Лабораторная работа №1

«Простые потоки»

**Цель работы**

Получить общие представления о потоках, о способах их создания и использования.

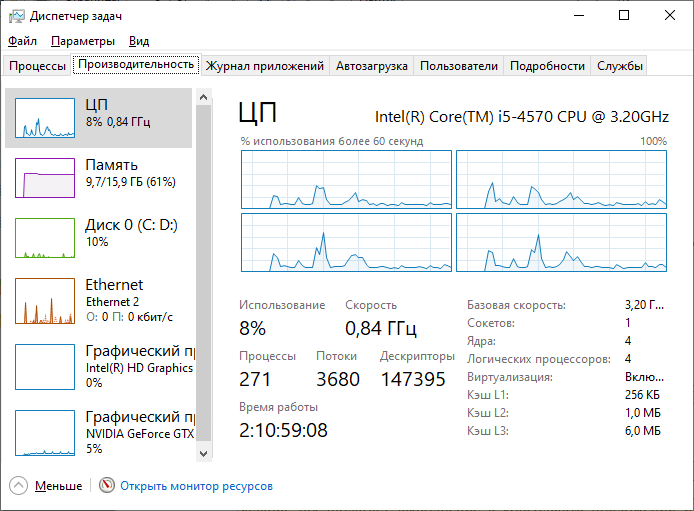
**Основные теоретические положения**

Лабораторная работа выполняется на основе материала, изученного