Параллельное программирование. Задачи и параллельные алгоритмы.

№ урока: 6 **Курс:** C++ Advanced

Средства обучения: Qt Creator

Обзор, цель и назначение урока

Научить студентов понимать и применять на практике базовые механизмы параллельного программирования на основе задач из стандартной библиотеки, разобрать принципы межпоточной коммуникации, показать существование параллельных алгоритмов в новом стандарте.

Изучив материал данного занятия, учащийся сможет

- Понимать, что такое параллельное программирование на основе задач.
- Уметь объяснить разницу между std::async, std::promise, std::packaged_task.
- Понимать, что такое синхронная и асинхронная операции.
- Знать основные механизмы межпоточной коммуникации из стандартной библиотеки.
- Знать особенности стандарта C++17, а именно наличие политик выполнения для параллельных алгоритмов.

Содержание урока

1. Параллельное программирование на основе задач.
2. async.
3. future, promise.
4. Стратегии запуска.
5. Параллельные алгоритмы стандартной библиотеки.

Резюме

«Предпочитайте программирование на основе задач программированию на основе потоков» (Скотт Мейерс)

В std::async передаваемый функциональный объект рассматривается как задача.

Преимущества:

•Имеем возвращаемое значение.
•Возможность получить доступ к вызову через функцию get() у возвращаемого значения.
•Доступ к сгенерированному исключению через функцию get().
•Ответственность за управление потоками лежит на плечах стандартной библиотеки.

Случаи, когда подход на основе потоков предпочтительнее:

•Вам нужен доступ к API, лежащей в основе реализации потоков. (Для предоставления доступа к API реализации потоков std::thread обычно предлагает функцию-член native_handle.).
•Вам требуется возможность оптимизации потоков в вашем приложении.

CyberBionic systematics
CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

•Вам требуется реализовать поточную технологию, выходящую за рамки API па-раллельных вычислений в C++, например пулы потоков на платформах, на кото-рых ваши реализации C++ их не предоставляют.

Std::packaged_task позволяет связать функциональный объект с фьючерсом.
Std::async позволяет выполнять задачу в другом потоке, в текущем, либо отложить выполнение.
Std::promise дает обещание установить значение результата, так, чтобы связанный с ним std::future мог позволить продолжить работу.

В C++17 есть 69 алгоритмов, которые способны выполняться в параллельном режиме, с учетом политик выполнения, таких как последовательный способ, параллельный, с учетом векторизации и некоторые их комбинации.

Закрепление материала

- Чем отличается программирование на основе потоков от программирования на основе задач?
- Чем отличается std::promise от std::packaged_task?
- Для чего необходим std::async? Преимущества перед std::thread?
- Как передать сообщение от одного потока к другому в рамках программирования на основе задач?
- Что такое execution policy?

Дополнительное задание

Задание
Изучите особенности атомарных операций, библиотеку std::atomic.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.

Задание 2
Реализуйте примитивную модель MapReduce распределенных вычислений, с помощью std::async & std::future.

Задание 3
Зайдите на сайт MSDN.
Используя поисковые механизмы MSDN, найдите самостоятельно описание темы по каждому примеру, который был рассмотрен на уроке, так, как это представлено ниже, в разделе «Рекомендуемые ресурсы», описания данного урока. Сохраните ссылки и дайте им короткое описание.

Рекомендуемые ресурсы

https://docs.microsoft.com/en-us/cpp/standard-library/future?view=vs-2017
https://www.modernescpp.com/index.php/promise-and-future
http://scrutator.me/post/2012/06/03/parallel-world-p2.aspx
https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#S-concurrency
https://ru.cppreference.com/w/cpp/thread/async

CyberBionic systematics
CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com