Annexe1 – Les tests unitaires avec CppUnit

Sommaire :

I - Introduction

I.1. Objectifs

L'objectif de ce document est de montrer comment **tester unitairement** les **méthodes** d'une **classe** codée en **C++** en utilisant **CppUnit**.

Liens :
– http ://sourceforge.net/projects/cppunit/
– http ://cppunit.sourceforge.net/doc/cvs/cppunit_cookbook.html
– http ://matthieu-brucher.developpez.com/tutoriels/cpp/cppUnit/

I.2. xUnit

La meilleure façon d'exécuter des **tests unitaires** est d'utiliser une **procédure automatique**. C'est pour cette raison que nous allons utiliser un **outil** permettant de réaliser ces tests. En effet, les tests doivent pouvoir être exécutés sans intervention humaine et donner un résultat chiffré.

Le terme générique « **xUnit** » désigne un outil permettant de réaliser des **tests unitaires** dans un langage donné (dont l'initiale remplace « x » le plus souvent) : **CppUnit** pour le **C++**, **CUnit** pour le **C**, **PHPUnit** pour **PHP**, **JUnit** pour **Java**. **CppUnit** est une bibliothèque de tests unitaires permettant de vérifier le bon fonctionnement de son code **C++**.

I.3. Installation de CppUnit

Sous **Debian** ou **Ubuntu** il suffit d'exécuter la commande suivante :

apt-get install libcppunit-dev

Remarque : Il est possible d'utiliser **CppUnit** avec **QtCreator** à l'aide du plugin **qt-creator-cppunit-plugin** (**https://github.com/dibben/qt-creator-cppunit-plugin** et **https://www.codegardening.com/post/2016/2016-05-04-using-cppunit-with-qtcreator/**).

II - Mise en œuvre de CppUnit

II.1. Présentation

Il faut créer une classe de test, par exemple **TestUnitaireNmea** ainsi qu'un **main()**. Dernière chose, le **Makefile** aussi sera à modifier, pour compiler le programme test. Tous ces fichiers seront impérativement positionnés dans un dossier nommé **tests** afin de ne pas modifier les sources du programme à tester.

II.2. Programme principal

L'avantage de cette solution, c'est que le fichier **.cpp** qui contient le **main()**, ne change jamais. Le voici (fichier **main.cpp**) :

```
#include <cppunit/BriefTestProgressListener.h>
#include <cppunit/CompilerOutputter.h>
#include <cppunit/extensions/TestFactoryRegistry.h>
#include <cppunit/TestResult.h>
#include <cppunit/TestResultCollector.h>
#include <cppunit/TestRunner.h>

int main( int argc, char* argv[] )
{
// Create the event manager and test controller
CPPUNIT_NS::TestResult controller;
// Add a listener that colllects test result
CPPUNIT_NS::TestResultCollector result;
controller.addListener( &result );
// Add a listener that print dots as test run.
CPPUNIT_NS::BriefTestProgressListener progress;
controller.addListener( &progress );
// Add the top suite to the test runner
CPPUNIT_NS::TestRunner runner;
runner.addTest( CPPUNIT_NS::TestFactoryRegistry::getRegistry().makeTest() );
runner.run( controller );
// Print test in a compiler compatible format.
CPPUNIT_NS::CompilerOutputter outputter( &result, CPPUNIT_NS::stdCOut() );
outputter.write();
// On retourne le code d'erreur 1 si un test a échoué
return result.wasSuccessful() ? 0 : 1;
}
```

II.3. Classe de test

Maintenant on créé une classe de test **TestUnitaireNmea** avec le framework **CppUnit**. Le principe est simple :

- On déclare la classe de test en héritant de la classe CPPUNIT_NS::TestFixture ;
- On crée une suite de tests unitaires en utilisant les macros fournies par CppUnit (CPPUNIT_TEST_SUITE et CPPUNIT_TEST_SUITE_END) ;
- On ajoute les méthodes de tests dans la suite de tests unitaires créée avec la macro CPPUNIT_TEST ;
- On déclare les méthodes de tests.

