ОНЛАЙН-
ОБРАЗОВАНИЕ

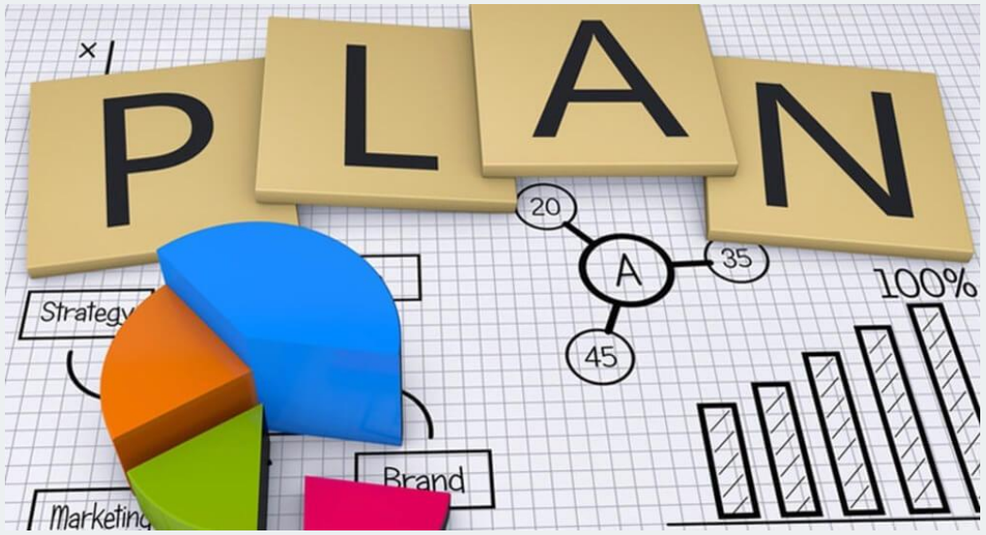
Разработчик C++

Базовый курс

Модульное тестирование

Сергей Кольцов
профессиональный программист

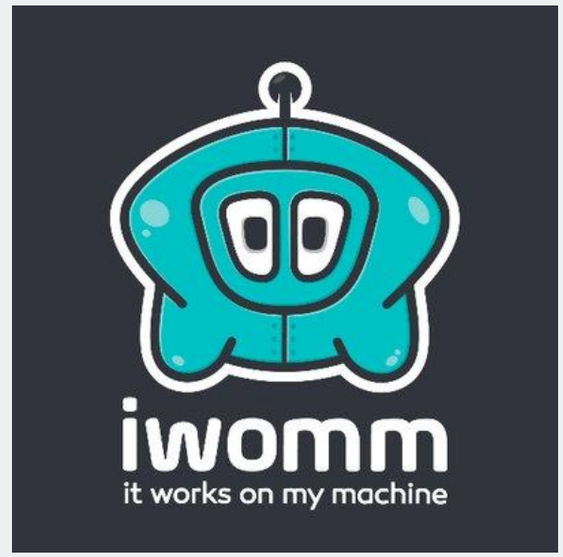
Что будем делать



- Стоит ли писать unit-тесты?
- Google Testing Framework (GTest)
- Как подключить GTest к проекту
- Паттерн «Arrange, Act, Assert» (AAA)
- Типы утверждений в GTest
- Возможности тестового приложения GTest
- Фикстуры (fixtures) и сфера их применения
- GMock - просто о сложном

1 **Что важнее – запланировать или сделать?**

Настройка окружения



Нам понадобятся:

- **CMake** (https://cmake.org/download/)

