Testing di un Programma: CUnit

Test di un programma

- A prescindere dalla metodologia da usare

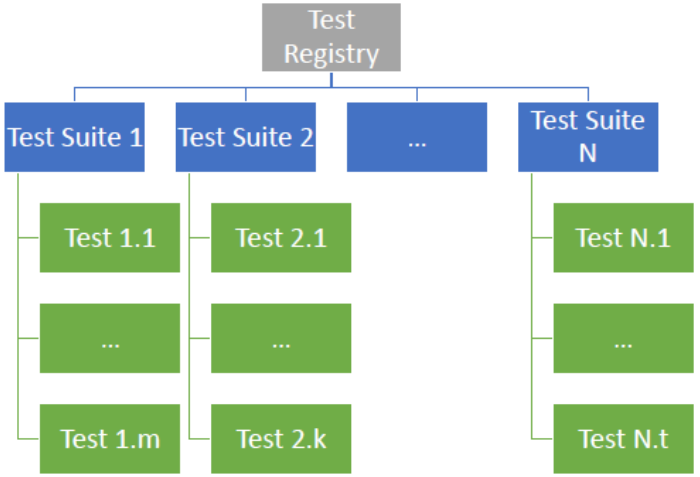
- il test deve essere eseguito parallelamente alla implementazione
- la direttiva principale è eseguire il test di unità
  - procedura e/o funzione
  - blocco di codice

Test di unità (di un programma)

- Test di unità riguarda il test di funzione e procedura
- Si definiscono casi di test che rappresentano "requisiti" che l'unità deve obbligatoriamente soddisfare.
- Si può usare CUnit, framework per unit test di programmi scritti in linguaggio C.
- Libreria da includere in Eclipse CDT (C Development Tooling)

CUnit

- Strutturato in
  - Test Registry
    - piano di tutti di casi di test
    - composto da test suite



  - Test Suite
    - composti dai casi di test su una singola funzione/procedura (test method)
    - test suite eseguite nello stesso ordine con cui sono inserite nel test registry
    - test method eseguiti nello stesso ordine con cui sono inserite nel test suite

Installazione CUnit in Eclipse CDT

Software necessary:

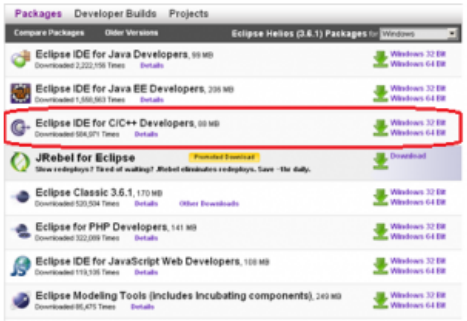
1. Eclipse CDT per Windows
2. MinGW
3. CUnit test framework

Installazione CUnit in Eclipse CDT

Installazione Eclipse CDT in Windows

Questo passo provvede a installare l'ambiente Eclipse CDT. Effettuate il download di "Eclipse IDE for C/C++ Developers", accertandovi di selezionare il sistema operativo corretto e l'architettura giusta (32 o 64 bit).

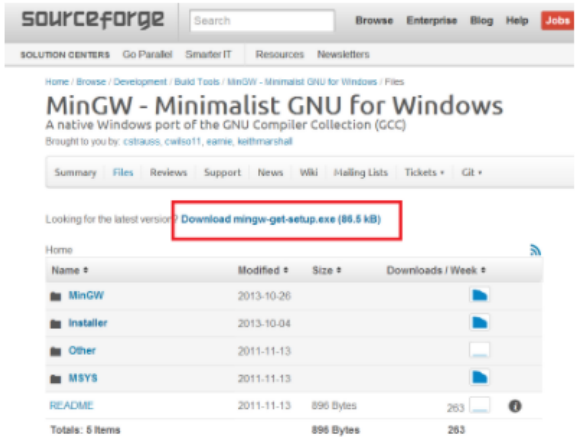
Per installare è sufficiente decomprimere l'archivio in una cartella qualunque, per es. "*C:\eclipsecdt*".

