Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

кафедра програмних засобів

Реферат

з дисципліни "Якість програмного забезпечення та тестування" на тему:

"Google Test"

Виконав:

студент групи КНТ-116 О.О. Палаш

2019

ЗМІСТ

ВСТУП

Тестування є важливою частиною процесу розробки програмного забезпечення. Модульне тестування в свою чергу є важливою частиною тестування програмного забезпечення.

Метою модульного тестування є ізолювати частини програми і окремо перевірити їх працездатність. Перевагами модульного тестування є:

- заохочення змін (рефакторингу);
- спрощення інтеграції;
- документування коду;
- відділення інтерфейсу від реалізації [1].

Більшість мов програмування не має вбудованих засобів модульного тестування, для тестування використовуються зовнішні інструменти (бібліотеки або фреймворки). Найбільш відомі наступні інструменти модульного тестування для мови C++:

- TypeMock Isolator++;
- CxxTest;
- CPPUnit;
- Boost Test;
- Google C++;
- Symbian;
- API Sanity Autotest;
- Qt Test framework.

1. GOOGLE C++ TESTING FRAMEWORK

1.1 Загальна інформація

Google C++ Testing Framework (Google Test) — бібліотека для модульного тестування на мові C++. Google Test побудована на методології тестування xUnit, коли окремі частини програми (класи, функції, модулі) перевіряються окремо один від одного. Бібліотека розповсюджується під ліцензією BSD з трьох пунктів [2].

Google Test може використовуватись для Windows та багатьох POSIX-сумісних операційних систем. Google Test дозволяє тестування коду, написаного мовою C, після мінімальних модифікацій коду [3].

Бібліотека розроблена з активним використанням тестування, коли при додаванні будь-яких змін необхідно написати набір тестів, які підтверджують їх коректність.

Бібліотека використовується в таких проектах, як: Chromium, LLVM, Protocol Buffers, OpenCV [4].

1.2 Особливості бібліотеки

Мінімальною одиницею тестування в Google Test є тест. Тести не потрібно реєструвати, кожен об'явлений тест буде запущений автоматично. Тести об'єднуються в групи, повна назва тесту формується з назви групи і власної назви тесту. Тести можуть використовувати тестові класи (test fixture), які дозволяють повторно використовувати одну і ту саму конфігурацію об'єктів для кількох різних тестів.

Бібліотека є безпечною для багатопотокового використання, але для одночасного використання тверджень в різних потоках необхідно самостійно розробити примітиви синхронізації.

У склад бібліотеки входить спеціальний скрипт, який упаковує її код в два файли: gtest-all.cc і gtest.h. Ці файли можуть бути включені в склад проекту без додаткових зусиль по збиранню бібліотеки [2].

2. ТЕСТУВАННЯ ЗА ДОПОМОГОЮ GOOGLE TEST

2.1 Загальні поняття

Ключовим поняттям в Google Test є поняття твердження (assert). Твердження уявляє собою вираз, результатом виконання якого може бути успіх (success), некритичний відказ (nonfatal failure) або критичний відказ (fatal failure). Критичний відказ викликає завершення тесту, в інших випадках тест продовжується. Сам тест є набором тверджень. Тести можуть бути згруповані в набори. Об'єднані набори тестів називають тестовою програмою (test program) [5].

2.2 Твердження

Твердження, які у випадку їх хибності викликають критичні відмови починаються з ASSERT_, некритичні — з EXPECT_. Твердження, які наявні в Google Test приведені в таблиці 2.1.

Таблиця 2.1 — Опис тверджень в Google Test

Назва твердження	Умова проходження тесту
ASSERT_TRUE(condition) EXPECT_TRUE(condition)	condition = true
ASSERT_FALSE(condition) EXPECT_FALSE(condition)	condition = false
ASSERT_EQ(val1, val2) EXPECT_EQ(val1, val2)	val1 == val2
ASSERT_NE(val1, val2) EXPECT_NE(val1, val2)	val1 != val2

