Cheatography

C++ (Gtest) Cheat Sheet

by blakecromar via cheatography.com/218070/cs/48237/

::testing::Test — GTest Fixture

```
class my_fixture : public
::testing::Test {
protected:
       // shared member
variables
       int shared _va riable;

       void SetUp() override {
                sha red _va -
riable = 42; // runs before each
test
       }

       void TearDown() override
{
               // cleanup after
each test
       }
};
```

Base class for Google Test fixtures, providing automatic setup and teardown for each test.

Inherit from it to create a fixture that shares member variables and helper functions across multiple tests.

SetUp() runs before each test, TearDown() runs after each test — both are automatically called by Google Test.

::testing::UnitTest::current_test_info()

```
const auto* info =
::testing::UnitTest::GetInstance()-
>current_test_info();
info-> name(); // returns the test
name as a string
info-> tes t_s uit e_n ame(); //
returns the test suite name
info-> res ult(); // returns the
test result
```

Returns a pointer to information about the currently running test.

Contains details like the test name, test suite name, and whether the test has failed.

Commonly used in SetUp() to get the test name for creating unique temporary files or

EXPECT_LT

```
EXPECT_LT(actual_value,
upper_bound);
// common usage
EXPECT _LT (el aps ed_ time,
std::c hro no: :se con ds(5));
// verify operation was fast
EXPECT _LT (count, 10); //
verify count is under 10
```

A Google Test macro that checks if the first value is less than the second, marking the test as failed if it isn't.
Unlike ASSERT_LT, the test continues running after a failure so other checks can still be evaluated.
Commonly used to verify something happened quickly or within a time limit.

::testing::Test::SetUp()

```
void SetUp() override {
        tem p_d ire ctory =
create _te mp_ dir();
        pre pen d_t o_p ath -
(fa ke_ bin _dir);
        sav ed_path = get_cu -
rre nt_ path();
}
```

A virtual function inherited from ::testing::Test that runs automatically before each individual test.

Override it to initialise member variables, create temporary files, or prepare any state needed by the tests.

Always paired with TearDown() to clean up anything created here.

::testing::UnitTest::GetInstance()

```
const auto* unit_test =
::testing::UnitTest::GetInstance();
const auto* test_info = unit_t -
est ->c urr ent _te st_ info();
// or chained together
const auto* test_info = ::test -
ing ::U nit Tes t:: Get Ins tan -
ce( )-> cur ren t_t est _in fo();
```

Returns a pointer to the single global instance of the UnitTest object.

ASSERT_TRUE

```
ASSERT_TRUE(condition); // stops
test immediately if false
EXPECT _TR UE( con dit ion); //
marks failed but continues
running
```

A Google Test macro that checks if a condition is true and immediately aborts the test if it isn't.

Use when the rest of the test cannot meaningfully continue if this check fails.

Contrast with EXPECT_TRUE which marks the test as failed but allows it to keep running.

EXPECT_EQ

```
EXPECT_EQ(actual_value,
expected_value);
```

A Google Test macro that checks if two values are equal and marks the test as failed if they aren't.

Unlike ASSERT_EQ, the test continues running after a failure so other checks can still be evaluated.

Prints both the expected and actual values in the failure message making it easy to diagnose.

testing::Test::TearDown()

```
void TearDown() override {
        res tor e_p ath (sa -
ved _path); // restore original
PATH
        rem ove _te mp_ dir -
ect ory (tm pDir); // delete
temporary files
}
```

A virtual function inherited from ::testing::Test that runs automatically after each individual test.

Override it to clean up anything created in SetUp() — delete temporary files, restore environment variables, free resources.

Always runs even if the test fails, ensuring

directories.

EXPECT_TRUE

```
EXPECT_TRUE(condition); // marks failed but continues running
ASSERT _TR UE( con dit ion); // stops test immedi ately if false
```

A Google Test macro that checks if a condition is true and marks the test as failed if it isn't.

Unlike ASSERT_TRUE, the test continues running after a failure so other checks can still be evaluated.

Use when you want to collect multiple failures in one test run rather than stopping at the first one.

Uses the Singleton pattern — only one instance ever exists, and this method is how you access it.

From it you can retrieve information about the currently running test suite and individual tests.

cleanup always happens.

TEST_F

```
TEST_F(fixture_class_name, test_name)
{
       // test body has full access to fixture members
       // SetUp() has already run before this
       // TearDown() will run automa tically after this
}
```

Defines a test that uses a fixture class, giving it access to the fixture's member variables and helper functions.

Google Test automatically calls SetUp() before the test and TearDown() after, with a fresh fixture instance each time.

The second argument is the test name, which must be unique within the fixture.

EXPECT_FALSE

```
EXPECT_FALSE(condition); //
marks failed but continues
running
ASSERT _FA LSE (co ndi tion); //
stops test immedi ately if true
```

A Google Test macro that checks if a condition is false and marks the test as failed if it isn't.

Unlike ASSERT_FALSE, the test continues running after a failure so other checks can still be evaluated.

Use when you want to verify something does not exist or did not happen.

EXPECT_NE

```
EXPECT_NE(actual_value,
other_value); // fails if they
are equal
// common usage - string ::find
returns npos if not found
EXPECT _NE (my _st rin g.f ind -
("ta rge t"), std::s tri -
ng: :npos); // passes if " tar -
get " was found
EXPECT _NE (re sult, 0); //
passes if result is not zero
```

A Google Test macro that passes when two values are not equal, and marks the test as failed if they are equal.

Unlike ASSERT_NE, the test continues running after a failure so other checks can still be evaluated.

Commonly used to verify a search or find operation returned a valid result.