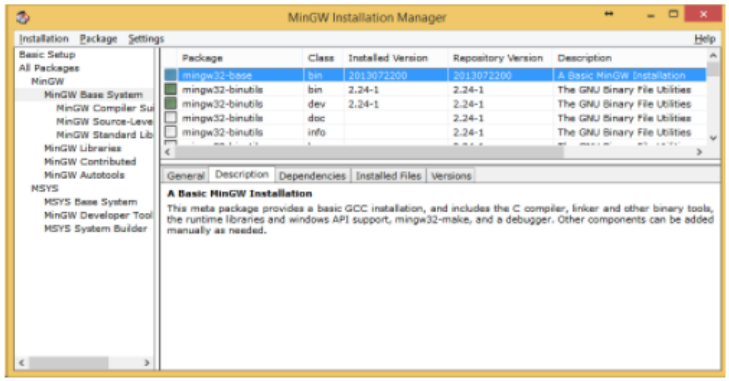


Installazione CUnit in Eclipse CDT

Installazione MinGW

Effettuate il download del compilatore MinGW (GCC) scaricando ed eseguendo il file "mingw-get-setup." (Nota: avrete bisogno di essere collegati a Internet per eseguire l'installazione). Lanciato, l'installer di MinGW, per prima cosa espandete la voce "All Packages", quindi selezionate "MinGW Base system" nell'albero a sinistra, individuate nella lista a destra la voce "mingw32-base" e con il tasto destro scegliete "Mark for installation". A questo punto, dal menu "Installation" in alto a sinistra scegliete la voce "Apply changes" e poi premete sul pulsante "Apply": l'installer si scaricherà e installerà automaticamente l'ultima versione del compilatore e le sue dipendenze.





Installazione CUnit in Eclipse CDT

Terminata l'installazione, aggiungete il percorso "**C:\MinGW\bin**" alla variabile di ambiente *Path*. Per fare questo, cliccate con il tasto destro su "Computer" e Scegliete "Proprietà". Selezionate il tab "Impostazioni Avanzate" e quindi premete il bottone "Variabili di ambiente". Selezionate la variabile di ambiente *Path* dalla lista e cliccate "Modifica". Nel campo "Valore variabile" dovete aggiungere il percorso "**C:\MinGW\bin**" . È importante che 1) non cancelliate i percorsi preesistenti; 2) tutti i percorsi in questo campo siano separati da dei ";" senza aggiungere spazi.

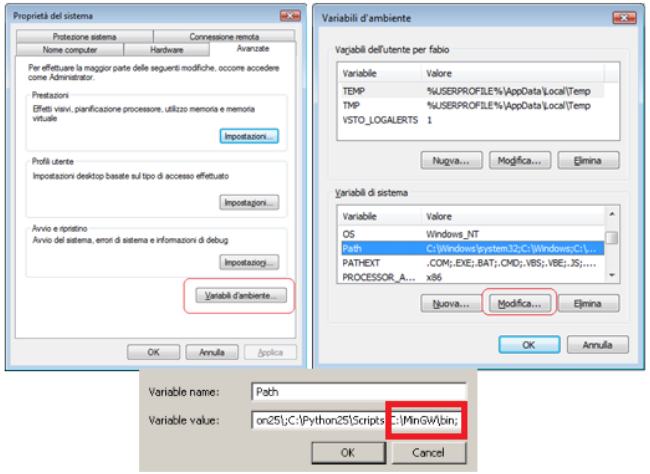
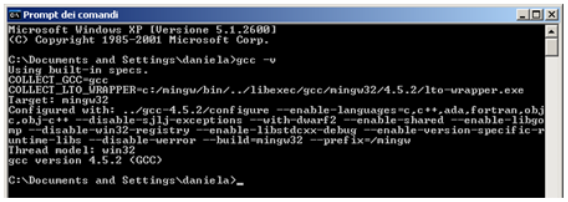


