Лабораторная работа №2. Параллельные алгоритмы в языке C++

2.1 Цель лабораторной работы

Получить навыки работы с параллельными алгоритмами в языке C++.

2.2. Теоретический материал

В стандарт C++17 включена поддержка параллельных алгоритмов. Для большинства алгоритмов стандартной библиотеки языка C++ добавлены параллельные версии. Также в библиотеку добавлено несколько новых алгоритмов.

Для того, чтобы использовать параллельную версию алгоритма, нужно первым параметром добавить политику выполнения. Стандарт поддерживает 3 политики выполнения:

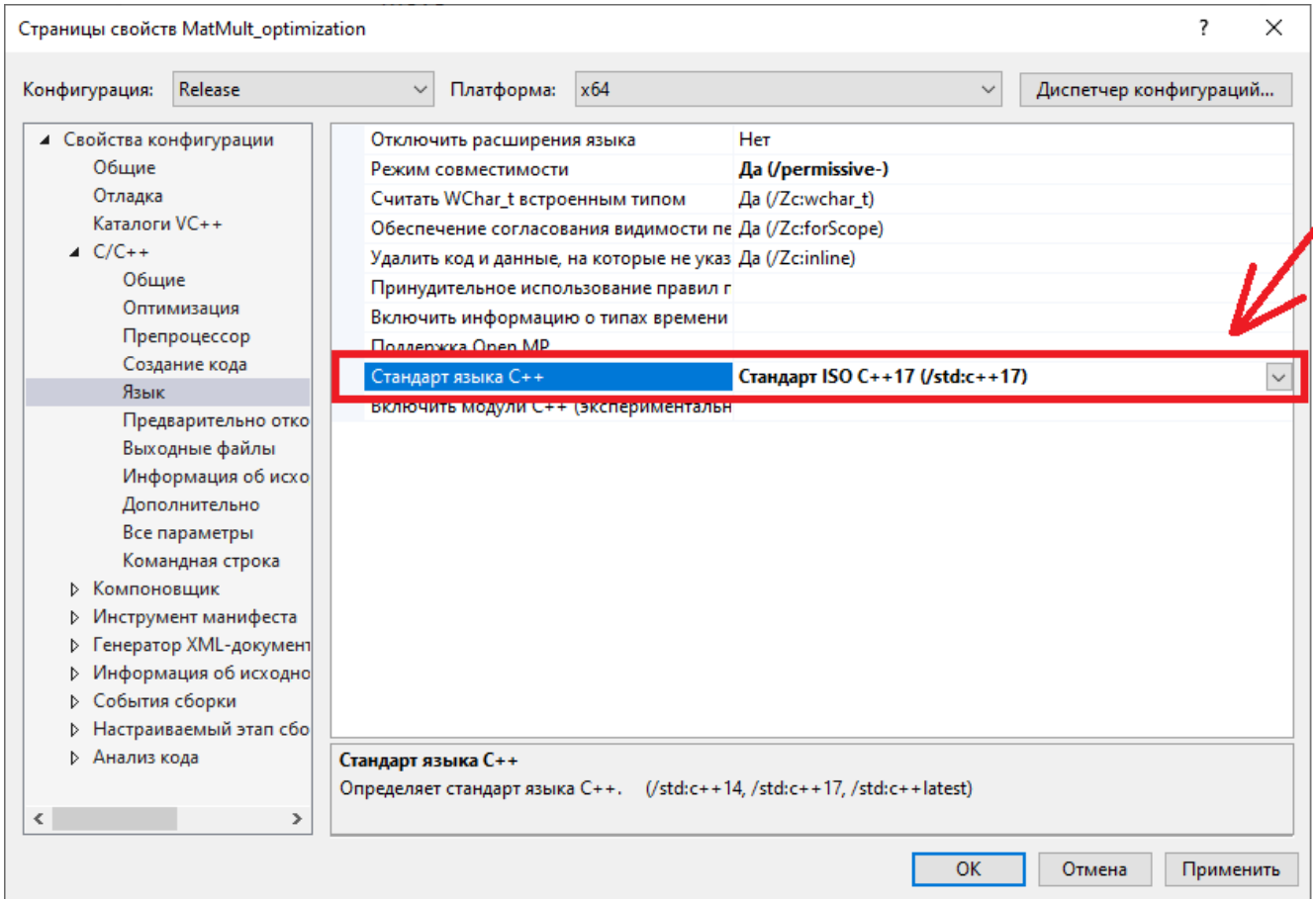
- ✓ ***sequenced_policy*** — тип политики выполнения, указывающий на то, что алгоритм выполняется последовательно; соответствующий глобальный объект — `std::execution::seq`;
- ✓ ***parallel_policy*** — тип политики выполнения, указывающий на то, что алгоритм может выполняться параллельно, пользовательские функции, вызываемые во время работы алгоритма, не должны порождать «гонку данных»; соответствующий глобальный объект — `std::execution::par`;
- ✓ ***parallel_unsequenced_policy*** — тип политики выполнения, указывающий на то, что выполняется распараллеливание и векторизация алгоритма; соответствующий глобальный объект — `std::execution::par_unseq`.