  CentOS: yum install cmake3

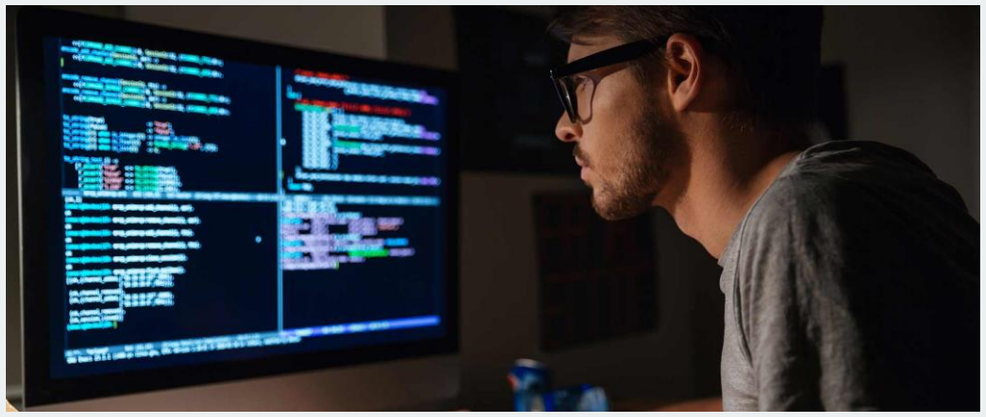
  Ubuntu: apt-get install cmake
- **git** (https://git-scm.com/downloads)

  CentOS: yum install git

  Ubuntu: apt-get install git
- любимый компилятор C++ (11 и выше стандарт)
- консоль

«Есть какая-то связь между теми, с кем расстаешься, и теми, с кем встречаешься.» А. Дюма.

Unit-тесты - стоит ли?



Потенциальные источники проблем:

- рефакторинг
- [поспешное] исправление багов
- добавление новой функциональности
- вырезание старой функциональности
- …
- в общем, почти любая повседневная деятельность программиста ☺

Unit-тесты - стоит ли?

Конечно, стоит!

Стоит ли тратить время на ожидание зелёного сигнала светофора?

Unit-тесты - стоит ли?

«Но на это же нужно

так много времени!»

Стоит ли тратить время на ожидание зелёного сигнала светофора?

Unit-тесты - стоит ли?

3 Стоит ли тратить время на ожидание зелёного сигнала светофора?

Google Testing Framework

он же GTest или googletest

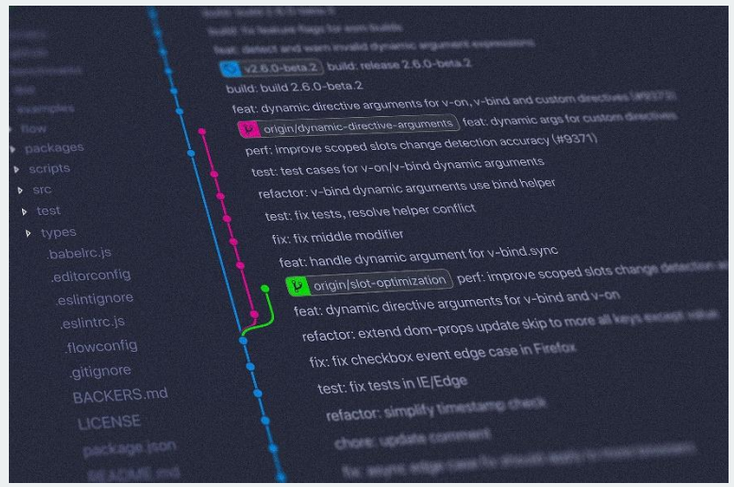
- open source
- бесплатный в том числе и для коммерческого применения
- кроссплатформенный
- компилируется на любом утюге с поддержкой C++11
- из коробки поддерживается CMake-ом
- xUnit архитектура (cases, fixtures, sutes, runners)
- широкие возможности

Подключаем к проекту

- ~~скопировать исходники~~
- git submodule
- CMake ExternalProject_Add
- заранее собрать и положить куда-нибудь в систему

Есть множество различных способов подключить одно и то же.