в ходе лекций и лабораторных работ по дисциплине «Объектно-ориентированное программирование» по теме «Работа с потоками в C#».

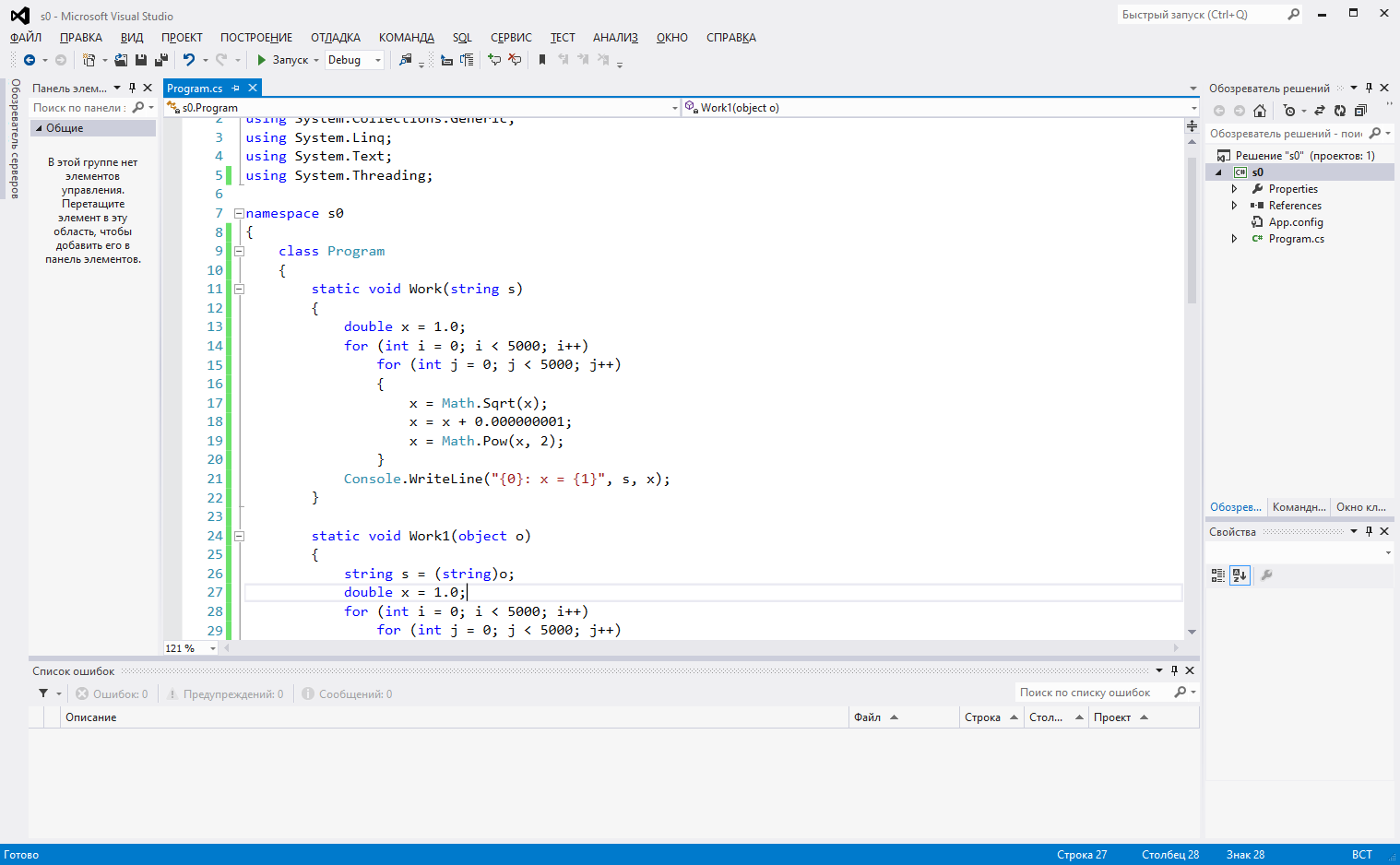
Задание предполагает реализацию консольного приложения на компьютере с многоядерным процессором.

