**КУРСОВОЙ ПРОЕКТ «ПОТОКОБЕЗОПАСНАЯ ОЧЕРЕДЬ», ПУЛ ПОТОКОВ НА БАЗЕ ПОТОКОБЕЗОПАСНОЙ ОЧЕРЕДИ.**

Что нужно сделать: создать потокобезопасную очередь, хранящую функции, предназначенные для исполнения. На основе этой очереди реализовать пул потоков. Этот пул состоит из фиксированного числа рабочих потоков, равного количеству аппаратных ядер.

Когда у программы появляется какая-то работа, она вызывает функцию, которая помещает эту работу в очередь. Рабочий поток забирает работу из очереди, выполняет указанную в ней задачу, после чего проверяет, есть ли в очереди другие работы.

**Класс thread\_pool — реализация пула потоков.**

Минимально требуемые поля класса **thread\_pool**:

1. **Вектор потоков**, которые инициализируют в конструкторе класса и уничтожают в деструкторе.
2. **Потокобезопасная очередь** задач для хранения очереди работ.
3. Остальные поля на усмотрение разработчика.

Минимально требуемые методы класса **thread\_pool**:

1. **work** — выбирает из очереди очередную задачу и исполняет её. Этот метод передаётся конструктору потоков для исполнения.
2. **submit** — помещает в очередь задачу. В качестве аргумента метод может принимать или объект шаблона **std::function**, или объект шаблона **package\_task**.
3. Остальные методы на усмотрение разработчика.

**Шаблонный класс safe\_queue — реализация очереди, безопасной относительно одновременного доступа из нескольких потоков.**

Минимально требуемые поля класса **safe\_queue**:

1. Очередь **std::queue** для хранения задач.
2. **std::mutex** для реализации блокировки.
3. **std::condtional\_variables** для уведомлений.

Минимально требуемые методы класса safe\_queue:

1. **push** — записывает в начало очереди новую задачу. При этом захватывает мьютекс, а после окончания операции нотифицируется условная переменная.
2. **pop** — находится в ожидании, пока не придёт уведомление на условную переменную. При нотификации условной переменной данные считываются из очереди.
3. Остальные методы на усмотрение разработчика.

**Алгоритм работы.**

1. Объявить объект класса **thread\_pool**.
2. Описать несколько тестовых функций, выводящих в консоль своё имя.
3. Раз в секунду класть в пул одновременно 2 функции и проверять их исполнение.