La déclaration de la classe **TestUnitaireNmea** sera la suivante (fichier **TestUnitaireNmea.h**) :

```
#ifndef _TESTUNITAIRENMEA_H
```

```
#define _TESTUNITAIRENMEA_H
#include <cppunit/extensions/HelperMacros.h>
#define TAILLE_MAX_TRAME          82
class Nmea; // La classe à tester
class TestUnitaireNmea : public CPPUNIT_NS::TestFixture
{
// On crée une suite de tests unitaires pour la classe
CPPUNIT_TEST_SUITE(TestUnitaireNmea);
CPPUNIT_TEST(testTrameValide);
CPPUNIT_TEST(testSommeDeControle );
CPPUNIT_TEST_SUITE_END();
private:
Nmea *nmea; // un pointeur sur une instance de la classe à tester
public:
TestUnitaireNmea();
virtual ~TestUnitaireNmea();
// Call before tests
void setUp();
// Call after tests
void tearDown();
// Liste des tests
void testTrameValide();
void testSommeDeControle();
};
#endif
```

La définition de la classe **TestUnitaireNmea** sera la suivante (fichier **TestUnitaireNmea.cpp**) :

```
#include <cppunit/config/SourcePrefix.h>
#include "TestUnitaireNmea.h"
#include "../nmea.h" // Classe à tester
// Enregistrement des différents cas de tests
CPPUNIT_TEST_SUITE_REGISTRATION( TestUnitaireNmea );
TestUnitaireNmea::TestUnitaireNmea()
{
}
TestUnitaireNmea::~TestUnitaireNmea()
{
}
void TestUnitaireNmea::setUp()
{
// Initialisation pour les tests
nmea = new Nmea();
}
void TestUnitaireNmea::tearDown()
{
// fin du test
delete nmea;
}
void TestUnitaireNmea::testTrameValide()
{
char  trameNmea0[TAILLE_MAX_TRAME] = "$GPGGA,,,,,,0,00,99.99,,,,,,*48\r\n";
CPPUNIT_ASSERT(nmea->trameValide(trameNmea0) != 0); // Trame non valide
char                                    trameNmea1[TAILLE_MAX_TRAME]                    =
"$GPGGA,164212,4341.4519,N,00016.4724,W,1,04,5.7,167.0,M,50.3,M,,*5B\r\n";
CPPUNIT_ASSERT(nmea->trameValide(trameNmea1) == 0); // Trame valide
char                                    trameNmea2[TAILLE_MAX_TRAME]                    =
```

```
"$GPRMC,164213,A,4341.4518,N,00016.4702,W,000.0,000.0,260313,002.1,W*7F\r\n";
CPPUNIT_ASSERT(nmea->trameValide(trameNmea2) == 0); // Trame valide
}
void TestUnitaireNmea::testSommeDeControle()
{
char  trameNmea0[TAILLE_MAX_TRAME] = "$GPGGA,,,,,,0,00,99.99,,,,,,*48\r\n";
CPPUNIT_ASSERT(nmea->sommeDeControle(trameNmea0) == 0x48); // valide
char                                    trameNmea1[TAILLE_MAX_TRAME]                    =
"$GPGGA,164212,4341.4519,N,00016.4724,W,1,04,5.7,167.0,M,50.3,M,,*5B\r\n";
CPPUNIT_ASSERT(nmea->sommeDeControle(trameNmea1) == 0x5B); // valide
char                                    trameNmea2[TAILLE_MAX_TRAME]                    =
"$GPRMC,164213,A,4341.4518,N,00016.4702,W,000.0,000.0,260313,002.1,W*7F\r\n";
CPPUNIT_ASSERT(nmea->sommeDeControle(trameNmea2) == 0x7F); // valide
}
```

II.4. Makefile du test

```
all: tests
tests: main.o ../nmea.o TestUnitaireNmea.o
	g++ -o tests main.o ../nmea.o TestUnitaireNmea.o -lcppunit -lpthread -lrt
# main.o
main.o: main.cpp
	g++ -c -o main.o main.cpp -Wall
# nmea.o
../nmea.o: ../nmea.cpp ../nmea.h
	g++ -c -o ../nmea.o ../nmea.cpp -Wall
# TestUnitaireNmea.o
TestUnitaireNmea.o: TestUnitaireNmea.cpp TestUnitaireNmea.h
	g++ -c -o TestUnitaireNmea.o TestUnitaireNmea.cpp -Wall
clean:
	rm -rf *.o tests
```

II.5. Exécution du test

```
$ cd tests
$ make all
g++ -c -o main.o main.cpp -Wall
g++ -c -o ../nmea.o ../nmea.cpp -Wall
g++ -c -o TestUnitaireNmea.o TestUnitaireNmea.cpp -Wall
g++ -o tests main.o ../nmea.o TestUnitaireNmea.o -lcppunit -lpthread -lrt
$ ./tests
TestUnitaireNmea::testTrameValide : OK
TestUnitaireNmea::testSommeDeControle : OK
OK (2)
```

On constate que le jeu de tests est validé.