**Порядок выполнения работы**

**Шаг 1**. Перед началом выполнения определите, сколько ядер на Вашем процессоре (далее будем предполагать что N). Для этого в Диспетчере задач (вызов Ctrl+Shift+Esc) откройте вкладку «Производительность». На тестовом компьютере (а в его роли выступал довольно немолодой домашний стационарный компьютер) их оказалось 4.

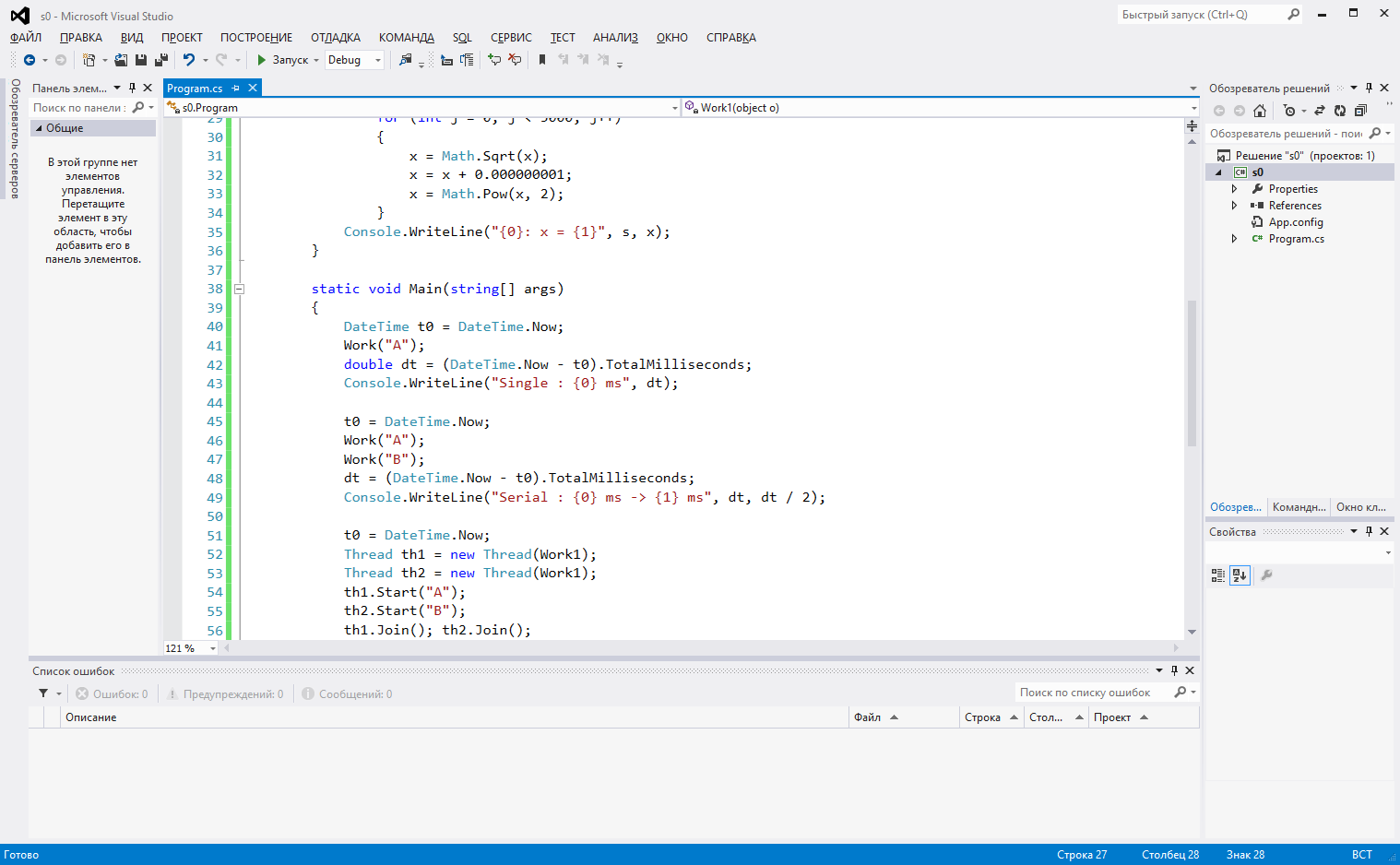


**Шаг 2**. Для начала посмотрим, насколько быстро работает наш компьютер. Для этого поручим ему выполнить довольно большое количество однотипных операций, которые оформим в виде статического метода Work():

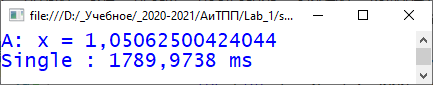


Оценим время вызова данного метода. Для этого используем типы DateTime (явно) и TimeSpan (неявно).

Результат работы получаем в миллисекундах.