В стандарт C++20 также включена политика выполнения ***unsequenced_policy***, позволяющая векторизовать алгоритм.

Политики выполнения содержатся в заголовочном файле <execution> стандартной библиотеки языка C++.

Политики выполнения поддерживаются MS Visual Studio 2017 начиная с версии 15.8, а также более свежими версиями Visual Studio.

Перед использованием политик выполнения убедитесь, что используемый стандарт не менее C++17 (Свойства проекта – C/C++ – Язык).



Примеры можно посмотреть:

- ✓ https://habr.com/ru/company/intel/blog/346822/ – для компилятора Intel;
- ✓ https://habr.com/ru/company/otus/blog/433588/ – анализ реализации параллельных алгоритмов для компилятора MSVC;
- ✓ https://en.cppreference.com/w/cpp/algorithm – справочник.

2.3. Задание на лабораторную работу

1. Используя параллельные алгоритмы стандартной библиотеки языка C++, написать программу, вычисляющую для каждого элемента целочисленного вектора количество его делителей. Элементы вектора – случайные натуральные числа из диапазона $[10^5, 10^6]$. Результат должен быть записан в новый вектор.

Провести тестирование программы на векторах размера $5 \cdot 10^5$, 10^6, $2 \cdot 10^6$ с различными политиками выполнения. На каждом примере запустить не менее трех раз. В таблицу занести среднее время выполнения на одном примере в секундах. Сравнить результаты. Сделать выводы.

Таблица 1 – Время вычисления количества делителей элементов целочисленного вектора, с

Размер вектора	**Политика выполнения**		
	`seq`	`par`	`par_unseq`
$\mathbf{5 \cdot 10^5}$			
$\mathbf{10^6}$			
$\mathbf{2 \cdot 10^6}$			

2. Используя параллельные алгоритмы стандартной библиотеки языка C++, написать программу, суммирующую элементы вектора.

Указание. Использовать алгоритм `std::reduce`.

Провести тестирование программы на векторах размера $5 \cdot 10^8$, 10^9, $2 \cdot 10^9$ с различными политиками выполнения. На каждом примере запустить не менее трех раз. В таблицу занести среднее время выполнения на одном примере в секундах. Сравнить результаты. Сделать выводы.

3. Используя параллельные алгоритмы стандартной библиотеки языка C++, реализовать классический алгоритм умножения матриц.

Указание. Алгоритмы использовать для вычисления скалярного произведения.

Провести тестирование программы на матрицах размера 512, 1024, 2048 с различными политиками выполнения. На каждом примере запустить не менее трех раз. В таблицу занести среднее время выполнения на одном примере в секундах. Сравнить результаты с результатами лабораторной работы 1. Сделать выводы.

4. Используя параллельные алгоритмы стандартной библиотеки языка C++, написать программу, сортирующую массив действительных чисел.

Протестировать на массивах разной размерности (3-5 вариантов) со всеми политиками выполнения. Замерить время выполнения. Размеры массивов подобрать таким образом, чтобы нагрузить систему. Сравнить результаты. Результаты должны быть представлены в удобном, понятном и наглядном виде.

2.4. Результаты лабораторной работы

Результаты лабораторной работы представляются в виде отчета по лабораторной работе. В отчет включается титульный лист, цель работы, задание на лабораторную работу, описание и обоснование правильности алгоритма, листинг с комментариями, скриншоты, доказывающие правильность работы программы, полученные результаты и выводы по лабораторной работе.

Пример оформления титульного листа приведен на следующей странице.

Отчет оформляется в электронном виде и высылается на e-mail vbyzov.vyatsu@gmail.com (в теме или тексте письма, а также в названии документа с отчетом должны фигурировать ФИ студента, его группа, номер лабораторной работы).

Лабораторная работа считается зачтенной после её устной защиты у преподавателя.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчёт по лабораторной работе №2

по дисциплине «Параллельное программирование»

Параллельные алгоритмы в языке C++

Выполнил: студент группы ФИб-4301-51-00 _____ / И.И. Иванов /

Проверил: ст. преподаватель каф. ПМиИ _____ / В.А. Бызов /

Киров 2021