# Практическое задание по теме: «Параллельное программирование с блокировками»

## thread\_local, once\_flag, call\_once, mutex, async, future, conditional\_variable

## Задание 1. thread\_local

Дан простой целевой класс, например:

class A{

int m\_a;

…

};

Запустите несколько разных потоков. Каждый поток может любым способом (на стеке, в heap-е, с потоковым временем жизни) создавать любое количество объектов целевого типа A. Реализуйте возможность в любой момент выполнения потока получать количество созданных **в данном** потоке объектов типа A.

## Задание 2. Асинхронный запуск потоков. Получение отложенного результата. Параллельная реализация std::accumulate()

Задача: сформировать среднее значение для контейнеров (vector, deque) с элементами любого типа (int, float,…)

1. Создайте и заполните значениями контейнер с элементами типа int (количество элементов и значения можно формировать случайным образом).  
   Замечание: количество элементов должно быть достаточно большим, чтобы оценить результаты распараллеливания задачи
2. Измерьте время работы std::accumulate() для суммирования элементов от begin() до end()
3. Решите ту же задачу посредством параллельного суммирования:
   1. Исходя из количества вычислительных ядер, запустите посредством std::async()разумное количество потоков, каждый из которых должен получать сумму заданного диапазона
   2. Полученные частичные результаты по мере готовности требуется тоже просуммировать
   3. Измерьте общее время
   4. Попробуйте задать количество потоков   
      меньше, чем количество вычислительных ядер,   
      больше чем количество вычислительных ядер
4. Сравните полученные результаты

## Задание 3. Обработка исключений посредством future

Реализуйте в отдельном дочернем потоке прием целых значений от пользователя.

Если пользователь вводит НЕ целое, должно быть сгенерировано исключение.

Обработку исключения требуется реализовать в родительском потоке.

## Задание 4. Потокобезопасная очередь на базе условных переменных

Требуется обеспечить безопасную работу приложения, в котором:

* несколько потоков генерируют заявки (каждый со своим периодом и значением), складывают заявки в очередь и сигнализируют о готовности ( потоки «клиенты»)
* а один поток «сервер» по мере готовности достает заявки из очереди и обрабатывает (имитация обработки – просто вывод значения на консоль)
* для потокобезопасной очереди реализовать операции копирования и присваивания

# Задание 5. Thread-safe Singleton

Создайте класс, реализующий паттерн проектирования Синглтон посредством std::once\_flag и std::call\_once.

Подсказка: для обертки значения логично использовать std::unique\_ptr.

Какие операции с таким объектом нужно запретить? и как?

1 вариант - один объект в рамках многопоточного приложения

2 вариант – для каждого потока один экземпляр