Продовження таблиці 2.1

Назва твердження	Умова проходження тесту
ASSERT_LT(val1, val2) EXPECT_LT(val1, val2)	val1 < val2
ASSERT_LE(val1, val2) EXPECT_LE(val1, val2)	val1 <= val2
ASSERT_GT(val1, val2) EXPECT_GT(val1, val2)	val1 > val2
ASSERT_GE(val1, val2) EXPECT_GE(val1, val2)	val1 >= val2
ASSERT_STREQ(str1, str2) EXPECT_STREQ(str1, str2)	рядки str1 та str2 дорівнюють один одному
ASSERT_STRNE(str1, str2) EXPECT_STRNE(str1, str2)	рядки str1 та str2 не дорівнюють один одному
ASSERT_STRCASEEQ(str2, str2) EXPECT_STRCASEEQ(str2, str2)	рядки str1 та str2 дорівнюють один одному, реєстронезалежна
ASSERT_STRCASENE(str1, str2) EXPECT_STRCASENE(str1, str2)	рядки str1 та str2 не дорівнюють один одному, реєстронезалежна
ASSERT_THROW(statement, exception) EXPECT_THROW(statement, exception)	при виконанні statement виникає виняток типу exception
ASSERT_ANY_THROW(statement) EXPECT_ANY_THROW(statement)	при виконанні statement виникає виняток будь-якого типу
ASSERT_NO_THROW(statement) EXPECT_NO_THROW(statement)	при виконанні statement не виникає винятків
ASSERT_PREDn(pred, args) ASSERT_PRED_FORMATn(pred, args) EXPECT_PREDn(pred, args) EXPECT_PRED_FORMATn(pred, args)	pred(args) == true

Продовження таблиці 2.1

Назва твердження	Умова проходження тесту
ASSERT_FLOAT_EQ(expected, actual) ASSERT_DOUBLE_EQ(expected, actual) EXPECT_FLOAT_EQ(expected, actual) EXPECT_DOUBLE_EQ(expected, actual)	expected ≈ actual
ASSERT_NEAR(val1, val2, abs_error) EXPECT_NEAR(val1, val2, abs_error)	abs(val1 - val2) < abs_error

2.3 Тести

Для визначення тесту використовується макрос TEST. Він визначає функцію в якій можливо використовувати твердження. TEST приймає два параметри, які ідентифікують тест — назву тестового набору і назву тесту [5].

Приклад тесту:

```
TEST(test_case_name, test_name)
{
    ASSERT_EQ(1, 0);
}
```

Для тверджень можливо задавати власні коментарі:

```
ASSERT_EQ(1, 0) << "1 is not equal 0";
```

2.4 Тестові класи

Тестові класи або фіксації використовуються у випадку, коли об'єкти, які беруть участь у тестуванні, потребують складного налаштування для кожного тесту. Фіксація уявляє собою клас, який наслідує від ::testing::Test, в якому об'явлені усі необхідні для тестування об'єкти. В конструкторі класу або в методі SetUp() виконується їх налаштування, а в методі TearDown() виконується

звільнення ресурсів.

Тести в яких використовуються фіксації оголошуються за допомогою макросу TEST_F, в якості першого параметру якого вказується не назва набору тестів, а назва фіксації.

Для кожного тесту створюється новий об'єкт фіксації, налаштовується за допомогою методу SetUp(), виконуються тести, звільнюються ресурси за допомогою методу TearDown() і видаляється об'єкт фіксації.

2.5 Запуск тестів

Для запуску всіх тестів використовується функція RUN_ALL_TESTS(). Функцію можливо викликати тальки один раз. Бажано, щоб програма повертала результат роботи функції RUN_ALL_TESTS(), тому що деякі автоматичні засоби проектування визначають результат тестування по значенню, яке повертається [5].

Для налаштування параметрів запуску тестів (які тести та в якому порядку запускати, скільки разів запускати тести, в якому форматі видавати результати тестування та інше) використовується функція InitGoogleTest(argc, argv). Викликана перед RUN_ALL_TESTS(), ця функція дає можливість налаштовувати тестування за допомогою аргументів командного рядку.

ВИСНОВКИ

Проаналізувавши інформацію про бібліотеку Google Test, можна зробити наступні висновки:

- бібліотека підтримує більшість розповсюджених платформ;
- бібліотека має можливість повторного використання налаштувань об'єктів за допомогою тестових класів (фіксацій);
- бібліотека підтримує багатопотокове використання;
- бібліотека містить інструменти, які полегшують збирання проектів, які використовують Google Test;
- бібліотека дає можливість групування тестів;
- бібліотека надає твердження для перевірки різних типів даних;
- бібліотека містить критичні та некритичні варіанти тверджень;
- бібліотека дає можливість конфігурації запуску тестів.

Підсумувавши, можна зробити висновок, що бібліотека Google Test задовільняє більшість запитів, яки виникають при модульному тестуванні програм мовою C++.

ВИКОРИСТАНА ЛІТЕРАТУРА

1. Модульное тестирование [електронний ресурс]. URL: https://ru.wikipedia.org/wiki/Модульное_тестирование

2. Google C++ Testing Framework [електронний ресурс]. URL: https://ru.wikipedia.org/wiki/Google_C%2B%2B_Testing_Framework

3. Google Test [електронний ресурс]. URL: https://en.wikipedia.org/wiki/Google_Test

4. Google Test [електронний ресурс]. URL: https://github.com/google/googletest

5. Google testing framework (gtest) [електронний ресурс]. URL: https://habr.com/ru/post/119090/