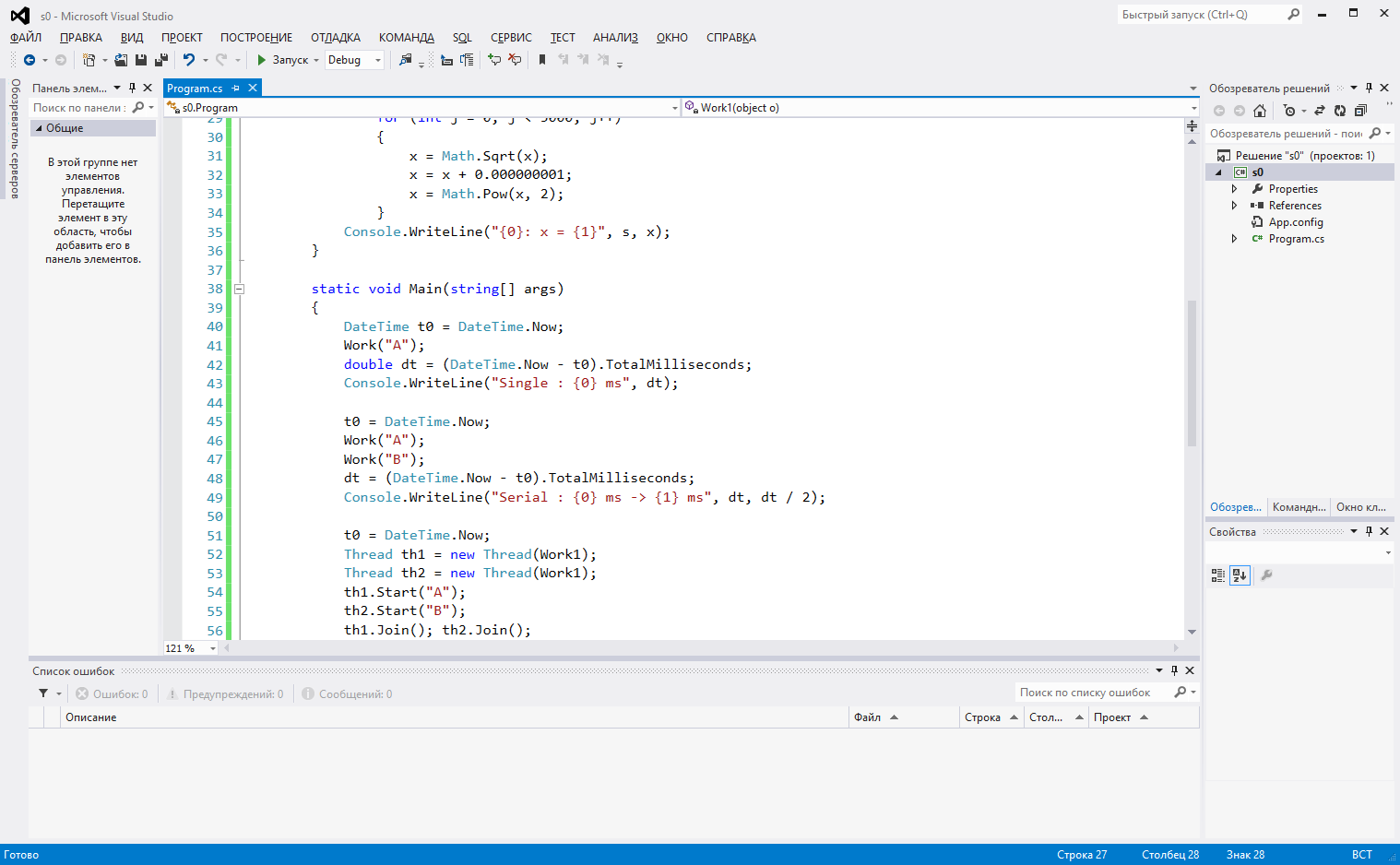


В результате получаем почти две секунды работы.

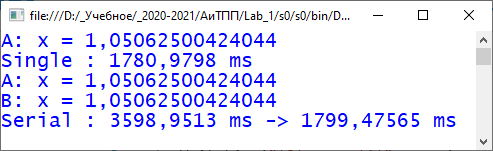


Это на тестовом компьютере. На вашем результат будет, очевидно, другой.

**Шаг 3**. Проверим, насколько долго компьютер будет выполнять две такие задачи. Для того, чтобы отличить, результаты какого из двух вызовов метод Work() выводятся на консоль, метод вызывается со строковым параметром (буквы А и В):

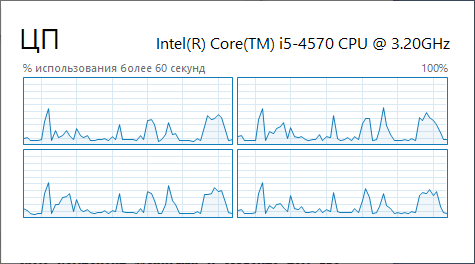


Как и ожидалось, время работы увеличилось примерно в 2 раза:

