Виконаємо завдання з використанням атомарних змінних та CAS\CMPXCHNG операцій. Перевіримо правильність роботи програми:

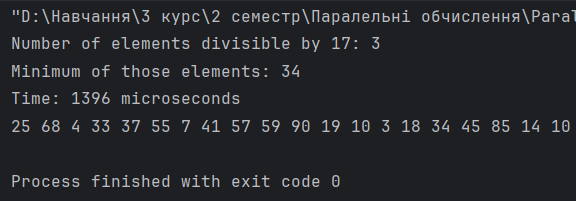


Рис. 5 – Перевірка правильності неблокуючим алгоримом

Розмірність масиву = 1000, кількість потоків = 6.

|  |  |
| --- | --- |
|  |  |
| (а) | (б) |

Рис. 6 – Результати роботи програми з атомарними змінними

6. Виміряємо час для різної розмірності масиву.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (а) – 10000 | (б) – 100000 | (в) – 1000000 |

Рис. 7 – Неблокуючий алгоритм

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (а) – 10000 | (б) – 100000 | (в) – 1000000 |

Рис. 8 – Блокуючий алгоритм

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (а) – 10000 | (б) – 100000 | (в) – 1000000 |

Рис. 9 – Послідовне виконання

7. Створимо порівняльний графік для всіх реалізацій

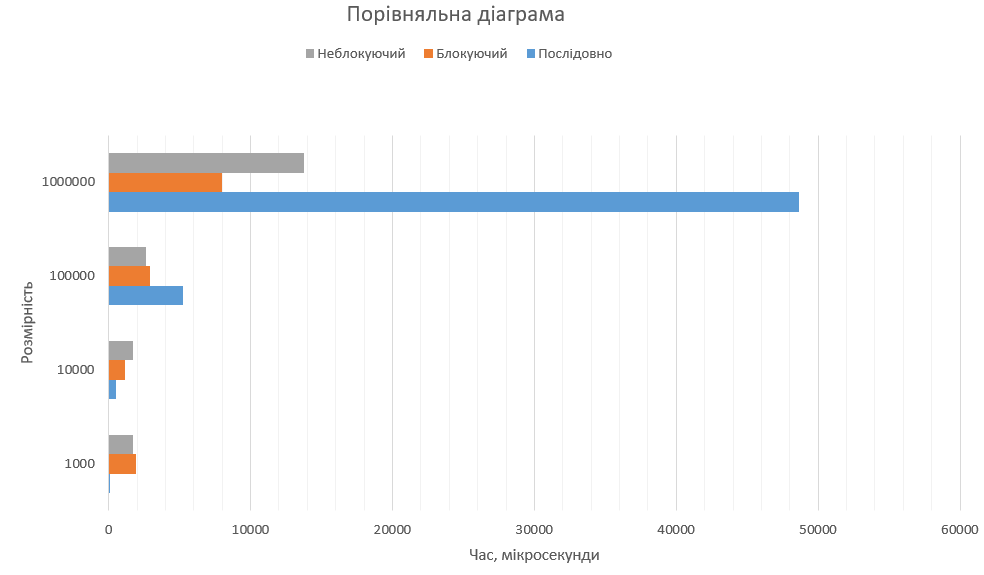


Рис. 10 – Загальна порівняльна діаграма

Порівняємо блокуючий та неблокуючий алгоритм.

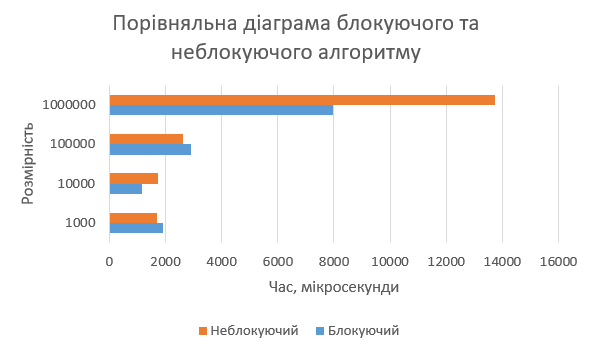


Рис. 11 – Порівняльна діаграма для блокуючого та неблокуючого алгоритму

Неблокуючий алгоритм виграє для малих і середніх обсягів, але при дуже великих обсягах може втрачати ефективність. Блокуючий все ще є найоптимальнішим для великих даних.

Теоретично неблокуючий алгоритм швидший, особливо на середніх і великих обсягах даних. Практично ефективність залежить від деталей реалізації – для дуже великих обсягів необхідно враховувати накладні витрати та можливе перевантаження системи.

Можливо, для найкращої продуктивності варто оптимізувати неблокуючий алгоритм або використовувати гібридний підхід.

**ВИСНОВОК**

У ході виконання лабораторної роботи було розглянуто концепцію атомарності, особливості роботи з атомарними змінними, а також реалізацію неблокуючого коду за допомогою атомарних операцій у порівнянні з блокуючими примітивами синхронізації.

Завдяки використанню атомарних змінних, можна досягти атомарності операцій без потреби в блокуванні, що є важливим аспектом при розробці паралельних програм. Виконання завдання без паралелізації дозволило зафіксувати базові показники часу, на основі яких були порівняні різні підходи до синхронізації потоків.

Реалізація з блокуючими примітивами синхронізації (мьютексами) показала свою ефективність при невеликих та середніх обсягах даних. Однак, використання атомарних змінних із операцією CAS дозволяє досягти кращих результатів при більшій кількості потоків і порівняно малих обсягах даних, оскільки цей підхід мінімізує накладні витрати, пов'язані з блокуванням ресурсів.

Однак, при великих обсягах даних, ефективність неблокуючого алгоритму може знижуватися через накладні витрати, пов'язані з управлінням атомарними операціями, в той час як блокуючий алгоритм з мьютексами залишається конкурентоспроможним для обробки великих обсягів інформації.

Порівняння результатів підтвердило, що неблокуючий підхід є більш ефективним для малих і середніх розмірів масивів, але для великих даних блокуючий алгоритм може бути кращим. Загалом, вибір між блокуючими та неблокуючими підходами залежить від специфіки задачі та обсягу даних.

Отже, для досягнення найкращої продуктивності може бути корисним використання гібридних підходів, де поєднуються переваги як блокуючих, так і неблокуючих алгоритмів, залежно від конкретних вимог та характеристик системи.

[Посилання на гітхаб.](https://github.com/yelyshch/Parallel)