2. Лабораторная работа №1

2.1. Задание №1

Замер времени работы не такая тривиальная вещь, какой кажется на первый взгляд. Для того, чтобы убедиться в этом проделайте следующие задания.

- Напишите простейшую подпрограмму, которая ждет некоторое время, например 1 секунду. Для ожидания используете встроенную процедуру `sleep` с аргументом 1.
- Замерьте время ее работы N раз с помощью `omp_get_wtime()`. Распечатайте результаты замеров времени, например для $N = 10$. Убедитесь, что почти всегда полученное время отличается от 1.
- Проведем теперь большее количество замеров, например $N = 10^6$. Процедура `sleep` для этого уже не годится, потому что принимает в качестве аргумента только целочисленные значения в секундах. Вместо `sleep` можно использовать какие-то произвольные вычисления в цикле. Имейте ввиду, что компилятор оптимизирует простые вычисления и при трансляции программы может заменить цикл формулой.
- Замерьте время работы новой подпрограммы для $N = 10^6$. Распечатывать 10^6 чисел не надо — их лучше отобразить в виде гистограммы или облака точек.
- Вычислите среднее значение и выборочную дисперсию от полученных замеров.

2.2. Задание №2

Работа с OpenMP начинается с создания необходимого количества потоков и обращения к конкретному потоку.

- Создайте программу с параллельной областью. Распечатайте количество созданных потоков. С помощь какой функции это делается?
- OpenMP позволяет задавать количество потоков динамически без необходимости перекомпилировать программу. OpenMP позволяет сделать это как минимум двумя способами: с помощью переменной окружения и с функции `omp_set_num_threads`. Создайте примеры, иллюстрирующие оба способа. Запустите программу несколько раз, всякий раз изменяя количество потоков и при этом не компилируя ее заново.
- Сколько потоков эффективно поддерживает ваш процессор? Как можно узнать эту информацию? Как в OpenMP можно автоматически создать такое количество потоков, которое процессор поддерживает оптимально?
- Создайте многопоточную программу с четырьмя потоками, которая принимает на вход ряд целых чисел. Первый поток должен просуммировать 1, 5, 9 и т.д. числа; второй поток — 2, 6, 10 и т.д.; третий — 3,7,11; четвертый — 4, 8, 12 и т.д. Результаты суммирования распечатываются с указанием, какой поток какой результат получил.
- Напишите автоматические тесты для данной программы, которые проверяют ее работоспособность для разных последовательностей чисел.

2.3. Задание №3

Почти все встроенные математические функции языка Fortran поддерживают векторные действия с массивами, то есть если им в качестве аргумента передать не скалярное значение, а массив, то функция будет вычислена от каждого элемента массива. Компилятор при этом оптимизирует код и использует SIMD инструкции процессора, что позволит автоматически увеличить быстродействие и получить параллельность вычислений в рамках одного ядра.

В качестве третьего задания придумайте некоторую свою функцию (составив ее произвольным образом из встроенных в Fortran элементарных математически функций) и вычислите ее значение от массива двумя способами.

- Первый способ заключается в вычислении значений функций от элементов массива в цикле, передавая каждый элемент массива в функцию по отдельности.
- Второй способ заключается в передаче всего массива в виде аргумента.

Второй способ должен дать выигрыш в производительности. Замерьте время выполнения с помощью `omp_get_wtime()`. Замеры проводите так, как описано в задании 1.

2.4. Задание №4

Реализовать функцию `map`, которая принимает в качестве первого аргумента функцию заданного вида, а в качестве второго — одномерный массив произвольной длины. Затем она поэлементно применяет переданную ей функцию к каждому элементу массива и возвращает результат этого применения. Передаваемая функция должна вычислять значение полинома от действительного числа. Реализовать точно такой же функционал с помощью функции, объявленной как `elemental`.

Для выполнения данного задания, изучите примеры из методического пособия, показывающие, как передать в одну процедуру другую процедуру в качестве аргумента, как возвратить из функции массив и как объявлять и использовать элементарные (elemental) функции.

Код и тест для данного задания необходимо написать самостоятельно.