Домашка №1: Система сборки CMake

Первая домашка будет про то, как собирать проекты. Мы обсуждали это на первой паре. Везде нужно будет использовать утилиту CMake для сборки многофайлового проекта. Задание будет состоять из нескольких частей.

Часть 0

Для начала возьмите свой код с матрицей, разделите его на файлы hpp и cpp, напишите отдельный main.cpp, который как-то использует класс матрицы. Отмечу, что если ваша матрица написана на собственном subvector, то с ним нужно сделать то же самое.

Напишите CMakeLists.txt, который собирает ваш проект.

Часть 0. Продвинутое

Здесь будет немного сложнее, необходимо написать несколько отдельных файлов с функцией main, которые будут исполнять разный код: это будут тесты. Возьмите матрицу и допишите туда несколько файлов, которые тестируют (проверяют корректность выполнения) некоторых методов класса матрицы. Сделайте не менее 3х тестов. Если есть код, решающий СЛАУ методом Гаусса, то он обязательно должен быть протестирван.

Напишите CMakeLists.txt, который будет собирать **одновременно все тесты**. Структуру проекта можете взять такую

```
.
├── CMakeLists.txt
├── build/
├── src/
│   ├── matrix.cpp
│   ├── matrix.hpp
│   ├── subvector.cpp
│   └── subvector.hpp
└── tests/
    ├── test1.cpp
    ├── test2.cpp
    ├── test3.cpp
    └── ...smth...
```

Добавить дополнительные CMakeLists.txt можете куда угодно, можете вообще никуда не добавлять. Дополнительные баллы даст сборка исходного кода в статическую библиотеку, а затем в подключении её к тестам. Часть 0 (обычная, не продвинутая) подразумевается в этом пункте как уже выполненная.

Часть 1

В этой части надо будет собрать проект, который писали не мы. Клонируем этот проект себе на машину и **пишем для него CMakeLists.txt**. Проект можно реструктурировать по вашему усмотрению и добавить себе в репозиторий с домашкой. Совет: сделайте отдельно папочку с исходным кодом (src/). Подумайте, нужно ли тут собирать исходный код в статическую библиотеку, насколько это оправдано.

Часть 2. Запускаем готовое

В репозитории лежит код, который работает с библиотеками VTK и GMSH. Файл simple.cpp работает только с библиотекой VTK, файл challenging.cpp запустится только при подключении обеих библиотек (и наличии файла t13_data.stl, который уже лежит в той же папке).

Задача: поставить себе необходимое количество библиотек, затем напишите CMakeLists.txt, который позволит собрать проект и получить файлы .vtu как результат исполнения программы. **Внимание:** каждая из программ генерирует по 100 файлов, будьте внимательны и собирайте код в отдельной директории. Скорее всего вы ничего не поймете в тексте этого файлов, но если интерено, то можете их открыть с помощью программы Paraview и посмотреть, что же вы там нагенерировали. Сборка проекта подразумевает в себе получение исполняемого файла, к которому подключены необходимые библиотеки. Далее прикрепляю ссылки на руководства по библиотекам

Как поставить VTK и подключить его с помощью CMake

Как поставить GMSH и подключить его с помощью CMake (это зеркало, так как офф.сайт доступен только через VPN)

В этом пунтке хочу отметить, что CMakeLists.txt должен быть написан так, что он должен корректно отработать на любоей машине. Не подуймайте, здесь он не такой сложный, чтобы писать разные инструкции для Windows и Linux, все однообразно. Как минимум, в CMakeLists.txt не должно быть абсолютных путей, только относительные (они считаются относительно директории, где лежит CMakeLists.txt).

Едиснтсвенный нюанс – это библиотека GMSH. Там можно прописать абсолютный путь, при проверке я поменяю его на свой. Пример того, как подключать GMSH к проекту, мы смотрели на семинаре, так что там придумывать ничего не нужно, только разобраться как поставить GMSH.

Операционные системы

В этой лабораторной советую использовать WSL, которую мы с Вами ставили в прошлом семестре. Ничего дополнительного, кроме консоли вам не будет нужно, вы спокойном сможете написать CMakeLists.txt и через консоль сделать собрать проект. Каждая из библиотек может быть легко поставлена с помощью нескольких консольных команд. Они находятся по запросу в гугле «как поставить *имя библиотеки* на Ubutntu».

Если не хочется использовать WSL, то можно попробовать сконфигурировать CMake для Windows внутри Visual Studio Code: можно почитать это. Если вы планируете дальше писать проект на Windows, то будет хорошо сразу сконфигурировать CMake. По обыкновению, ставить библиотеки на Windows сложнее, чем на Linux, но те, что с задании, должны довольно быстро ставиться.

Оценивание и прием

- «Часть 0» оценивается в 1 балл.
- «Часть 0. Продвинутая» оценивается дополнительно к первой части: 2 балла за собранные тесты + 1 балл, если вы смогли собрать исходники в статическую библиотеку.

- «Часть 1» оценивается в 2 балла.
- «Часть 2» с «simple.cpp» оценивается в 3 балла, с «challenging.cpp» оценивается в 5 баллов.

Единственное, что **нужно будет показать со своей машины – это последнее задание**, так как в нем важно, чтобы библиотеки стояли и проект собирался именно у вас. Это пригодится в дальнейшем.

Итого за лабе можно получить $1 + 2 + 1 + 2 + 3 + 2 = 12$ баллов. Зачетных будет 10, два дополнительных балла можно заработать и потом добавить к своему значению в конце семестра. Дополнительные баллы не учитываются при вычислении «возможного максимума баллов за семестр».

А зачем вообще я это делаю?

Все части лабы призваны показать базовые вещи, для которых используется CMake – сборка своего и чужого кода, а также подключение и сборка сторонних библиотек. Более остро вы все это сможете ощутить, когда начнете писать проекты в несколько человек и когда начнете сами писать свои CMakeLists.txt для них. Все примеры проектов, которые у вас есть так или иначе используют CMake и билиотеки, так что после этой лабы вы сможете их позапускать.