Figura 3

Verificate che l'installazione di gcc sia andata a buon fine. Aprite la finestra dei comandi e digitate il comando ***gcc -v***. Se il setup è andato a buon fine, vedrete un output simile al seguente:



Installazione CUnit in Eclipse CDT

Terminata l'installazione, aggiungete il percorso "**C:\MinGW\bin**" alla variabile di ambiente *Path*. Per fare questo, cliccate con il tasto destro su "Computer" e Scegliete "Proprietà". Selezionate il tab "Impostazioni Avanzate" e quindi premete il bottone "Variabili di ambiente". Selezionate la variabile di ambiente *Path* dalla lista e cliccate "Modifica". Nel campo "Valore variabile" dovete aggiungere il percorso "**C:\MinGW\bin**" . È importante che 1) non cancelliate i percorsi preesistenti; 2) tutti i percorsi in questo campo siano separati da dei ";" senza aggiungere spazi.

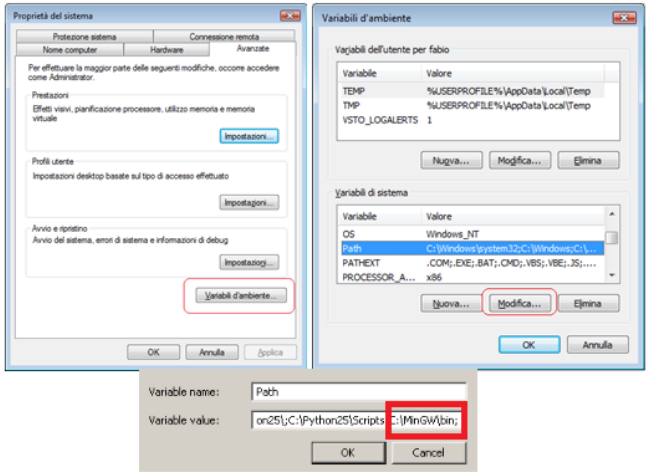
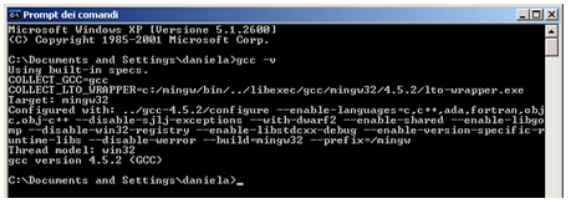


Figura 3

Verificate che l'installazione di gcc sia andata a buon fine. Aprite la finestra dei comandi e digitate il comando ***gcc -v***. Se il setup è andato a buon fine, vedrete un output simile al seguente:



