Framework Unit Test

1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 Tester Namespace Reference

Este namespace es el encargado de la definición de los test suites, los test y la propia llamada a estos últimos.

4.1.1 Detailed Description

Este namespace es el encargado de la definición de los test suites, los test y la propia llamada a estos últimos.

Chapter 5

Class Documentation

5.1 Calculadora Class Reference

`#include <calculadora.hpp>`

Public Member Functions

- Calculadora ()
  Este metodo crea y inicia la calculadora.
- ~Calculadora ()
  Este metodo es el destructor de la calculadora.
- bool llegeixNumero (int iter)
  Este metodo lee un valor de operando por teclado.
- bool llegeixOperacio ()
  Este metodo lee la operación deseada por teclado.
- float getNum1 ()
  Este metodo devuelve el primer operando de dos valores.
- float getNum2 ()
  Este metodo devuelve el segundo operando de dos valores.
- char getOp ()
  Este metodo devuelve el operador.
- void setNum1 (float num)
  Este metodo setea el primer operando.
- void setNum2 (float num)
  Este metodo setea el segundo operando.
- void setOp (char op)
  Este metodo setea el operador.
- float calcula ()
  Este metodo realiza la operación introducida.

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Calculadora()

```
Calculadora::Calculadora ( )
```

Este metodo crea y inicia la calculadora.

5.1.1.2 ~Calculadora()

```
Calculadora::~Calculadora ( )
```

Este metodo es el destructor de la calculadora.

5.1.2 Member Function Documentation

5.1.2.1 calcula()

```
float Calculadora::calcula ( )
```

Este metodo realiza la operación introducida.

Returns

Devuelve el valor de la operación ejecutada

5.1.2.2 getNum1()

```
float Calculadora::getNum1 ( )
```

Este metodo devuelve el primer operando de dos valores.

Returns

operando numero 1

5.1.2.3 getNum2()

```
float Calculadora::getNum2 ( )
```

Este metodo devuelve el segundo operando de dos valores.

Returns

operando numero 2

5.1.2.4 getOp()

```
char Calculadora::getOp ( )
```

Este metodo devuelve el operador.

Returns

operador

5.1.2.5 llegeixNumero()

```
bool Calculadora::llegeixNumero (
            int iter )
```

Este metodo lee un valor de operando por teclado.

Parameters

iter	define que operando se le, iter igual a 1 lee operando 1, si no, lee operando 2

Returns

Devuelve true cuando ha leido el valor con exito

5.1.2.6 llegeixOperacio()

```
bool Calculadora::llegeixOperacio ( )
```

Este metodo lee la operación deseada por teclado.

Returns

Devuelve true si ha leido una operacion valida ('+','∗','/','-',''), false si no es valido

5.1.2.7 setNum1()

```
void Calculadora::setNum1 (
            float num )
```

Este metodo setea el primer operando.

Parameters

num	Valor del operando 1

5.1.2.8 setNum2()

```
void Calculadora::setNum2 (
            float num )
```

Este metodo setea el segundo operando.

Parameters

num	Valor del operando 2

5.1.2.9 setOp()

```
void Calculadora::setOp (
            char op )
```

Este metodo setea el operador.

Parameters

op	Valor del operando

The documentation for this class was generated from the following files:

- AppEjemplo/calculadora.hpp
- AppEjemplo/calculadora.cpp

Chapter 6

File Documentation

6.1 AppEjemplo/calculadora.cpp File Reference

Implementacion de los metodos de la clase calculadora.

```
#include "calculadora.hpp"
#include <iostream>
#include <string>
```

6.1.1 Detailed Description

Implementacion de los metodos de la clase calculadora.

Author

Francisco Magdaleno francisco.magdalenog@gmail.com

6.2 AppEjemplo/calculadora.hpp File Reference

Definición de la clase calculadora.

Classes

- class Calculadora

6.2.1 Detailed Description

Definición de la clase calculadora.

Author

Francisco Magdaleno francisco.magdalenog@gmail.com

6.3 AppEjemplo/main.cpp File Reference

Archivo con la funcion main de la calculadora.

```
#include "calculadora.hpp"
#include <iostream>
#include <iomanip>
```

Functions

- int main ()

6.3.1 Detailed Description

Archivo con la funcion main de la calculadora.

