СТРАТЕГІЇ БАГАТОПОТОКОВИХ ІНТЕЛЕКТУАЛЬНИХ УКАЗНИКІВ

Підготував: Житнецький Олексій, студент ІПЗ-3

Керівник: доцент, кандидат фіз.-мат. наук Бублик В. В.

Постановка задачі

- Мета дослідження – знаходження ефективних підходів до проектування потокобезпечного програмного забезпечення в C++ з урахуванням принципів сучасного управління пам’яттю та метапрограмування шаблонів.

- Об’єкт дослідження – процес проектування потокобезпечного програмного забезпечення мовою C++ з використанням засобів STL.

Основи багатопоточності у C++

- Апаратна vs симульована конкурентність.
- Вартість запуску потоку (виділення стеку, перемикання контексту).
- std::thread – клас створення та керування потоками.
- Правила join, detach; метод joinable.
- std::future, std::promise, std::packaged_task, std::async – інструменти для роботи з результатами.

Основи синхронізації потоків

- std::mutex, std::lock_guard – взаємовиключення та RAII.
- std::atomic – безблокові (lock-free) атомарні операції.
- Баланс безпеки та продуктивності.
- Патології: deadlock, thread starvation.

Моделі пам’яті та типові проблеми

- Стек (швидкий, унікальний для кожного потоку, маленький обсяг).
- Купа (повільніша, загальна, великий обсяг).
- Витоки пам’яті, «завислі» указники, подвійний виклик delete.
- Конкурентний код потребує автоматичних рішень.

Інтелектуальні указники

- std::unique_ptr.
  1. Унікальне володіння.
  2. Нульові накладні витрати.
  3. Повернення із функції, семантика переміщення.
- std::shared_ptr та std::weak_ptr.
  1. Лічильник посилань – атомарний та потокобезпечний.
  2. Циклічні залежності, усунення через std::weak_ptr.
  3. Відчутні накладні витрати.

Взірець «стратегія»

- Мінімізує умовні розгалуження, покращує модульність
- Динамічна – інтерфейс та поліморфні класи-реалізації.
- Статична – метапрограмування за допомогою шаблонів.
- Вибір поведінки на етапі компіляції без накладних витрат.

Демонстраційна програма «Архіватор»

- Алгоритм LZ77.
- Паралельне стиснення блоків через std::async.
- Токени стисненої інформації видаються класом TokenPool з кешу.
- Інтелектуальні указники керують циклом життя токенів даних.
- std::mutex забезпечує потокобезпечність мапи кешу.

Висновки

- Систематизовано знання про реалізацію багатопоточності у C++ та засоби керування потоками.
- Досліджено механізми синхронізації, зокрема примітиви std::mutex, std::lock_guard та std::atomic.
- Розглянуто структуру пам’яті програм – стек і купу – та визначено основні проблеми управління пам’яттю у багатопотокових застосунках.
- Досліджено взірець проектування “стратегія” у двох варіантах реалізації — як динамічну поведінкову конструкцію за допомогою поліморфізму, так і як статичну конструкцію етапу компіляції через шаблони.

Перспективи подальших досліджень

- Вивчити actor-model та корутини C++20/23 для кращого вивчення асинхронності.
- Розробити статичний аналізатор для виявлення потенційних станів гонки.
- Дослідити lock-free контейнери та їх вплив на продуктивність.

ДЯКУЮ ЗА УВАГУ!