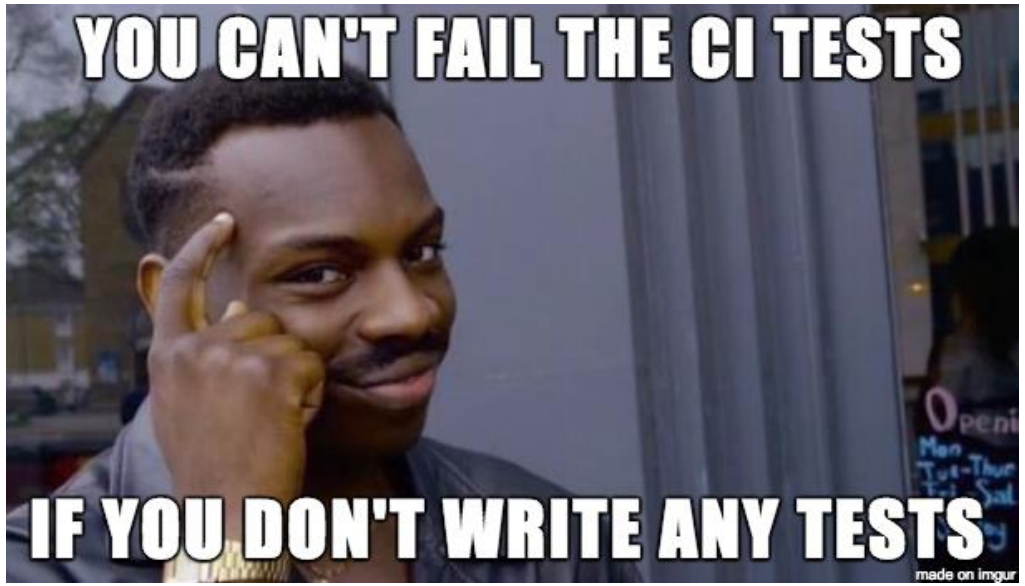
Лабиринт, TDD, Catch2, орг-вопросы

Важные замечания по лабиринту

- Поиск пути не является NP-полной задачей
- Используемую матрицу можно свети к графу и решать с помощью графовых алгоритмов
- Смотрим C++ - код (ещё раз про зло глобальных переменных на примере)

Зачем нужны тесты?

- Алгоритмы – вещь детерминированная
- Требования к входным и выходным данными не изменяются (если произведено надлежащее разделение реализации и интерфейса модуля)
- Учесть и обработать граничные и сложные случаи
- При проведении оптимизаций или изменению вида алгоритма (т.е. при изменении кода) убедиться, что поведение система остаётся таким же



Зачем нужны тесты?

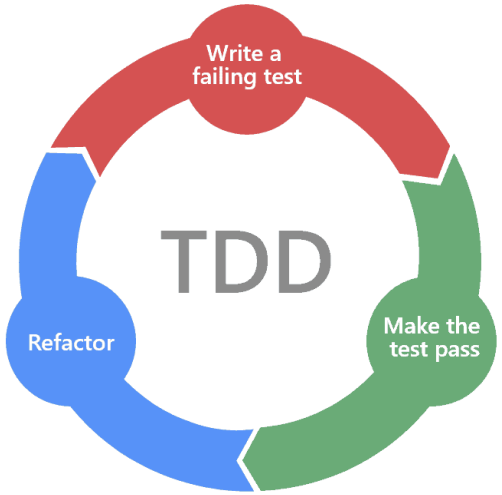
- Алгоритмы – вещь детерминированная
- Требования к входным и выходным данными не изменяются (если произведено надлежащее разделение реализации и интерфейса модуля)
- Учесть и обработать граничные и сложные случаи
- При проведении оптимизаций или изменению вида алгоритма (т.е. при изменении кода) убедиться, что поведение система остаётся таким же

А что если писать тесты перед тем как писать основной код?

Да, так тоже можно! И название этому – TDD.

TDD

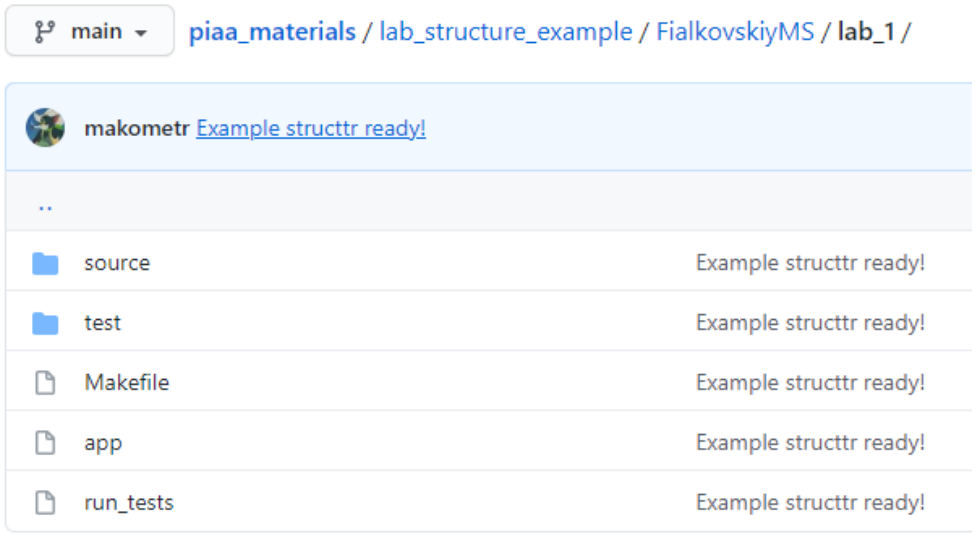
TDD (Test Driven Development) – методология программирования, предполагающая написание тестовых случаев работы перед реализацией



- Заставляет думать перед созданием кода
- Наш случай – юнит тестирование отдельных модулей.
- Наш случай – требования к интерфейсу алгоритма вряд ли сильно изменятся

Требования к сдаваемым работам

- Структура проекта – шаблон уже в репозитории makometr / piaa_materials
- Библиотека тестирования - Catch2 (заголовочник комиттить не надо)
- Сборка makefile/Cmake
- Требования – обновлю
- Табличку сдачи – сделаю
- Чтобы всё работало – сделать вот так →



/ FialkovskiyMS / lab_1 / test / includeCatch.hpp

```
#define MAXIM_CHECK


#ifdef MAXIM_CHECK
    #include "../../../Libs/catch.hpp"
#else
    // ваш путь до библиотеки
#endif
```

Требования к тестам

- Тесты для одной функции должны быть разными, т.е. покрывать различные случаи (быть в разных классах эквивалентости)
- Тестировать – важные части программы и алгоритма, можно и больше. В идеале – всё.
- Готовность дать пояснения, почему тут есть тесты, а там нет и почему они покрывают различные случаи и при этом *все (?)*
- В начале будет сложно и времязатратно, поэтому и задание на отл
- Зато наверняка будет интересно, следуя и TDD