Домашняя работа №5

**OpenMP**

**Цель работы:** знакомство со стандартом распараллеливания команд OpenMP.

**Инструментарий и требования к работе:** рекомендуется использовать C, C++. Возможно использовать Python и Java.

# **Порядок выполнения работы:**

1. Изложить в письменной форме:
   1. описание принципа работы OpenMP и ключевых элементов (конструкций и переменных окружения), которые понадобились для написания программы;
   2. описание алгоритма согласно [варианту](#_Варианты);
2. Написать программу согласно [варианту](#_Варианты).

# **Содержание отчета**

1. Теоретическая часть (пункт 1 из Порядка выполнения);
2. Описание работы написанного кода (пункт 2 из Порядка выполнения, экспериментальная часть);
3. Привести графики времени работы программы:
   1. при различных значениях числа потоков при одинаковом параметре schedule;
   2. при одинаковом значении числа потоков при различных параметрах schedule;
   3. с выключенным openmp и с включенным с 1 потоком.
4. Листинг кода с указанием компилятора/интерпретатора (подробнее [Оформление кода в отчёте](#_Оформление_кода_в)).

# **Примечания:**

* + - 1. Файл с отчётом подгружаем в саму форму: <https://vk.cc/bW2r1l>;
      2. В поле «Ссылка на отчет» требуется ссылка на ответ, а не на диск, где лежит отчет. Данное поле заполняется, если по каким-либо причинам не удалось приложить файл в форму. Также можно подстраховаться на случай, если файл приложится некорректно, заполнив это поле. Если ссылка на отчет приложена неверно и в форме нет приложенного файла, то отчет не принимается;
      3. В поле «Ссылка на код» можно вставлять как ссылку на файл (исходного кода либо архив с исходниками), так и на диск (где именно вы будете хранить файлы не принципиально, главное – открытый доступ по ссылке до обозначенного времени) или репозиторий (git – репозиторий должен быть закрытым и расшаренным со мной (RonoveRaum));
      4. «Шаблон отчета»: <https://vk.cc/aAWqZm>;
      5. **Важно:** Будет оцениваться как правильность результата, так и скорость работы.

# **Дополнительные сведения (код)**

1. Аргументы программе передаются через командную строку:

**hw5.exe <кол-во\_потоков> <параметры\_алгоритма> [<имя\_выходного\_файла>]**

Если указано последнее, то результат работы пишется в этот текстовый файл;  
Число потоков может равняться 0 и больше. 0 соответствует значению числа потоков по умолчанию.

1. Корректно выделяется и освобождается память, закрываются файлы, есть обработка ошибок: не удалось открыть файл, формат файла не поддерживается.  
   Если программе передано значение, которое не поддерживается – следует сообщить об ошибке;
2. В программе можно вызывать только стандартные библиотеки (например, <bits/stdc++.h> таковой не является и ее использование влечет за собой потерю баллов). То есть сторонние библиотеки использовать нельзя (библиотека <omp.h> и модули для подключения openmp конечно разрешены);
3. Если программа использует библиотеки, которые явно не указаны в файле с исходным кодом (например, <algorithm>), то за это также будут снижаться баллы.

# **Оформление кода в отчёте**

* + - 1. Никаких скринов кода – код в отчет добавляется только текстом;
      2. Шрифт: Consolas (размер 10-14 на ваше усмотрение);
      3. Выравнивание по левому краю;
      4. Подсветка кода допустима. Текст должен быть читаемым (а не светло-серый текст, который без выделения на белом не разобрать);
      5. В раздел Листинг код вставляется полностью в следующем виде:

**<Название файла>**<Его содержимое>  
Файлы исходных кодов разделяются новой строкой.  
Например,  **main.cpp**

int main()

{

return 0;

}

**tmain.cpp**

int tmain()

{

return 666;

}

* + - 1. Фон белый (актуально для тех, у кого копипаста кода идет вместе с фоном темной темы из IDE).

# **Варианты**

**Для всех вариантов:**

Сравните время работы последовательной и параллельной версий (при различных кол-вах тредов). В данном случае имеется в виду время работы алгоритма (без замера времени на считывание данных и вывод результат).

Формат вывода: “\nTime (%i thread(s)): %f ms\n”

Время работы выводится только в консоль, после вывода результата. Если результат выводится в файл, то время работы всё равно выводится в консоль.

Во всех PNM файлах (pgm, ppm) комментарии отсутствуют.

Обратите внимание на параметр schedule директив openmp.

**Easy**

**Вариант 1.** **Вычисление интеграла методом прямоугольников с заданным числом прямоугольников**

Вычисление интеграла методом прямоугольников с заданным числом прямоугольников следующих функций:

**<параметры\_алгоритма> = <число\_прямоугольников> (int; >= 1)**

Формат вывода: “Result 1: %f\nResult 2: %f\n”

**Вариант 2.** **Вычисление интеграла методом прямоугольников с заданной точностью интегрирования**

Вычисление интеграла методом прямоугольников с заданной точностью интегрирования следующих функций:

**<параметры\_алгоритма> = <точность\_интегрирования> (float; < 1.f)**

Формат вывода: “Result 1: %f\nResult 2: %f\n”

**Вариант 3. Перемножение матриц**

Перемножение матриц типа float размерами NxK и KxM.