Author

Francisco Magdaleno francisco.magdalenog@gmail.com

6.3.2 Function Documentation

6.3.2.1 main()

```
int main ( )
```

6.4 Test/AllTests.cpp File Reference

Punto de entrada del framework.

```
#include "gtest/gtest.h"
#include <cppunit/TestCase.h>
#include <cppunit/TestFixture.h>
#include <cppunit/TestRunner.h>
#include <cppunit/TestResult.h>
#include <cppunit/TestResultCollector.h>
#include <cppunit/ui/text/TextTestRunner.h>
#include <cppunit/extensions/HelperMacros.h>
#include <cppunit/extensions/TestFactoryRegistry.h>
#include <cppunit/CompilerOutputter.h>
#include <cppunit/BriefTestProgressListener.h>
#include "test.h"
```

Functions

- int main (int argc, char **argv)

6.4.1 Detailed Description

Punto de entrada del framework.

Author

Francisco Magdaleno Garrido francisco.magdalenog@gmail.com

6.4.2 Function Documentation

6.4.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

6.5 Test/test.h File Reference

Macros para sobrecargar los metodos de las diferentes plataformas.
Macros [1/2] corresponden a GOOGLE TEST
Macros [2/2] corresponden a CPPUNIT

Macros

- #define CHECK_EQ_INT(X, Y) ASSERT_EQ(X,Y)

  Comprueba que los valores enteros X e Y son iguales.
- #define CHECK_EQ_FLOAT(X, Y) ASSERT_FLOAT_EQ(X,Y)

  Comprueba que dos valores flotantes son iguales.
- #define CHECK_NE(X, Y) ASSERT_NE(X,Y)

  Comprueba que los valores X e Y no son iguales.
- #define CHECK_LT(X, Y) ASSERT_LT(X,Y)

  Comprueba que el valor X es menor que el valor Y.
- #define CHECK_LE(X, Y) ASSERT_LE(X,Y)

  Comprueba que el valor X es menor o igual al valor Y.
- #define CHECK_GT(X, Y) ASSERT_GT(X,Y)

  Comprueba que el valor X es mayor al valor Y.
- #define CHECK_GE(X, Y) ASSERT_GE(X,Y)

  Comprueba que el valor X es mayor o igual al valor Y.
- #define CHECK_STREQ(X, Y) ASSERT_STREQ(X,Y)

  Comprueba que las cadenas X e Y son iguales.

- #define CHECK_STRNE(X, Y) ASSERT_STRNE(X,Y)

  Comprueba que las cadenas X e Y son diferentes.
- #define CHECK_BOOLEAN_TRUE(X) ASSERT_TRUE(X)

  Comprueba que el booleano o expresión X es verdadera.
- #define CHECK_BOOLEAN_FALSE(X) ASSERT_FALSE(X)

  Comprueba que el booleano o expresión X es falsa.
- #define CHECK_NULL(X) ASSERT_TRUE(X == nullptr)

  Comprueba que el puntero X es nulo.
- #define TEST_SUITE_BEGIN(X) class X : public INHERITANCE{};

  Crea el test suite.
- #define TEST_METHOD(X, Y) TEST_F(X,Y){Y();}

  Añade y ejecuta el test Y.
- #define TEST_SUITE_END()

  No ejecuta codigo, da logica e intuicion a la escritura de tests.
- #define CHECK_EQ_INT(X, Y) CPPUNIT_ASSERT_EQUAL((float) X,(float) Y)

  Comprueba que los valores enteros X e Y son iguales.
- #define CHECK_EQ_FLOAT(X, Y) CPPUNIT_ASSERT_DOUBLES_EQUAL(X,Y,DELTA)

  Comprueba que dos valores flotantes son iguales.
- #define CHECK_NE(X, Y) CPPUNIT_ASSERT(X!=Y)

  Comprueba que los valores X e Y no son iguales.
- #define CHECK_LT(X, Y) CPPUNIT_ASSERT(X<Y)

  Comprueba que el valor X es menor que el valor Y.
- #define CHECK_LE(X, Y) CPPUNIT_ASSERT(X<=Y)

  Comprueba que el valor X es menor o igual al valor Y.
- #define CHECK_GT(X, Y) CPPUNIT_ASSERT(X>Y)

  Comprueba que el valor X es mayor al valor Y.
- #define CHECK_GE(X, Y) CPPUNIT_ASSERT(X>=Y)

