Unit Testing with C++ : The How and Why

by - Eric Gnoehelbecker

★ What is Unit Testing?

⇒ Unit testing means writing code that verifies individual parts or (unit) of application.

↳ {Smallest testable part of an application}.

⇒ Unit tests assess code in isolation.

⇒ In C++ this means writing tests for methods or functions.

↳ Tests only examine code within a single object, they don't rely on external resources.

★ Why Unit test with C++?

① Find Stupid Bugs Early

↳ Without unit tests, we don't catch these errors until we get to integration testing or worse.

② Avoid Regressions

↳ Not all bugs are stupid. Some bugs are quite intelligent. We call them regression.

Scanned with CamScanner

Regressions:
- Your system has been working in production for a long time
- But you need to add a new feature or address a deficiency.
- So you modify the code and roll out a new version.
- and something else breaks.

③ Better Design

→ Writing unit tests for code means writing code that can be broken down into discrete units.

★ How to Unit Test with C++

① Pick a Test Runner

→ Google Test is the most well-known cross-platform test runner for C++.
→ It comes with a mocking library.
→ It is open-source.

② Pick a Mocking FrameWork

Mocking
↳ Concealing an object that mimics the behavior of another object.

Example: If we are testing an object that interacts with a messaging API, we would mock the messaging connection, rather than write a test that requires connectivity with a messaging broker.

Scanned with CamScanner

⇒ The Google Test project bundles Google Mock with the test runner.

↳ You can use it to mock C++ classes and templates but the library has limitations that make working with Concrete and free functions difficult.

① Use Dependency Injection

⇒ You don't need a framework to use DI.

★ Write AAA Tests

⇒ Tests that are hard to decipher are almost as bad as no test at all.

⇒ Clean tests are easy to read and focus on a single piece of behavior.

- Arrange gathers the test requirements and prepares them for the test.
  ↳ It's where you Create the object to to test and set any preconditions it needs.
  ↳ Its where you Create your mocks.
- Act is the test operation itself.
- Assert verifies that the test succeeded.

Scanned with CamScanner