# Практическое задание №1

**Запуск потоков посредством класса thread. Ожидание завершения – join(). Отключение объекта от системного потока – detach(). Определение количества исполняемых ядер. Измерение и задание интервалов времени – chrono. Блокировка потока на указанный интервал. Функции для работы с текущим потоком, пространство имен this\_thread. Класс jthread. Класс std::mutex. Класс lock\_guard.**

## Задание 1. Запуск потоков с ожиданием завершения

Позаимствуйте тексты, например, на <http://do-you-speak.ru/elementary-pre-intermediate> и создайте для каждого текста файл - \*.txt. Задайте спецификации существующих на диске (в текущей директории или в отдельной поддиректории) текстовых файлов, например:   
const char\* filespec[] = {"1.txt", "2.txt", "3.txt", "4.txt"…};  
или std::string filespec[] = {"1.txt", "2.txt", "3.txt", "4.txt"…};  
Обработку каждого файла требуется сделать в отдельном потоке. Каждый поток должен прочитать содержимое файла, перевести все буквы в верхний регистр (а цифры и другие символы оставить прежними) и записать в новый файл. Имена новых файлов можно формировать с помощью заданного префикса или суффикса,   
например: 1.txt -> \_1.txt

Поэкспериментируйте с количеством одновременно запускаемых потоков, каждый раз измеряя время выполняемого фрагмента. Время выполнения хотелось бы выводить в привычном для пользователя виде, например:

2s::10ms::150mks

Подумайте, какие часы Вы будете использовать для измерения времени? А может быть, стоит сравнить использование разных часов (каких?)

В общем случае удобно было бы иметь возможность задавать количество запускаемых потоков.

1.а Последовательно запускаем потоки. Очередной поток запускаем только после завершения текущего

Измерьте время работы фрагмента.

1.б Запускаем столько потоков, сколько файлов требуется обработать

Измерьте время работы фрагмента.

1.в Запускаем столько потоков, сколько ядер в системе

Измерьте время работы фрагмента.

## Задание 2. Запуск обособленных потоков. Блокировка потока на заданный интервал времени

Каждый поток должен «издать» заданное количество звуков с заданной частотой.

В ОС Windows для этого используется системная функция Beep(), другие ОС предоставляют свои системные средства.

Соответствие:

Beep(261, <интервал\_в\_мс>); // соответствует ДО

Beep(293, интервал\_в\_мс); // РЕ

Beep(329, интервал\_в\_мс); // МИ

Beep(349, интервал\_в\_мс); // ФА

Beep(392, интервал\_в\_мс); // СОЛЬ

Beep(440, интервал\_в\_мс); // ЛЯ

Beep(493, интервал\_в\_мс); // СИ

На крайний случай можно «пискнуть» с помощью  
std::cout<<’\a’;  
к сожалению, изменить частоту звука в этом случае невозможно.

После запуска потоков требуется отключить потоки от соответствующих объектов thread.   
**Но!** при завершении первичного потока в ОС Windows все остальные потоки принудительно завершаются => нужно каким-то образом заставить первичный поток завершиться после того, как завершились все вторичные. Например, отправить его на длительное время «в спячку» - std::this\_thread::sleep\_for()

2.a

Сыграйте гамму

2.б

Попробуйте сыграть аккорд??? Подсказка: функция Beep() использует системный динамик для генерации звука и не возвращает управление, пока не истек заданный интервал

## Задание 3.this\_thread

В первичном потоке выведите на экран алфавит (‘A’ – ‘Z’) таким образом, чтобы скорость вывода плавно увеличивалась.

## Задание 4. Параллельная реализация std::transform(). Определение числа вычислительных ядер. Измерение времени.

Требуется посредством алгоритма std::transform() преобразовать элементы любого базового контейнера (vector, list, deque) с элементами любого арифметического типа (short, int, long long, float, double) в элементы другого по правилу: cont2[i] = abs(cont1[i]);

Выведите пользователю информацию о количестве вычислительных ядер и предоставьте пользователю возможность задавать количество потоков.

Подсказка: количество вторичных потоков может быть на единицу меньше заданного пользователем значения, так как первичный поток тоже может выполнить свою часть работы, пока вторичные потоки выполняют свою.

Определите размер диапазона элементов контейнера источника для каждого вторичного потока. Остаток элементов должен трансформировать первичный поток.

Измерьте время выполнения полной трансформации всего контейнера при задании разного количества потоков. Важно! при этом нужно понимать, что оценка является скорее качественной, чем количественной, а о точности судить трудно, так как для достоверной оценки нужно учитывать большое количество других факторов (в системе параллельно выполняется большое количество системных потоков, пропускная способность общей шины данных, кеширование для каждого вычислительного ядра…)

# Задание 5. Класс jthread, namespace this\_thread, std::mutex

Задан массив, например:

const char\* strings[] = { "Long-long ago..." ,"I am very sorry", "My Dog", …};

Требуется запустить количество потоков, равное числу строк в массиве.

Каждый поток должен выводить свой идентификатор и принимаемую строку, например:

Id: <значение\_идентификатора> string: <строка>

, пока пользователь не нажмет любую клавишу.

# Задание 6. Класс jthread

Задан большой вектор целых чисел (сгенерируйте значения). Требуется узнать, есть ли в векторе заданное значение. Реализовать параллельный поиск по отдельным диапазонам вектора. При обнаружении первого совпадения все потоки должны прекратить работу.

# Задание 7. Класс jthread. mutex

Пустой вектор. Потоки параллельно добавляют каждый свое значение. Как только сумма превысит заданное значение, все потоки должны завершиться!

# Задание 8. std::mutex, std::lock\_guard, std::stop\_source, std::stop\_token, std::stop\_callback

Задана сумма (например, 1000). Требуется запустить несколько потоков, каждый из которых должен с заданным тайм-аутом (например, 1 ms, 2 ms…) «забирать» из общей суммы свою долю, значение которой задается в качестве параметра (например, 1,2,…). Как только любой поток обнаруживает, что «деньги закончились», он завершает выполнение. При этом каждый поток должен вывести диагностику: идентификатор потока и сколько денег поток смог забрать из общей суммы.