Unit testing: cppunit

Jacob de Fine Skibsted

Main purpose

- Test individual components of software.
- Automation.

Methodology

- Usually applied to object oriented programing.
- Uses common baselines known as test fixtures for individual components.
- Typically uses a framework although unit testing may be done without.

Advantages

- Designing software with unit testing in mind may yield better software - the programmer is forced to isolate components.
- Easy refactoring of code.
- The unit test is a form of documentation in itself.

Drawbacks

- Multiple code bases must be maintained.
- Time consuming.
- Programmers tend to believe everything works when the unit testing is ok.

Frameworks

- Boost test library.
- cppunit - no template support or exception support.
- NUnit - unit testing for .NET

cppunit

- C++ framework.
- Derived from jUnit.
- Centered around unit testing as a concept - C++ language specific constructs are not supported. (Use TUT instead).
- Object oriented.
- User defined tests are created by inheriting cppunit classes.
- Using assertion macros.

cppunit - concepts

- Test fixtures:
  - Basis for running test cases.
  - Establishes a fixed environment in which tests are run.
- Test callers:
  - Wraps the test cases in a fixture by registering each of the functions which performs the individual test cases.
- Test suites:
  - Added to a test fixture to group test cases together.
- Test runners:
  - Wraps the test suite in each test fixture.

```
#include <iostream>
#include <stack>
#include "TestCase.h"

class StackFixtureTest : public CppUnit::TestFixture {
private:
  std::stack<int> S;
public:
  void setUp() {}
  void tearDown() {}
  void TestStack()
  {
    CPPUNIT_ASSERT(S.empty() == true);
    S.push(100);
    CPPUNIT_ASSERT(((unsigned int) S.size()) == 1);
    CPPUNIT_ASSERT(S.top() == 100);
  }

  static CppUnit::Test *suite()
  {
    CppUnit::TestSuite *ts = new CppUnit::TestSuite( "StackTest" );
    ts->addTest( new CppUnit::TestCaller<StackFixtureTest>( "testStack",
                                                            &StackFixtureTest::TestStack ) );
    return ts;
  }
};
```

```
CppUnit::TextUi::TestRunner runner;
runner.addTest( StackFixtureTest::suite() );
runner.addTest( NextFixture::suite() );
runner.run();
```

Shortcomings

- Lack of exception handling.
- Lack of template handling.
- Does not take advantage of C++ specific constructs such as functors.