Задание по курсу.
Необходимо портировать (распараллелить) предоставленный вычислительный код на графический ускоритель. Для успешного выполнения задания ваш код должен выполняться как минимум на одном ГПУ. На тестовом сервере доступно 4 ГПУ – две tesla C2050 поколения Fermi, tesla K40 и tesla K20x поколения Kepler. Код должен успешно работать на каждом ГПУ по отдельности и должен быть оптимизирован для каждой архитектуры.

Для начала работы необходимо настроить пути к компилятору gcc 5.0. Для этого необходимо выполнить команду source <path>/gcc.sh (конкретный путь будет добавлен позднее). Чтобы проверить, что вы пользуетесь версией 5.0, необходимо выполнить команду gcc --version

Структура проекта:
Папка **bin** – туда компилируются исполняемые файла проекта.
Папка **include** – содержит два файла, первый описывает классы данных для задачи, второй описывает все данные, с которыми работает программа, а также весь набор функций.
Папка **obj** содержит объектные файлы после компиляции.
Папка **src** содержит все исходные коды вычислительной программы.

Выполнение программы начинается с функции main, которая находится в файле _main_program.cpp. В файле header.h содержится краткое описание всех функций, в том числе тех, которые необходимо изучить.

Примерный план действий:
1) запуск и компиляция программы на разных классах данных. С помощью флага **TRACE** в файле **data_params.h,** можно выполнить профилировку программы, либо воспользоваться известными вами средствами профилирования.
2) найти времяемкие фрагменты кода. Выбрать какой-то из них. В выбранном участке кода найти циклы, которые необходимо преобразовать в ядра. Во время преобразования циклов в ядра необходимо обратить внимание на наличие зависимостей между витками цикла. Если в цикле есть зависимость между витками цикла, то необходимо классифицировать зависимость по ее типу и устранить, если возможно. Зависимости могут быть следующие: приватизируемые переменные, редукционные операции, регулярные зависимости по данным между операциями записи и чтения. Первые два типа не препятствуют распараллеливанию.
3) Получить программу на CUDA, отладить ее эффективность.
4) Получить времена выполнения последовательной (исходной) программы на классах 2/3/4 и получить времена выполнения параллельной программы на ГПУ на тех же классах.
5) написать отчет по проделанной работе.

Данный план является планом минимумом.