МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
ІМЕНІ ВАСИЛЯ СТЕФАНИКА

Кафедра комп’ютерних наук та інформаційних систем

**Лабораторна робота №3**

з курсу «Паралельні та розподілені обчислення»

на тему:

«Синхронізація паралельних операцій за допомогою механізму подій»

**Виконав:**

студент групи КН-41

Сенів П.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**Перевірив:**

викладач кафедри КНІС

к.т.н., Ізмайлов А. В. «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оцінка, підпис)

м. Івано-Франківськ

2021

***Мета роботи*:** Навчитись синхронізувати паралельні операції за допомогою  механізму подій.

**Теоретичні відомості**

Одним із способів синхронізації паралельних операцій є проста перевірка  спільного для потоків прапорця, який захищений м'ютексом, вважаючи, що один  із потоків підніме цей прапорець, коли завершить свою операцію. Такий підхід  має ряд недоліків, оскільки, передбачає витрачання ресурсів системи на постійну  перевірку значення прапорця. Одним із способів подолання цього недоліку є  застосування функції this\_thread::sleep\_for(), яка змушує потік «спати» між  перевірками значення прапорця. Наприклад, за допомогою наступної  конструкції:

unique\_lock<mutex> lk(m);

while (!flag)

{

lk.unlock();

this\_thread::sleep\_for(chrono::milliseconds(100));

lk.lock();

}

Наведений спосіб не позбавлений більшості недоліків, пов’язаних з  опитуванням значень прапорів, тому рекомендованим способом синхронізації  потоків є використання засобів стандартної бібліотеки С++, які дозволяють  потоку очікувати на події. Найпростіший механізм очікування на події, які  виникають у іншому потоці (наприклад, поява нового завдання), дають умовні  змінні. Концептуально, умовна змінна асоційована з якоюсь подією або іншою умовою, причому, один або кілька потоків можуть чекати, коли ця умова почне  виконуватись. Якщо деякий потік вирішить, що умова виконана, він може  сповістити про це один або кілька потоків, які очікують умовну змінну, в  результаті чого, вони відновлять роботу.

Стандартна бібліотека С++ надає не одну, а дві реалізації умовних змінних:  std::condition\_variable і std::condition\_variable\_any. Обидва класи оголошені в  заголовку <condition\_variable>. В обох випадках для забезпечення синхронізації  необхідна взаємодія з м’ютексами; перший клас може працювати тільки з  std::mutex, другий – з будь-яким класом, який відповідає мінімальним вимогам  до «м’ютексоподібності». Оскільки, клас std::condition\_variable\_any є більш  загальним, то його використання може обійтися дорожче з точки зору обсягу  споживаної пам'яті, продуктивності та ресурсів операційної системи. Тому, якщо додаткова гнучкість не потрібна, то краще обмежитися класом  std::condition\_variable.

**Хід роботи:**

1.

**Код:**

**Результати виконання роботи**:

Висновок………

2)

**Код:**

**Результати виконання роботи**:

3)

**Код:**

**Результати виконання роботи:**

4)

**Код:**

**Результати виконання роботи**:

**Висновок:** на лабораторній роботі я навчився синхронізувати паралельні операції за допомогою  механізму подій. Мої вихідні файли з проектом знаходяться за адресою: <https://github.com/PavloSeniv/parallel-and-distributed-calculations/tree/main/Laba3>