A quick guide to CppUnit

Fan Zhang

March 11, 2010

Contents

Abstract

This document gives a simple example of CppUnit. The content covers the download of CppUnit,an example of CppUnit, compilation and execution of the example. From the implementation of the example, it illustrates the basic idea and method of utilizing unit test with the tool of CppUnit.

Chapter 1

Introduction to Unit Test and CppUnit

1.1 What are unit test and Cppunit

An definition of unit test from wiki is as follows:

> In computer programming, unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure.
>
> CppUnit is a unit testing framework module for C++, described as a C++ port of JUnit.

1.2 Why do we need unit test

- easy to maintain the software, quick to shoot bugs
- isolate each part of the program and show that the individual parts are correct

- Facilitates change
- etc...

1.3 Where to download CppUnit

http://sourceforge.net/projects/cppunit/files/

Chapter 2

An Example of CppUnit

An class ComplexNumber will fist be defined and then its test class TestComplexNumber is implemented. The main function is generic under the following ways.

2.1 The class: ComplexNumber

ComplexNumber.cc

```
#include <iostream>
using namespace std;
class TestComplexNumber;
class ComplexNumber
{
  friend bool operator==(const ComplexNumber& a, const ComplexNumber& b)
  {
    return (a.realPart==b.realPart)&&(a.imagePart==b.imagePart);
  }
  friend  ComplexNumber operator+(const ComplexNumber& a, const ComplexNumber &b)
  {
    ComplexNumber c;
    c.realPart=a.realPart+b.realPart;
    c.imagePart=a.imagePart+b.imagePart;
    return c;
   }
   public:
   ComplexNumber(double r, double i):realPart(r),imagePart(i){}
   ComplexNumber():realPart(0.0),imagePart(0.0){}
```

```
   private:
   double realPart, imagePart;
   friend class TestComplexNumber;
};
```

The class *ComplexNumber* has two private data members *realPart* and *imagePart*. Two friend operators "==" and "+" is also defined.
The class *TestComplexNumber* is declared as the test class of *ComplexNumber*. It is also noticed that *TestComplexNumber* is declared as the friend class of *ComplexNumber* in order to visit its private members, e.g. realPart, imagePart...

2.2 The class: TestComplexNumber

TestComplexNumber.h

```
#include <cppunit/extensions/HelperMacros.h>
#include "ComplexNumber.h"
/*
 * A test case that is designed to produce
 * example errors and failures
 *
 */

class TestComplexNumber : public CPPUNIT_NS::TestFixture
{
  //establish the test suit of TestComplexNumber
  CPPUNIT_TEST_SUITE( TestComplexNumber);
  //add test method testInit
  CPPUNIT_TEST( testInit);
  //add test method testEquals
  CPPUNIT_TEST( testEquals );
  // add test method testAdd
  CPPUNIT_TEST( testAdd );
  // finish the process
  CPPUNIT_TEST_SUITE_END();

public:
  // overide setUp(), init data etc
  void setUp();
  //overide tearDown(), free allocated memory,etc
  void tearDown();
protected:
  //test method testAdd
```

```
  void testAdd();
  //test method testEquals
  void testEquals();
  //test method testInit
  void testInit();
private:
//three instances of CompleNumber for test
 ComplexNumber *a, *b, *c;
};
```

HelperMacros is utilized to help implement routin works to get the instance of test case.
TestComplexNumber.cc

```
#include "TestComplexNumber.h"

CPPUNIT_TEST_SUITE_REGISTRATION( TestComplexNumber );

void TestComplexNumber::setUp()
{
        a= new ComplexNumber(1.0,2.0);
        b= new ComplexNumber(1.0,2.0);
        c= new ComplexNumber(2.0,4.0);
}

void TestComplexNumber::tearDown()
{
        delete a;
        delete b;
        delete c;
}

void TestComplexNumber::testInit()
{
        CPPUNIT_ASSERT(a->realPart==1.0);
}
void TestComplexNumber::testAdd()
{
        CPPUNIT_ASSERT(*a+*b==*c);
}

void TestComplexNumber::testEquals()
{
        CPPUNIT_ASSERT(*a==*b);
        CPPUNIT_ASSERT(!(*a==*c));
```

```
}
```

Notice that *CPPUNIT_TEST_SUITE_REGISTRATION(TestComplexNumber)* is used to registrate the test factory so as that the main function will not need to include the fixture file (TestComplexNumber.h). CPPUNIT_ASSERT will assert the expression passed to it. If it's not turn, a error message will be got.

2.3 The main function

main.cc

```
#include <cppunit/extensions/TestFactoryRegistry.h>
#include <cppunit/ui/text/TestRunner.h>

int main( int argc, char **argv)
{
  CppUnit::TextUi::TestRunner runner;
  CppUnit::TestFactoryRegistry &registry
     = CppUnit::TestFactoryRegistry::getRegistry();
  runner.addTest( registry.makeTest() );
  runner.run();
  return 0;
}
```

This main is generic in the sense that it can be used for any test case.

Chapter 3

Compilation and Execution of the Example

3.1 Compilation

3.1.1 Include files

under the fold of CppUnitRoot/include

3.1.2 Library files

the lib file *libcppunit.a* is enough for this example. For further use, it is still under exploration.

3.1.3 Compile Command

g++ main.cc TestComplexNumber.cc -I include -L lib -lcppunit -o main
The include file folder and library file folder have been move the the current file path.

3.2 Execution Result

The result of the execution of main is as follows:

```
./main
...
```

```
OK (3 tests)
```

It shows all the three tests are passed If we change the data member of TestCompleNumber c be (3.0,4.0) instead of (2.0,4.0). Therefore, the testAdd function will not be passed during test. The following error messages will be returned:

```
./main
 ...F


!!!FAILURES!!!
Test Results:
Run:  3   Failures: 1   Errors: 0


1) test: TestComplexNumber::testAdd (F) line: 25 TestComplexNumber.cc
assertion failed
- Expression: *a+*b==*c
```

Chapter 4

Referrence

- http://cppunit.sourceforge.net/doc/lastest/cppunit_cookbook.html
- http://en.wikipedia.org/wiki/Unit_testing
- for more examples, please also refer to examples in the package of CppUnit.