Подключаем к проекту



git submodule

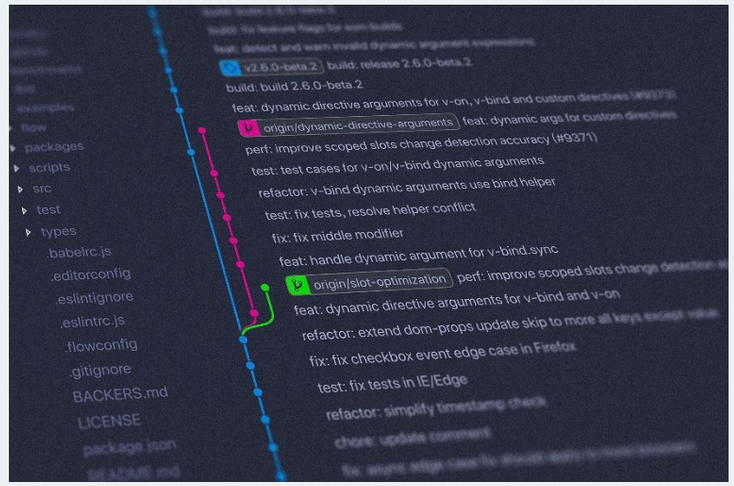
Добавляем к себе в репозиторий:

```
git submodule init

git submodule add [url_to_repo] [path_to_the_folder]

cd [path_to_the_folder]

git checkout [target_version]
```

При клонировании - выкачиваем зависимости:

```
git submodule update --init --recursive
```

Подключаем к проекту



git submodule

Дальше работаем с исходниками, как с обычным подпроектом:

```
# Build googletest
add_subdirectory(
    "${CMAKE_CURRENT_SOURCE_DIR}/googletest"
    "googletest"
)
```

Подключаем к проекту

CMake **ExternalProject_Add**

Один из рекомендованных авторами GTest способов.

```
include(ExternalProject)
ExternalProject_Add(googletest
  GIT_REPOSITORY    https://github.com/google/googletest.git
  GIT_TAG           master
  SOURCE_DIR        "${CMAKE_CURRENT_BINARY_DIR}/googletest-src"
  BINARY_DIR        "${CMAKE_CURRENT_BINARY_DIR}/googletest-build"
  CONFIGURE_COMMAND ""
  BUILD_COMMAND     ""
  INSTALL_COMMAND   ""
  TEST_COMMAND      ""
)
```

Подключаем к проекту

Предустановленный GTest

cmake .. -DGTEST_ROOT=[path_to_gtest]

```
# Lookup for pre-built gtest
find_package(GTest)

# Add googletest to the include directories for the test target
target_include_directories(test_list PRIVATE ${GTEST_INCLUDE_DIRS})

# Link test target against gtest libraries
target_link_libraries(test_list PRIVATE ${GTEST_BOTH_LIBRARIES})
```

Паттерн AAA

- **Arrange** - секция настройки теста:

  - создание переменных

  - формирование данные

  - любая иная подготовка
- **Act** - секция выполнения интересующего действия (вызов метода)
- **Assert** - секция проверки утверждения (валидация результатов)

GTest - утверждения

- **Фатальные:**

  ASSERT_EQ, ASSERT_NE, ASSERT_GT,

  ASSERT_GE, ASSERT_LT, ASSERT_LE …

- **Не фатальные**:

  EXPECT_EQ, EXPECT_NE, EXPECT_GT,

  EXPECT_GE, EXPECT_LT, EXPECT_LE …

GTest - утверждения

- **Строковые:**

  ASSERT_STREQ, ASSERT_STRCASEEQ,

  EXPECT_STREQ, EXPECT_STRCASEEQ

- **Проверка исключений**:

  ASSERT_THROW, ASSERT_NO_THROW

  EXPECT_THROW, EXPECT_NO_THROW

GTest - утверждения

- **Сложные строковые:**

  ASSERT_THAT
  EXPECT_THAT(bar_string, MatchesRegex("\\w*\\d+"));

- **compile-time**:

  ::testing::StaticAssertTypeEq<int, T>()

В общем, их очень и очень много.

Возможности приложения

Собранный unit-тест умеет:

- выводить список тестов
- запускать тесты выборочно
- повторять выполнение несколько раз
- запускать тесты в случайном порядке
- генерировать отчёт выполнения
- и многое другое… просто наберите --gtest_help

GMock

GMock - framework построения mock-классов или классов-заместителей.

- можем манипулировать условиями теста
- можем изменять поведение нижних уровней
- можем много всего ещё ☺

Спасибо
за внимание!

Заполните, пожалуйста
опрос о занятии.

Ответы на вопросы