  Comprueba que el valor X es mayor o igual al valor Y.
- #define CHECK_STREQ(X, Y) CPPUNIT_ASSERT(X==Y)

  Comprueba que las cadenas X e Y son iguales.
- #define CHECK_STRNE(X, Y) CPPUNIT_ASSERT(X!=Y)

  Comprueba que las cadenas X e Y son diferentes.
- #define CHECK_BOOLEAN_TRUE(X) CPPUNIT_ASSERT(X)

  Comprueba que el booleano o expresión X es verdadera.
- #define CHECK_BOOLEAN_FALSE(X) CPPUNIT_ASSERT(!X)

  Comprueba que el booleano o expresión X es falsa.
- #define CHECK_NULL(X) CPPUNIT_ASSERT(X==NULL)

  Comprueba que el puntero X es nulo.
- #define TEST_SUITE_BEGIN(X)

  Crea el test suite.
- #define TEST_METHOD(X, Y)

  Añade y ejecuta el test Y.
- #define TEST_SUITE_END()

  No ejecuta codigo, da logica e intuicion a la escritura de tests.

6.5.1 Detailed Description

Macros para sobrecargar los metodos de las diferentes plataformas.
Macros [1/2] corresponden a GOOGLE TEST
Macros [2/2] corresponden a CPPUNIT

Author

Francisco Magdaleno Garrido francisco.magdalenog@gmail.com

6.5.2 Macro Definition Documentation

6.5.2.1 CHECK_BOOLEAN_FALSE [1/2]

```
#define CHECK_BOOLEAN_FALSE(
              X ) ASSERT_FALSE(X)
```

Comprueba que el booleano o expresión X es falsa.

Parameters

X	variable booleana a comprobar

6.5.2.2 CHECK_BOOLEAN_FALSE [2/2]

```
#define CHECK_BOOLEAN_FALSE(
              X ) CPPUNIT_ASSERT(!X)
```

Comprueba que el booleano o expresión X es falsa.

Parameters

X	variable booleana a comprobar

6.5.2.3 CHECK_BOOLEAN_TRUE [1/2]

```
#define CHECK_BOOLEAN_TRUE(
              X ) ASSERT_TRUE(X)
```

Comprueba que el booleano o expresión X es verdadera.

Parameters

X	variable booleana a comprobar

6.5.2.4 CHECK_BOOLEAN_TRUE [2/2]

```
#define CHECK_BOOLEAN_TRUE(
              X ) CPPUNIT_ASSERT(X)
```

Comprueba que el booleano o expresión X es verdadera.

Parameters

X	variable booleana a comprobar

6.5.2.5 CHECK_EQ_FLOAT [1/2]

```
#define CHECK_EQ_FLOAT(
              X,
              Y ) ASSERT_FLOAT_EQ(X,Y)
```

Comprueba que dos valores flotantes son iguales.

Parameters

X	valor esperado
Y	valor actual

6.5.2.6 CHECK_EQ_FLOAT [2/2]

```
#define CHECK_EQ_FLOAT(
              X,
              Y ) CPPUNIT_ASSERT_DOUBLES_EQUAL(X,Y,DELTA)
```

Comprueba que dos valores flotantes son iguales.

Parameters

X	valor esperado
Y	valor actual

6.5.2.7 CHECK_EQ_INT [1/2]

```
#define CHECK_EQ_INT(
              X,
              Y ) ASSERT_EQ(X,Y)
```

Comprueba que los valores enteros X e Y son iguales.

Parameters

X	valor esperado
Y	valor actual

6.5.2.8 CHECK_EQ_INT [2/2]

```
#define CHECK_EQ_INT(
              X,
              Y ) CPPUNIT_ASSERT_EQUAL((float) X,(float) Y)
```

Comprueba que los valores enteros X e Y son iguales.

Parameters

X	valor esperado
Y	valor actual

6.5.2.9 CHECK_GE [1/2]

```
#define CHECK_GE(
              X,
              Y ) ASSERT_GE(X,Y)
```

Comprueba que el valor X es mayor o igual al valor Y.

Parameters

X	valor esperado
Y	valor actual

6.5.2.10 CHECK_GE [2/2]

```
#define CHECK_GE(
              X,
              Y ) CPPUNIT_ASSERT(X>=Y)
```

Comprueba que el valor X es mayor o igual al valor Y.