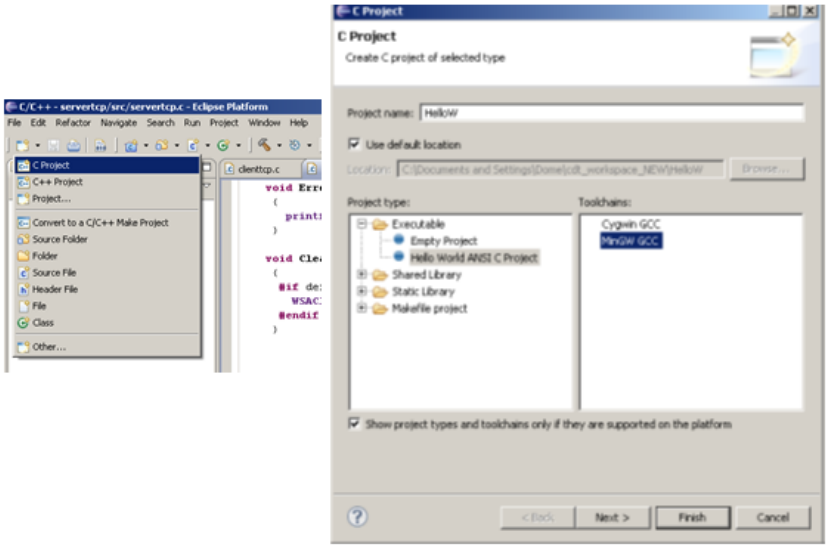
Installazione CUnit in Eclipse CDT

Installazione CUnit

Scaricate lo zip di CUnit e scompattatelo in una cartella a scelta, per esempio **C:\CUnit-2.1-0-winlib\CUnit-2.1-0**. All'interno della sottocartella include troverete un'altra cartella chiamata *CUnit*. Prendetela e copiatela all'interno della cartella **C:\MinGW\include**. Ora invece, andate nella cartella **C:\CUnit-2.1-0-winlib\CUnit-2.1-0\lib**. Vi troverete due file, *libcunit.a* e *libcunit_dll.a*, che dovrete copiare in **C:\MinGW\lib**. Infine, copiate il file libcunit.dll che si trova in **C:\CUnit-2.1-0-winlib\CUnit-2.1-0\bin** all'interno della cartella **C:\MinGW\bin**.

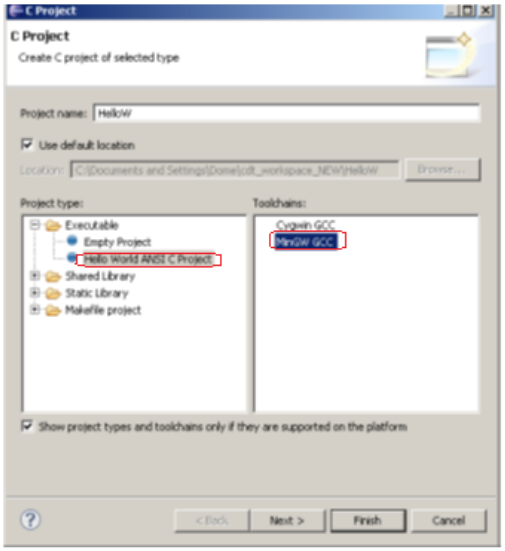
Configurazione Eclipse CDT

A questo punto siete pronti ad eseguire eclipse.exe e a compilare programmi C/C++. Avviate Eclipse e selezionate il workspace, cioè la cartella dove risiederanno i progetti. Per creare un nuovo progetto C, selezionate il wizard corrispondente come da figura.



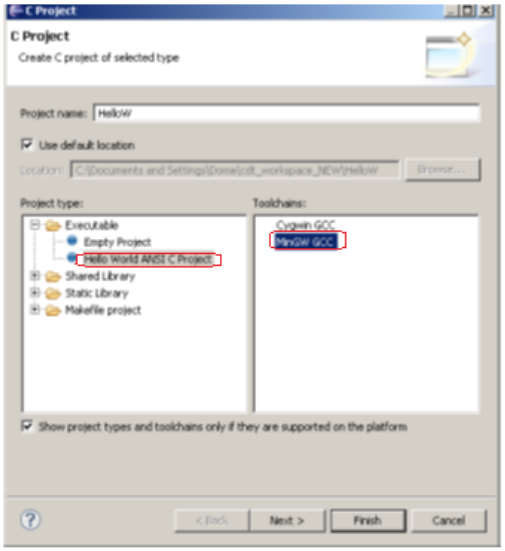
Installazione CUnit in Eclipse CDT

Nel wizard inserite il nome progetto (in questo caso “HelloW”). Dopodiché, selezionate Executable (perché vogliamo creare un programma eseguibile in questo caso) e “Hello World Ansi C Project”. Fate attenzione a scegliere “MinGW GCC” nella lista toolchain (è il compilatore che abbiamo installato nei passi precedenti). A questo punto, premete “Finish” per creare il progetto.



Installazione CUnit in Eclipse CDT

Nel wizard inserite il nome progetto (in questo caso “HelloW”). Dopodiché, selezionate Executable (perché vogliamo creare un programma eseguibile in questo caso) e “Hello World Ansi C Project”. Fate attenzione a scegliere “MinGW GCC” nella lista toolchain (è il compilatore che abbiamo installato nei passi precedenti). A questo punto, premete “Finish” per creare il progetto.