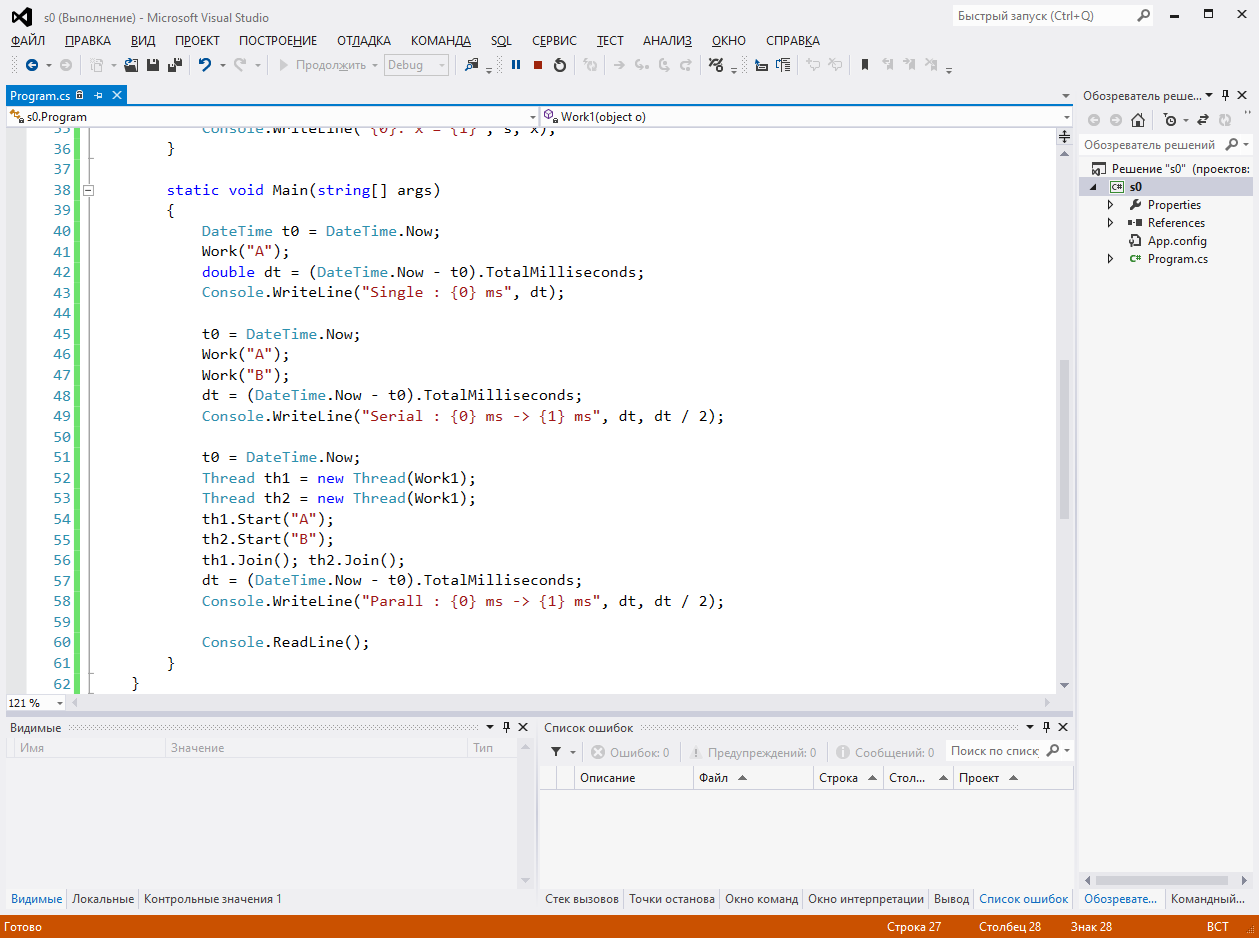


**Объясните, почему выполнение метода Work(), а мы его запускали уже 4 раза, всё время, хоть немного, но разное? При этом результаты расчётов всегда одинаковые.**

**Шаг 4**. Если посмотреть на результат загрузки процессора нашего компьютера, то заметно, что при выполнении программы он не сильно то и старался – его ядра больше простаивали, чем работали (на рисунке ниже это бугорок на последних 20% временной шкалы.