Parameters

X	valor esperado
Y	valor actual

6.5.2.11 CHECK_GT [1/2]

```
#define CHECK_GT(
              X,
              Y ) ASSERT_GT(X,Y)
```

Comprueba que el valor X es mayor al valor Y.

Parameters

X	valor esperado
Y	valor actual

6.5.2.12 CHECK_GT [2/2]

```
#define CHECK_GT(
              X,
              Y ) CPPUNIT_ASSERT(X>Y)
```

Comprueba que el valor X es mayor al valor Y.

Parameters

X	valor esperado
Y	valor actual

6.5.2.13 CHECK_LE [1/2]

```
#define CHECK_LE(
              X,
              Y ) ASSERT_LE(X,Y)
```

Comprueba que el valor X es menor o igual al valor Y.

Parameters

X	valor esperado
Y	valor actual

6.5.2.14 CHECK_LE [2/2]

```
#define CHECK_LE(
              X,
              Y ) CPPUNIT_ASSERT(X<=Y)
```

Comprueba que el valor X es menor o igual al valor Y.

Parameters

X	valor esperado
Y	valor actual

6.5.2.15 CHECK_LT [1/2]

```
#define CHECK_LT(
              X,
              Y ) ASSERT_LT(X,Y)
```

Comprueba que el valor X es menor que el valor Y.

Parameters

X	valor esperado
Y	valor actual

6.5.2.16 CHECK_LT [2/2]

```
#define CHECK_LT(
              X,
              Y ) CPPUNIT_ASSERT(X<Y)
```

Comprueba que el valor X es menor que el valor Y.

Parameters

X	valor esperado
Y	valor actual

6.5.2.17 CHECK_NE [1/2]

```
#define CHECK_NE(
```

```
X,
Y ) ASSERT_NE(X,Y)
```

Comprueba que los valores X e Y no son iguales.

Parameters

X	valor esperado
Y	valor actual

6.5.2.18 CHECK_NE [2/2]

```
#define CHECK_NE(
              X,
              Y ) CPPUNIT_ASSERT(X!=Y)
```

Comprueba que los valores X e Y no son iguales.

Parameters

X	valor esperado
Y	valor actual

6.5.2.19 CHECK_NULL [1/2]

```
#define CHECK_NULL(
              X ) ASSERT_TRUE(X == nullptr)
```

Comprueba que el puntero X es nulo.

Parameters

X	puntero con valor nulo

6.5.2.20 CHECK_NULL [2/2]

```
#define CHECK_NULL(
              X ) CPPUNIT_ASSERT(X==NULL)
```

Comprueba que el puntero X es nulo.

Parameters

X	puntero con valor nulo

6.5.2.21 CHECK_STREQ [1/2]

```
#define CHECK_STREQ(
              X,
              Y ) ASSERT_STREQ(X,Y)
```

Comprueba que las cadenas X e Y son iguales.

Parameters

X	string esperado
Y	string actual

6.5.2.22 CHECK_STREQ [2/2]

```
#define CHECK_STREQ(
              X,
              Y ) CPPUNIT_ASSERT(X==Y)
```

Comprueba que las cadenas X e Y son iguales.

Parameters

X	string esperado
Y	string actual

6.5.2.23 CHECK_STRNE [1/2]

```
#define CHECK_STRNE(
              X,
              Y ) ASSERT_STRNE(X,Y)
```

Comprueba que las cadenas X e Y son diferentes.

Parameters

X	string esperado
Y	string actual

6.5.2.24 CHECK_STRNE [2/2]

```
#define CHECK_STRNE(
              X,
              Y ) CPPUNIT_ASSERT(X!=Y)
```

Comprueba que las cadenas X e Y son diferentes.

Parameters

X	string esperado
Y	string actual

6.5.2.25 TEST_METHOD [1/2]

```
#define TEST_METHOD(
              X,
              Y ) TEST_F(X,Y){Y();}
```

Añade y ejecuta el test Y.

