ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Департамент прикладной математики**

# ОТЧЕТ
# К ЛАБОРАТОРНОЙ РАБОТЕ 2
## по дисциплине «Алгоритмизация и программирование»

Работу выполнила
студентка группы БПМ 173 _____ М.В. Самоделкина
дата, подпись

Работу проверил _____ С.А. Булгаков
дата, подпись

Москва 2019

## Содержание

# Постановка задачи

Использовать классы *thread*, *mutex* (или более подходящий подкласс), *lock_guard* / *unique_lock*, *condition_variable*, *atomic*, *future* и *async* при решении заданий (в зависимости от условий задания). Программа дуэль: есть секундант (отдельный поток), отсчитывающий время, и есть два дуэлянта (еще два потока), получающих сигнал о выстреле независимо (параллельно), реализовать выстрелы: первого стрелка – «первый», второго – «второй» и реализовать механизм в соответствии с вариантом 18 (3): один стрелок реагирует мгновенно, стреляет с задержкой, второй наоборот, реализовать таймеры или иной механизм иллюстрирующий эти задержки.

# 1 Основная часть

## 1.1 Общая идея решения задачи

Для решения задачи были использованы классы *thread*, *condition_variable* и *mutex* (*unique_lock*).

## 1.2 Структура и принципы действия

Программа содержит глобальные переменные *mutex lock*, *condition_variable cv* и *bool ready*.

В начале программы инициализируется генератор случайных чисел с помощью функций *srand* и *time*. Далее создаются 3 новых объекта потока: секундант (*sec*) и два дуэлянта (*d1* и *d2*), и связываются с потоком выполнения, новый поток выполнения вызывает, соответственно, функции *go*, *threadFunction1* и *threadFunction2*.

Функция *go* для защиты от одновременного доступа нескольких потоков создает экземпляр класса *unique_lock*, который принимает не захваченный *mutex lock* в конструкторе. Далее выводится сообщение о готовности, с помощью метода *sleep_for* имитируется задержка секунданта перед стартом, глобальная переменная *ready* меняет свое значение на *true* и подается сигнал для старта. После чего *condition_variable cv* уведомляет все ожидающие потоки.

Функция *threadFunction1* принимает сигнал от *cv* мгновенно выводит сообщение о получении сигнала, имитирует задержку с помощью метода *sleep_for* и выводит сообщение о выстреле.

Функция *threadFunction2* принимает сигнал от *cv* имитирует задержку перед получением сигнала с помощью метода *sleep_for* и выводит сообщение о получении сигнала и выстреле.

## 1.3 Процедура получения исполняемых программных модулей

Программный код был скомпилирован с среде *Visual Studio 2017*. Код программы содержится в одном исходном файле. Для ускоренной компиляции программы используются предварительно откомпилированные заголовки *”pch.h”*. Помимо этого никаких дополнительных ключей не добавлялось, использовались ключи, которые добавляются по умолчанию.

## 1.4 Результаты тестирования

Тестирование программы представлено в файле *"Task6.cpp"* в функции *Main()*. Ожидаемый вывод функции:

```
Ready..
Go!
Pushkin react
Dantes react
Dantes shoot
Pushkin shoot
```

# Приложение А

полный код программы

**A.1 - Task6.cpp**

```
#include "pch.h"
#include <iostream>
#include <thread>
#include <mutex>
#include <chrono>
#include <ctime>

std::mutex lock;
std::condition_variable cv;
bool ready = false;

void threadFunction1()
{
        {
                std::unique_lock<std::mutex> lck(lock);
                while (!ready) cv.wait(lck);
        }
        std::cout << "Pushkin react\n";
        std::this_thread::sleep_for(
                std::chrono::seconds(rand() % 10));
        std::cout << "Pushkin shoot\n";
}

void threadFunction2()
{
        {
                std::unique_lock<std::mutex> lck(lock);
                while (!ready) cv.wait(lck);
        }
        std::this_thread::sleep_for(
```

```
                std::chrono::seconds(rand() % 10));
        std::cout << "Dantes react\nDantes shoot\n";
}

void go() {
        std::unique_lock<std::mutex> lck(lock);
        std::cout << "Ready.." << std::endl;
        std::this_thread::sleep_for(
                std::chrono::seconds(rand() % 10));
        ready = true;
        std::cout << "Go!" << std::endl;
        cv.notify_all();
}

int main()
{
        srand((unsigned int)time(NULL));
        std::thread d1(threadFunction1);
        std::thread d2(threadFunction2);
        std::thread sec(go);
        sec.join();
        d1.join();
        d2.join();
        return 0;
}
```