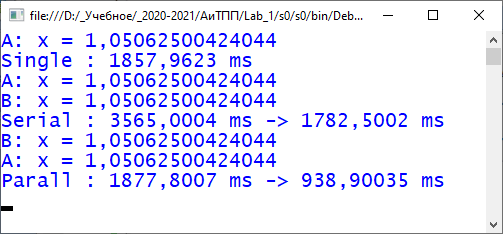


Попробуем нагрузить процессор одновременным (а не последовательным, как на шаге 3), выполнением обоих экземпляров метода Work(). Для этого воспользуемся потоками, работа с которыми инкапсулирована в классе Thread.



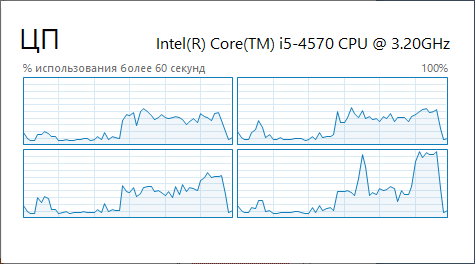
Здесь создаются два объекта th1 и th2, каждый из которых управляет независимым потоком (нитью), обслуживающим отдельный экземпляр метода Work(). После создания методом Start() потоки запускаются на выполнение. Метод Join() приостанавливает текущий поток (в данном случае это главный поток консольного приложения), пока указанный поток не завершит свою работу. Если этого не сделать, то сначала вычислится интервал времени dt (а он несомненно будет равен 0), и потом кванты времени будут получать дополнительные потоки.

Обратите внимание, что в потоке запускается не сам метод Work(), а его брат-близнец, метод Work1(). Единственное его отличие в том, что он принимает на вход не строку, а ссылку на объект, а в самом методе по этой ссылке извлекается строка. Идентификатор вызова метода (буква А или В) указывается при вызове Start().



Как видим, общее время, затраченное на запуск обеих экземпляров метода Work() в режиме распараллеливания оказалась равно времени запуска одного экземпляра этого метода в обычном режиме. Другими словами, мы сделали в два раза больше работы за одно и то же время. Этого удалось добиться за счёт большей нагрузки на каждое ядро процессора. На рисунке это немного сильнее приподнятая правая часть последнего бугорка нагрузки.

Более наглядно это видно, если мы увеличим число итераций каждого из циклов метода Work() с 5000 до 10000 (посмотрите на график для 4-го процессора, принявшего на себя всю нагрузку):



**Кстати, объясните, почему в параллельном режиме экземпляр метода Work() с индексом В завершился раньше, чем его напарник, хотя был запущен позже.**

Остальные шаги вам придётся выполнять самостоятельно. Для отображения результатов надо поэтапно заполнять указанную таблицу:

Таблица 1. Сравнение времени последовательных и параллельных вычислений.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Организация алгоритма | Последовательная обработка | | | Параллельная обработка | | |
| общее время работы | время работы одного метода | порядок  вывода | общее время работы | время работы одного метода | порядок  вывода |
| *(1)* | *(2)* | *(3)* | *(4)* | *(5)* | *(6)* | *(7)* |
| 1 блок |  |  |  |  |  |  |
| 2 блока |  |  |  |  |  |  |
| 3 блока |  |  |  |  |  |  |
| … |  |  |  |  |  |  |
| N блоков |  |  |  |  |  |  |
| N+1 блоков |  |  |  |  |  |  |
| N+2 блоков |  |  |  |  |  |  |