Installazione CUnit in Eclipse CDT

Una volta creato, il progetto conterrà uno scheletro main.c per la stampa di una stringa. Per compilare, premete il simbolo del martello oppure CTRL+b. Se tutto è andato a buon fine, troverete una cartella "Binaries" tra le cartelle di progetto. Selezionate il file exe nella cartella Binaries. A questo punto potete eseguire il programma compilato all'interno di eclipse, premendo il tasto "Run" sulla toolbar e selezionate "Run As > Local C/C++ Application", come da figura. L'output del programma comparirà nella vista "Console" in basso.

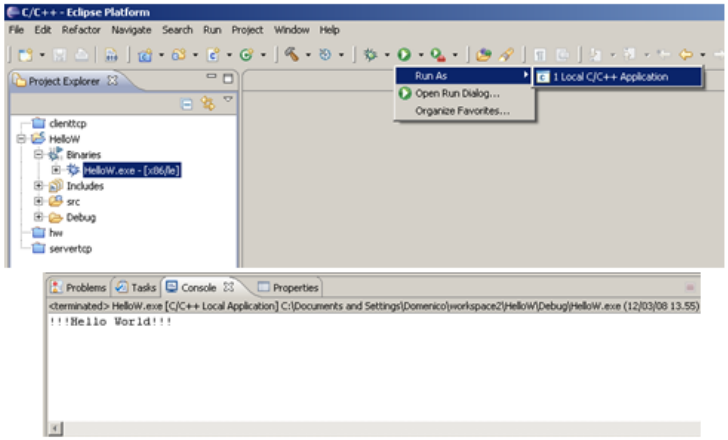
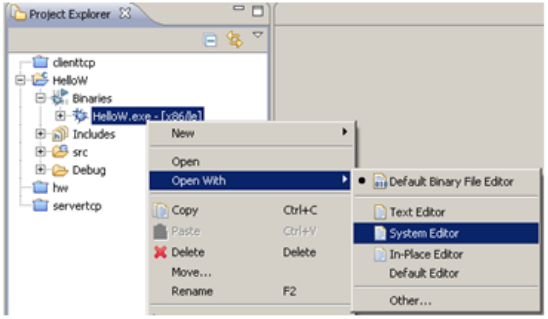


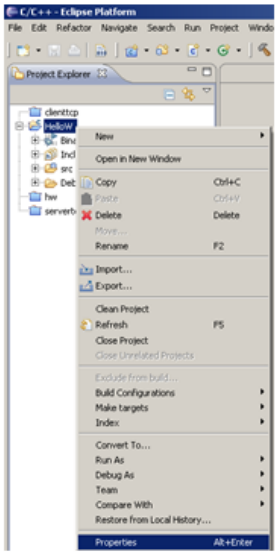
Figura 7

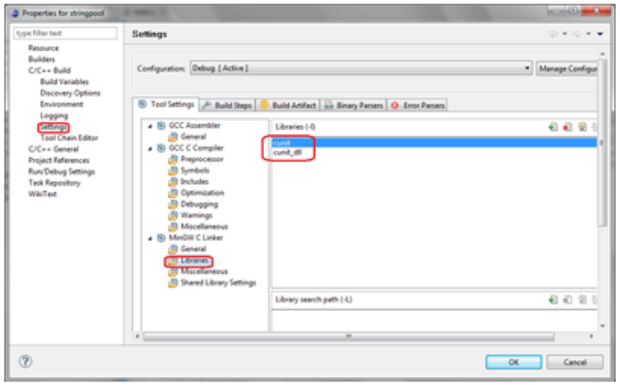
Invece, per eseguire il programma "esternamente" dal prompt dei comandi, selezionate il file eseguibile, premete il tasto destro e dal menu contestuale selezionate "Open with > System editor".