**<параметры\_алгоритма> = <имя\_входного\_файла>**

Входной файл содержит данные следующим образом:

n k m

/\* первая матрица, элементы строки разделены пробелом \*/

/\* вторая матрица, элементы строки разделены пробелом \*/

Пример:

4 2 3

1.0 2.0

3.0 4.0

2.0 5.2

6.3 7.8

10.10 11.11 12.12

13.13 14.14 15.666

Вывод:

<размерность\_матрицы (строки столбцы)>

<матрица, элементы строки разделены пробелом>

**Normal**

**Вариант 4. Подсчет простых чисел**

Подсчет количества простых чисел в интервале от 2 до n методом проверки делимости на все нечётные числа от 3 до .

**<параметры\_алгоритма> = <n> (int; > 2)**

Формат вывода: “Result: %i\n”

**Вариант 5. Префиксная сумма**

Подсчет инклюзивной префиксной суммы для заданного массива типа float.

**<параметры\_алгоритма> = <имя\_входного\_файла>**

Входной файл содержит данные следующим образом:

n

/\* входной массив данных, элементы разделены пробелом \*/

Вывод:

n

/\* выходной массив данных, элементы разделены пробелом \*/

**Вариант 6. Gaussian Blur**

Применение гауссовского размытия к изображению в оттенках серого.

**<параметры\_алгоритма> = <ksize> <sigma> <имя\_входного\_файла>**

* **<ksize> - диаметр ядра (kernel size), нечётное целое число**
* **<sigma> - параметр алгоритма, положительный float**

Типичное значение диаметра ядра составляет .

Входной файл содержит данные в формате PGM (P5).

В качестве выходного файла будет новое изображение в формате PGM (P5). Имя выходного файла для данного варианта гарантировано будет указано в аргументах.

**Hard**

**Вариант 7. Определитель матрицы**

Определение детерминанта квадратной матрицы типа float.

**<параметры\_алгоритма> = <имя\_входного\_файла>**

Входной файл содержит данные следующим образом:

n

/\* матрица, элементы разделены пробелом \*/

Пример:

4

1.0 2.0 8.8 9.9

3.0 4.0 3.3 4.4

2.0 5.2 2.2 5.5

6.3 7.8 6.3 7.8

Формат вывода: “Determinant: %g\n”

**Вариант 8. Gaussian Blur (с использованием** **Box Blur Approximation)**

Применение гауссовского размытия с применением Box Blur Approximation к изображению в оттенках серого.

**<параметры\_алгоритма> = <numbox> <sigma> <имя\_входного\_файла>**

* **<numbox> - number of boxes, натуральное число**
* **<sigma> - параметр алгоритма, положительный float**

Входной файл содержит данные в формате PGM (P5).

В качестве выходного файла будет новое изображение в формате PGM (P5). Имя выходного файла для данного варианта гарантировано будет указано в аргументах.

**Brutal**

**Вариант 9. Нормализация яркости изображения**

Автоматическая коррекция яркости изображения в цветовом пространстве YCbCr.601.

Значение пикселей изображения находится в диапазоне [0; 255]. Изображение может иметь плохую контрастность: используется не весь диапазон значений, а только его часть. Например, если самые тёмные места изображения имеют значение 20.

Задание состоит в том, чтобы изменить значения пикселей таким образом, чтобы получить максимальную контрастность (диапазон значений [0; 255]) и при этом не изменить цветность (оттенок). Этого можно достигнуть регулировкой контрастности в канале яркости Y цветового пространства YCbCr (601 в PC диапазоне: [0; 255]).

Важно: согласно стандарту PNM изображения хранятся в цветовом пространстве RGB.

**<параметры\_алгоритма> = <имя\_входного\_файла>**

Входной файл содержит данные в формате PPM (P6).

В качестве выходного файла будет новое изображение в формате PPM (P6). Имя выходного файла для данного варианта гарантировано будет указано в аргументах.

**Вариант 10. Псевдотонирование изображения**

Применение алгоритма дизеринга к изображению в формате PGM (P5) с учетом гамма-коррекции.

**<параметры\_алгоритма> = <битность> <гамма> <имя\_входного\_файла>**

**<битность> - битность результата дизеринга [1..8]**

**<гамма>: 0 – sRGB гамма, иначе – обычная гамма с указанным значением**

Вид дизеринга: ordered 8x8

Входной файл содержит данные в формате PGM (P5).

В качестве выходного файла будет новое изображение в формате PGM (P5). Имя выходного файла для данного варианта гарантировано будет указано в аргументах.

Если кто-то всё-таки желает взяться за это задание, то для дальнейших комментариев нужно написать Виктории (@viktoriya\_yve).