**Шаг 5**. Определите, сколько времени будет выполняться при последовательной обработке программа, содержащая два таких фрагмента следующих друг за другом; три фрагмента и т.д., до N+1, где N – количество ядер процессора Вашего компьютера. Запишите результаты в столбец (2) таблицы. В столбец (3) таблицы запишите среднее время потраченное на один метод. А в столбец (4) запишите порядок завершения вызванных методов. По данным 3-го шага нашего примера в строке «2 блока» можно было бы написать

|  |  |  |
| --- | --- | --- |
| **3599 ms** | **1799 ms** | **A, B** |

**Шаг 6**. Определите, сколько времени будет выполняться при многопоточной обработке программа, содержащая два таких фрагмента следующих друг за другом; три фрагмента и т.д., до N+1, где N – количество ядер процессора Вашего компьютера. Запишите результаты в столбец (5) таблицы. В столбец (6) таблицы запишите среднее время потраченное на один метод. А в столбец (7) запишите порядок завершения вызванных методов. По данным 4-го шага нашего примера в строке «2 блока» можно было бы написать

|  |  |  |
| --- | --- | --- |
| **1878 ms** | **939 ms** | **В, A** |

**Шаг 7**. По данным столбца (6) исследуйте, как влияет количество потоков по сравнению с числом ядер процессора на время их выполнения. Объясните причину обнаруженной Вами закономерности.

**Шаг 8**. Организуйте Таблицу 2 «Влияние приоритета потока на время вычислений»

Исследуйте, как влияет приоритет потоков на порядок их выполнения. Присвойте каждому потоку разный приоритет из 5 возможных вариантов приоритета и для каждого из возможных вариантов заполните соответствующую ячейку таблицы. Сравните результаты, полученные при количестве потоков меньше количества ядер процессора, при количестве потоков равному количеству ядер, при количестве потоков больше количества ядер. Сделайте вывод: в каких ситуация приоритет повлиял на время выполнения вычислений, в каких нет.

Если число блоков меньше числа вариантов приоритета, то организуйте программу так, чтобы для данного числа блоков было несколько попыток запуска этих блоков с разными приоритетами. Например для трёх блоков можно сделать две попытки с вариантами (Lowest, BelowNormal, Normal) и (AboveNormal, Highest, Normal).

Если число блоков больше числа вариантов приоритета, то использовать все варианты, при этом некоторые варианты можно повторять. Например для 6 блоков можно сделать одну попытку с вариантами (Lowest, BelowNormal, Normal, AboveNormal, Highest, Normal).

Если для данного числа блоков было несколько запусков метода Work() с оинаковым вариантом приоритета, то в соответствующую ячейку таблицы 2 следует записать среднее значение.

Таблица 2. Влияние приоритета потока на время вычислений.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Число блоков | Среднее время работы одного экземпляра метода | | | | |
| Lowest | BelowNormal | Normal | AboveNormal | Highest |
| *(1)* | *(2)* | *(3)* | *(4)* | *(5)* | *(6)* |
| 1 блок |  |  |  |  |  |
| 2 блока |  |  |  |  |  |
| 3 блока |  |  |  |  |  |
| … |  |  |  |  |  |
| N блоков |  |  |  |  |  |
| N+1 блоков |  |  |  |  |  |
| N+2 блоков |  |  |  |  |  |

**Требования к отчету**

Отчет по лабораторной работе должен состоять из разделов:

1. информация о студенте – ФИО, группа;

2. цель работы;

3. информация о компьютере, на котором выполнялись задания (наименование процессора, количество ядер, тактовая частота);

4. заполненная таблица 1;

5. заполненная таблица 2;

6. выводы по шагу 7 с указанием установленной зависимости времени вычислений от числа потоков и ядер процессора;

7. выводы по шагу 8 с указанием установленных потокам приоритетов и времени их выполнения;

8. скриншоты, демонстрирующие результаты работы программы (консоль, окно Диспетчера задач, загрузка ядер процессора), с комментариями;

9. исходный код приложения с комментариями в тексте (по одному комментарию к каждому методу, каждой значимой переменной, каждому циклу и каждому этапу работы).