Installazione CUnit in Eclipse CDT

Poiché la nostra applicazione avrà bisogno di eseguire casi di test in CUnit, è necessario informare il compilatore su dove trovare la libreria esterna. Per far questo, selezionate la cartella del progetto, premete il tasto destro e quindi scegliete "Proprietà". Dalla finestra delle proprietà, scegliete "C/C++ Build > Settings". Dopodichè, nel tab "Settings" scegliete la voce "Libraries" sotto la voce "MinGW C Linker". Quindi cliccate sul tasto per aggiungere la libreria. Per includere le librerie "libcunit.a" e "libcunit_dll.a" dovete soltanto scrivere "cunit" e "cunit_dll".





Esecuzione di Unit Test

1. Includere CUnit nei moduli coinvolti nel test:

```
…
#include < stdio.h  >
#include  < stdlib.h >
#include “CUnit/Basic.h”
…
```

Esecuzione di Unit Test

2. Scrivere test method
   - È una procedura con lista di parametri vuota

     ```
     void test_myfunction (void)
     ```
   - `myfunction` è funzione/procedura da testare
   - uso di asserzioni per verificare che funzioni/procedure producono esattamente valori attesi
   - **asserzione**: predicato logico a valore booleano che verifica una espressione

     ```
     CU_ASSERT(expression)
     ```
     - CU_ASSERT(0==0)= true
     - CU_ASSERT(0==1)= false

Esecuzione di Unit Test

2. Scrivere test method
   - CUnit offre diversi tipi di asserzioni, ad esempio
   - `CU_ASSERT_EQUAL(expression,value)=true,` se il valore restituito dalla espressione è uguale a quello fornito
   - `CU_ASSERT_NOT_EQUAL(expression,value)=true,` se il valore restituito dalla espressione non è uguale a quello fornito
   - esempio, `CU_ASSERT_EQUAL(minimo_voto(),18))`

     ovvero,''asserisco che il voto minimo di un esame'' è uguale (EQUAL) a 18.

Esecuzione di Unit Test

2. Scrivere test method
   - `CU_ASSERT_EQUAL(minimo_voto(),18))`

     ovvero,''asserisco che il voto minimo di un esame'' è uguale (EQUAL) a 18.
   - I valori usati nelle asserzioni sono quelli usati per la definizione dei casi limite e per le classi di equivalenza

```
CU_ASSERT_TRUE(esame_superato(18)))
CU_ASSERT_FALSE(esame_superato(17)))
```

Esecuzione di Unit Test

2. Tipi principali di asserzioni:

Asserzione	Significato
CU_ASSERT(*int espressione*) **CU_TEST**(*int espressione*)	Asserisce che espressione è TRUE (diverso da 0)
CU_ASSERT_TRUE(*valore*)	Asserisce che valore è TRUE (diverso da 0)
CU_ASSERT_FALSE(*valore*)	Asserisce che valore è FALSE (uguale a 0)
CU_ASSERT_EQUAL(*reale, atteso*)	Asserisce che reale == atteso
CU_ASSERT_NOT_EQUAL(*reale, atteso*)	Asserisce che reale != atteso
CU_ASSERT_STRING_EQUAL(*reale, atteso*)	Asserisce che le stringhe reale e atteso coincidono
CU_ASSERT_STRING_NOT_EQUAL(*reale, atteso*)	Asserisce che le stringhe reale e atteso differiscono

Esecuzione di Unit Test

2. Esempio di test method
   - funzione da testare: max(int, int)

```
#define MAXINT 1000
void test_max(void){
     CU_ASSERT_EQUAL(max(-MAXINT,+MAXINT), +MAXINT);  //verificata
    CU_ASSERT_TRUE(max(-MAXINT,+MAXINT) = =  +MAXINT); //verificata
    CU_TEST(max(0,0)==0) //verificata
    CU_ASSERT_TRUE(max(5,6) = =  2); // non verificata
        }
```

Esecuzione di Unit Test

2. Esempio di test method
    - funzione da testare: factorial(int)

```
void test_factorial(void){
      CU_ASSERT_EQUAL(factorial(4), 12);  //non verificata
     CU_ASSERT (factorial(0) = = 1);  // verificata
      CU_TEST(factorial(1) = = 1);  // verificata
}
```