Parameters

X	nombre de la clase test
Y	nombre de metodo test

6.5.2.26 TEST_METHOD [2/2]

```
#define TEST_METHOD(
              X,
              Y )
```

Value:

```
class X##Y : public X{ \
             public: \
                CPPUNIT_TEST_SUB_SUITE(X##Y,X);\
                CPPUNIT_TEST(run##Y); \
                CPPUNIT_TEST_SUITE_END(); \
                void run##Y(){Y();};\
             }; \
             CPPUNIT_TEST_SUITE_REGISTRATION(X##Y);
```

Añade y ejecuta el test Y.

Parameters

X	nombre de la clase test
Y	nombre de metodo test

6.5.2.27 TEST_SUITE_BEGIN [1/2]

```
#define TEST_SUITE_BEGIN(
              X ) class X : public INHERITANCE{};
```

Crea el test suite.

Parameters

X	nombre del test suite

6.5.2.28 TEST_SUITE_BEGIN [2/2]

```
#define TEST_SUITE_BEGIN(
              X )
```

Value:

```
class X : public INHERITANCE{ \
               CPPUNIT_TEST_SUITE(X); \
               CPPUNIT_TEST_SUITE_END(); \
               };\
               CPPUNIT_TEST_SUITE_REGISTRATION(X);
```

Crea el test suite.

Parameters

X	nombre del test suite

6.5.2.29 TEST_SUITE_END [1/2]

```
#define TEST_SUITE_END( )
```

No ejecuta codigo, da logica e intuicion a la escritura de tests.

6.5.2.30 TEST_SUITE_END [2/2]

```
#define TEST_SUITE_END( )
```

No ejecuta codigo, da logica e intuicion a la escritura de tests.

6.6 Test/test_clase.cpp File Reference

Tests sobre los metodos de la calculadora.

```
#include "gtest/gtest.h"
#include <cppunit/TestCase.h>
#include <cppunit/TestFixture.h>
#include <cppunit/TestRunner.h>
#include <cppunit/TestResult.h>
#include <cppunit/TestResultCollector.h>
#include <cppunit/ui/text/TextTestRunner.h>
#include <cppunit/extensions/HelperMacros.h>
#include <cppunit/extensions/TestFactoryRegistry.h>
#include <cppunit/CompilerOutputter.h>
#include <cppunit/BriefTestProgressListener.h>
#include "test.h"
#include "test_clase.test"
```

Namespaces

- Tester

  Este namespace es el encargado de la definición de los test suites, los test y la propia llamada a estos últimos.

6.6.1 Detailed Description

Tests sobre los metodos de la calculadora.

Author

Francisco Magdaleno francisco.magdalenog@gmail.com

6.7 Test/test_clase.test File Reference

Tests sobre los metodos de la calculadora.

```
#include "../AppEjemplo/calculadora.hpp"
```

Functions

- void test_resta (void)

  Este metodo comprueba la resta de dos valores.
- void test_suma (void)

  Este metodo comprueba la suma de dos valores.
- void test_multiplicacio (void)

  Este metodo comprueba la multiplicacion de dos valores.
- void test_divisio (void)

  Este metodo comprueba la division de dos valores.
- void test_modul (void)

  Este metodo comprueba el modulo de dos valores.
- void test_defines (void)

  Este metodo comprueba el correcto funcionamiento de las macros que no se usan en la calculadora.
- void test_cal_run_tests (void)

  Este metodo resume todos los demas en uno.

6.7.1 Detailed Description

Tests sobre los metodos de la calculadora.

Author

Francisco Magdaleno francisco.magdalenog@gmail.com

6.7.2 Function Documentation

6.7.2.1 test_cal_run_tests()

```
void test_cal_run_tests (
            void  )
```

Este metodo resume todos los demas en uno.

6.7.2.2 test_defines()

```
void test_defines (
            void  )
```

Este metodo comprueba el correcto funcionamiento de las macros que no se usan en la calculadora.

6.7.2.3 test_divisio()

```
void test_divisio (
            void  )
```

Este metodo comprueba la division de dos valores.

6.7.2.4 test_modul()

```
void test_modul (
            void  )
```

Este metodo comprueba el modulo de dos valores.

6.7.2.5 test_multiplicacio()

```
void test_multiplicacio (
            void  )
```

Este metodo comprueba la multiplicacion de dos valores.

6.7.2.6 test_resta()

```
void test_resta (
            void  )
```

Este metodo comprueba la resta de dos valores.

6.7.2.7 test_suma()

```
void test_suma (
            void  )
```

Este metodo comprueba la suma de dos valores.

Index