

ALGORITMOS Y ESTRUCTURAS DE DATOS I

Taller de Testing

Facultad de Ciencias Exactas y Naturales

2019

20 de Mayo, 2019

GOOGLE TEST

¿QUÉ ES GOOGLE TEST?

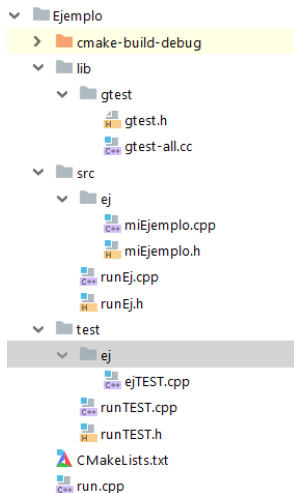
- Google Test es una biblioteca que provee funcionalidades para realizar pruebas unitarias (de unidad) automatizadas.
- Una prueba de unidad es cuando se realiza una prueba específica a cada uno de los componentes individuales de un software.
 - En nuestro caso, la unidad a probar será "La Función".
- En cada prueba específica(Test) se identifica las entradas requeridas y los resultados esperados.
- En GoogleTest, las pruebas utilizan ASSERT(afirmaciones) para verificar el comportamiento del código probado.

ASSERT...

- Los ASSERT de googletest son macros que se asemejan a lo que conocemos como funciones.
- Cuando falla un ASSERT, googletest imprime el archivo de origen de la aserción y la ubicación del número de línea, junto con un mensaje de error.
- Hay un variado conjunto de macros, las más usadas:
 - ASSERT_TRUE(Condición).
 - ASSERT_FALSE(Condición).
 - ASSERT_EQ(esperado, actual)
 - ASSERT_NE(val1, val2)
 - ASSERT_DOUBLE_EQ(esperado, actual)
 - ASSERT_NEAR(val1, val2, abs_err)
- Para ampliar:
 - [Google Test Quick Reference](#):
 - [Google Test en GitHub](#):

EJEMPLO DE PROYECTO

• Ver EJEMPLO de código



EJECUTAR TEST

- La función `::testing::InitGoogleTest()` inicializa googleTest. Se debe invocar antes de ejecutar los test.
- Se puede ejecutar con `RUN_ALL_TESTS()`. Retorna 0 si todas las pruebas son exitosas, o 1 caso contrario.

```
#include <iostream>
#include "runTEST.h"
#include "../lib/gtest/gtest.h"
int runTest(int argc, char **argv){
    ::testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

```
#include "../src/ej/miEjemplo.h"
#include "../lib/gtest/gtest.h"

TEST(ejTEST1, PotenciaNumEntPosOK_Test ) {
    ASSERT_EQ(8, pot(2,3));
    ASSERT_EQ(9, pot(3,2));
    ASSERT_EQ(27, pot(3,3));
}
```

- Usar la `TEST()` macro para definir y nombrar una función de prueba.

EJEMPLO DE PROYECTO

- Ver EJEMPLO de código. Bajarlo y ejecutarlo.

COBERTURA DE CÓDIGO

COBERTURA DE CÓDIGO

La cobertura de código es una medida de la cantidad de líneas, declaraciones o bloques del código que se prueban utilizando un conjunto de pruebas automatizadas.

COBERTURA DE CÓDIGO - TESTS DE CAJA BLANCA - EJEMPLO A ANALIZAR

Sofía está jugando un juego. Tiene una bolsa con bolitas y mete la mano para sacar un puñado. Gana puntos según la cantidad de bolitas con las siguientes reglas:

- Si la cantidad de bolitas es menor que 10, gana dos puntos por cada bolita que sacó. Si no, un punto por cada una.
- Además, si la cantidad de bolitas que sacó es múltiplo de 3, gana 10 puntos. Si no, pierde 10 puntos.

Cuántos puntos ganó Sofía?

COBERTURA DE CÓDIGO - TESTS DE CAJA BLANCA - EJEMPLO A ANALIZAR

```
1  int puntaje(int b) {  
2      int res;  
3      if (b < 10) {  
4          res = 2 * b;  
5      } else {  
6          res = b;  
7      }  
8      if (b % 3 == 0) {  
9          res = res + 10;  
10     } else {  
11         res = res - 10;  
12     }  
13     return res;  
14 }
```

LCOV

- <http://ltp.sourceforge.net/coverage/lcov.php>:
- Es una herramienta que se utiliza para medir la cobertura de código de un proyecto de software.
- LCOV es una interfaz gráfica para la herramienta de prueba de cobertura de GCC gcov .
- Recopila datos gcov para múltiples archivos de origen y crea páginas HTML que contienen el código fuente marcado con información de cobertura.

INSTALAR - EJECUTAR LCOV

- Instalar lcov
 - [Linux Test Project](#):
 - [lcov en github](#):
- Luego de bajarlo e instalarlo, dirigirse a la carpeta:
 - `cd /lcov/bin`

EJECUTAR LCOV - PRIMER PASO: GENERAR COVERAGE.INFO

`./lcov -capture -directory «pdir» -output-file «odir»/coverage.info`

- lcov debe ser ejecutado desde línea de comando.
- Es su ejecución crea en el «odir» el archivo 'coverage.info'.
- Previamente deben estar generados los .gcno y .gcda y estos se encuentren en el directorio «pdir».
- El directorio relativo «pdir» será «miProyecto»/cmake-build-debug/CMakeFiles/«miProyecto».dir
- El directorio «odir» podrá ser el que se desee.

EJECUTAR LCOV - SEGUNDO PASO: GENERAR INFORME HTML

`./genhtml «odir»/coverage.info --output-directory «odir»/cobertura`





- «odir» directorio donde se encuentra coverage.info, creado en primer paso.
- /cobertura será el directorio donde lcov dejará el html generado.

EJEMPLO DE HTML GENERADO

- Informe salido del ejemplo de código

LCOV - informe de cobertura de código

Vista actual: nivel superior		Golpear		Total	Cobertura	
Prueba: cobertura.info		Líneas:	80	125	64.0%	
Fecha: 2019-05-19 21:36:35		Funciones:	60	97	61.9%	

Directorio	Cobertura de línea ↕			Funciones ↕	
/c/PROGRA-2/MINGW~/i/mingw32/lib/gcc/i686-w64-mingw32/7.1.0/include/c++		100.0%	3/3	50.0%	1/2
/c/PROGRA-2/MINGW~/i/mingw32/lib/gcc/i686-w64-mingw32/7.1.0/include/c++/bits		100.0%	2/2	100.0%	1/1
Ejemplo_1		100.0%	4/4	100.0%	3/3
Ejemplo_1 / lib / gtest		49.4%	43/87	45.5%	30/66
Ejemplo1 / src		100.0%	6/6	100.0%	3/3
Ejemplo1 / src / ej		83.3%	5/6	100.0%	1/1
Ejemplo1 / prueba		100.0%	4/4	100.0%	3/3
Ejemplo1 / test / ej		100.0%	13/13	100.0%	18/18

Generado por: [LCOV versión 1.13-16-ge675080](#)