Esecuzione di Unit Test

3. Inizializzazione test registry

```
60
61 // ************************************************
62 //  TEST di UNITA'
63
64 int main() {
65     /* inizializza registro - e' la prima istruzione */
66     CU_initialize_registry();
67
```

Esecuzione di Unit Test

4. Creazione test suite ed aggiornamento test registry
   - una test suite è definita da un descrizione testuale (stringa), procedura di inizializzazione (init), procedura di pulitura (clean)

```
/* Aggiungi le suite al test registry */
CU_pSuite pSuite_A = CU_add_suite("Suite_A", init_suite_default, clean_suite_default);
CU_pSuite pSuite_B = CU_add_suite("Suite_B", init_suite_default, clean_suite_default);
```

   - le test suite devono essere inizializzate e ripulite prima e dopo l'uso, al fine di preparare (init) e liberare (clean) le risorse (esempio: file, memoria,…) usate specificatamente per il caso di test
   - le procedure init() e clean() devono essere scritte dallo sviluppatore

Esecuzione di Unit Test

4. Creazione test suite ed aggiornamento test registry

```
// Alloca tutte le risorse necessarie all'esecuzione dei test
 int init_suite_default(void)  {
     return 0; // tutto ok
 }

 // dealloca tutte le risorse allocate all'inizializzazione
 int clean_suite_default(void) {
     return 0; // tutto ok
 }
```

Esecuzione di Unit Test

5. Aggiungere test method alle test suite
   - si specifica la test suite, descrizione testuale e nome del test method

```
/* Aggiungi i test alle suite
 * NOTA - L'ORDINE DI INSERIMENTO E' IMPORTANTE
 */
CU_add_test(pSuite_A, "test of f1()", test_f1);
CU_add_test(pSuite_A, "test of f3()", test_f3);

CU_add_test(pSuite_B, "test of f4()", test_f4);
CU_add_test(pSuite_B, "test of f2()", test_f2);
```

Esecuzione di Unit Test

6. Esecuzione del test registry
   - la procedura CU_basic_run_tests() esegue tutte le suite del registry e visualizza i risultati
   - è possibile settare il livello di dettaglio della visualizzazione

```
/* Esegue tutti i casi di test con output sulla console */
CU_basic_set_mode(CU_BRM_VERBOSE);
CU_basic_run_tests();
```

Esecuzione di Unit Test

7. Terminazione e pulizia del test registry attraverso la procedura CU_cleanup_registry(void)
   - la clausola *return* fornirà l'eventuale codice di errore di CUnit

```
/* Pulisce il registro e termina lo unit test */
CU_cleanup_registry();

return CU_get_error();
```

Riepilogo

```c
#include <stdio.h>
 #include "CUnit/Basic.h"

/* Funzione di inizializzazione. */
 int init_suite1(void) {
     return 0;  // nessuna inizializzazione
 }

 /* Funzione di cleanup. */
 int clean_suite1(void) {
     return 0; // nessun cleanup
 }
```

(1) Inclusione della libreria

(2) Metodi di Inizializzazione e Pulitura

Riepilogo

(3) Metodi di Test

```
/* Test method */
 void test_fun1(void) {
      CU_FAIL("test method not implemented"); // metodi vuoti
 }
 /* Test method */
 void test_fun2(void) {
      CU_FAIL("test method not implemented");
 }
 /* Test method */
 void test_fun3(void) {
      CU_FAIL("test method not implemented");
 }
```

Riepilogo

(4) Main del programma

```
int main()
 {
    CU_initialize_registry());
    CU_pSuite pSuite = CU_add_suite("Suite_1",init_suite1,clean_suite1);
    CU_add_test(pSuite, "test1", test_fun1));
    CU_add_test(pSuite, "test2", test_fun2));
    CU_add_test(pSuite, "test2", test_fun3));
    …

    CU_basic_set_mode(CU_BRM_VERBOSE);
    CU_basic_run_tests();
    CU_cleanup_registry();
    return CU_get_error();
}
```