C++

Spring 2021

compscicenter.ru

Basharin Egor

t.me/egorbasharin

Lection IV

Testing

Section 0

Организационные вопросы

- Зачет дорешек
- Добавление тестов к дорешкам

Section 1

Tests

Motivation

- качество программного обеспечения
  - надежность
  - безопасность
  - эффективность
  - ...
- уменьшение рисков
- ...
- в долгосрочной перспективе: оптимизация затрат на поддержку продукта

Types of tests

- **модульные**
- интеграционные
- системные
- приемочные

- performance: load, stress
- регрессионные
- functional
- ...

Automation

- CI
- CD (delivery)
- CD (deployment)

Section 2

Unit tests

Motivation

- обеспечение стабильного роста проекта
- качество кода (+ архитектура)
- косвенная документация
- проект без тестов скорее всего будет переписан с нуля

Нужны ли тесты?

Нужны почти всегда, особенно в долгосрочных проектах.

Вопрос "нужны ли" был вытеснен вопросом "как писать хорошие?".

Не нужны в случае:

> *Вы всегда пишете код без ошибок, обладаете идеальной памятью и даром предвидения. Ваш код настолько крут, что изменяет себя сам, вслед за требованиями клиента. Иногда код объясняет клиенту, что его требования не нужно реализовывать*

Types of projects

- Без тестов
- С тестами, которые не поддерживаются (и запускаются редко)
- Поддерживаемые тесты (встроены в CI)

Хорошие модульные тесты

- не зависят от окружения
- один тест проверяет какую-то одну функциональность
- легко писать и читать
- регулярные запуск (CI)
- сначала тестируется "черный ящик", затем "белый"
- rule of thumb: перед фиксом бага пишет тест на него
- отсутствуют флапающие тесты
- информативные сообщения в случае падения теста

Что тестируем?

- сложный код без зависимостей
- простой код с зависимостями (интеграционный тест)

Как тестируем?

Для тестирования используем AAA-подход

```
AUTO_TEST_CASE( PurchaseInStoreWithEnoughInventary )
{
    // Arrange
    Store store;
    store.addInventory("milk", 10);
    Customer customer;

    // Act
    auto success = customer.Purchase(store, "milk", 5);

    // Assert
    CHECK(success);
    CHECK_EQUAL(5, store.inventory("milk"));
}
```

Метрики тестов

- code coverage
- branch coverage

Метрики тестов

- code coverage
- branch coverage

Проблемы:

- Нет гарантии, что проверены все результаты
- Не учитывается код во внешних библиотеках

Метрики тестов

- code coverage
- branch coverage

Выводы:

- Используйте метрики как индикатор, а не как цель
- Низкое покрытие говорит о проблемах с тестами
- Высокое покрытие не гарантирует высокого качества тестов

Паттерны

- Dependency inversion
- Фейки: mocks & stubs
- Фикстуры

Антипаттерны

- тестирование приватных методов и приватного состояния
- утечка доменных знаний в тесты
- загрязнение кода
- мокирование конкретных классов

TDD (XP)

Состоит из коротких циклов разработки:

- Red: Добавление теста, Запуск тестов
- Green: Написание кода, Запуск тестов
- Refactor: Рефакторинг, Запуск тестов

Book: Modern C++ Programming with Test-Driven Development

Section 3

Unit test framework

Frameworks

- Catch
- Boost.Test
- Google Test
- ...

Full list

Section 4

Fuzzing

Генерация случайных (неправильных) входных данных для проверки граничных случаев.

- libFuzzing (llvm)
- honggfuzz