

C++ programok egységtesztelése googletest segítségével (GKxB_INTM006)

Dr. Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

https://github.com/wajzy/GKxB_INTM006.git

2019. július 13.

Tesztelés célja: a hibákat megtalálni üzembe helyezés előtt

Tesztelés alapelvei

- 1 A tesztelés bizonyos hibák jelenlétét jelezheti (ha nem jelzi, az nem jelent automatikusan hibamentességet)
- 2 Nem lehetséges kimerítő teszt (a hangsúly a magas kockázatú részeken van)
- 3 Korai teszt (minél hamarabb találjuk meg a hibát, annál olcsóbb javítani)
- 4 Hibák csoportosulása (azokra a modulokra/bemenetekre kell tesztelni, amelyekre a legvalószínűbben hibás a szoftver)
- 5 Féregirtó paradoxon (a tesztesetek halmazát időnként bővíteni kell, mert ugyanazokkal a tesztekkel nem fedhetünk fel több hibát)
- 6 Körülmények (tesztelés alapossága függ a felhasználás helyétől, a rendelkezésre álló időtől, stb.)
- 7 A hibátlan rendszer téveszméje (A megrendelő elsősorban az igényeinek megfelelő szoftvert szeretne, és csak másodsorban hibamenteset; verifikáció vs. validáció)

Tesztelési technikák

Fekete dobozos (black-box, specifikáció alapú)

A tesztelő nem látja a forrást, de a specifikációt igen, és hozzáfér a futtatható szoftverhez. Összehasonlítjuk a bemenetekre adott kimeneteket az elvárt kimenetekkel.

Fehér dobozos (white-box, strukturális teszt)

Kész struktúrákat tesztelünk, pl.:

- kódsorok,
- elágazások,
- metódusok,
- osztályok,
- funkciók,
- modulok.

Lefedettségi: a struktúra hány %-át tudjuk tesztelni a tesztesetekkel?

Egységteszt (unit test): a metódusok struktúra tesztje.

A tesztelés szintjei:

- 1 komponensteszt (egy komponens tesztelése)
 - 1 egységteszt
 - 2 modulteszt
- 2 integrációs teszt (kettő vagy több komponens együttműködése)
- 3 rendszerteszt (minden komponens együtt)
- 4 átvételi teszt (kész rendszer)

Kik végzik a tesztelést?

1-3 Fejlesztő cég

4 Felhasználók

Komponensteszt

- fehér dobozos teszt
- egységteszt
 - bemenet → kimenet vizsgálata
 - nem lehet mellékhata
 - regressziós teszt: módosítással elronthattunk valamit, ami eddig jó volt → megismételt egységtesztek
- modulteszt
 - nem funkcionális tulajdonságok: sebesség, memóriaszivárgás (memory leak), szűk keresztmetszetek (bottleneck)

Integrációs teszt

- Komponensek közötti interfészek ellenőrzése, pl.
 - komponens - komponens (egy rendszer komponenseinek együttműködése)
 - rendszer - rendszer (pl. OS és a fejlesztett rendszer között)
- Jellemző hibaokok: komponenseket eltérő csapatok fejlesztik, elégtelen kommunikáció
- Kockázatok csökkentése: megelőbbi integrációs tesztekkel

Rendszerteszt: a termék megfelel-e a

- követelmény specifikációnak,
- funkcionális specifikációnak,
- rendszertervnek.

Gyakran fekete dobozos, külső cég végzi (elfogulatlanság)

Leendő futtatási környezet imitációja

Átvételi teszt, fajtái:

- alfa: kész termék tesztelése a fejlesztőnél, de nem általa (pl. segédprogramok)
- béta: szűk végfelhasználói csoport
- felhasználói átvételi teszt: minden felhasználó használja, de nem éles termelésben. Jellemző a környezetfüggő hibák megjelenése (pl. sebesség)
- üzemeltetői átvételi teszt: rendszergazdák végzik, biztonsági mentés, helyreállítás, stb. helyesen működnek-e

Rengeteg C++ egységteszt keretrendszerből lehet választani:

- Wiki oldal
- Exploring the C++ Unit Testing Framework Jungle
- C++ Unit Test Frameworks

Részletesen megvizsgáljuk: googletest

A googletest főbb tulajdonságai

- platformfüggetlen (Linux, Windows, Mac)
- független és megismételhető tesztek
- struktúrálható tesztek (teszt program → teszt csomag → teszteset)
- informatív
- leveszi a tesztelés technikai részének terhét a tesztelőről
- gyors (megosztott erőforrások)
- könnyen tanulható (xUnit architektúra)

Telepítés (Ubuntu 18.04 LTS)

```
sudo apt install libgtest-dev
```

Teszt keretrendszer forrásainak beszerzése.

```
sudo apt install cmake
```

Ezzel végezzük a forráskódok automatizált fordítását.

```
cd /usr/src/gtest
```

Ebben a mappában találhatóak a források.

```
sudo cmake CMakeLists.txt
```

Összeállító (build) környezet előkészítése.

```
sudo make
```

Összeállítás indítása.

```
sudo ln -st /usr/lib/ /usr/src/gtest/libgtest.a  
sudo ln -st /usr/lib/ /usr/src/gtest/libgtest_main.a
```

Szimbolikus hivatkozások létrehozása.

Feladat

Készítsünk mátrixműveleteket megvalósító osztályt, ami elsőként egy mátrixszorzást valósít meg.

Az $A[a_{i,j}]_{m \times n}$ és $B[b_{i,j}]_{n \times p}$ mátrixok szorzatán azt a $C[c_{i,j}]_{m \times p}$ mátrixot értjük, amelyre $c_{i,j} = a_{i,1} \cdot b_{1,j} + a_{i,2} \cdot b_{2,j} + \dots + a_{i,n} \cdot b_{n,j} = \sum_{k=1}^n a_{i,k} \cdot b_{k,j}$

matrix01.h

```
1 #include<vector>
2 #include<iostream>
3 namespace szeMatrix {
4
5 template<class T>
6 class Matrix {
7     protected:
8         std::vector<std::vector<T>> mtx;
9
10    public:
11        Matrix(std::vector<std::vector<T>> src) {
12            mtx = src;
13        }
```

matrix01.h

```
14     Matrix<T> mul(Matrix<T> right);  
15     void print();  
16     int getRowCount() { return mtx.size(); }  
17     int getColCount() { return mtx[0].size(); }  
18     T get(int row, int column) { return mtx[row][column]; }  
19 };
```

matrix01.h

```
21 template<class T>
22 void Matrix<T>::print() {
23     for(std::vector<T> row : mtx) {
24         for(T elem : row) {
25             std::cout << elem << '\t';
26         }
27         std::cout << std::endl;
28     }
29 }
```

matrix01.h

```
31 template<class T>
32 Matrix<T> Matrix<T>::mul(Matrix<T> right) {
33     // Rows of left matrix and result matrix
34     int i = mtx.size();
35     // Columns of right matrix and res. matrix
36     int j = right.mtx[0].size();
37     // Columns of left matrix and rows of right matrix
38     int k = right.mtx.size();
39
40     // Creating an empty result matrix
41     std::vector<std::vector<T>>> res;
42     // Resizing and filling it with zeros
43     res.resize(i, std::vector<T>(j, 0.));
```


matrix01.h

```
45     for(int r=0; r<i; r++) { // Matrix multiplication
46         for(int c=0; c<j; c++) {
47             for(int item=0; item<k; item++) {
48                 res[r][c] += mtx[r][item]*right.mtx[item][c];
49             }
50         }
51     }
52
53     return Matrix(res);
54 }
55
56 }
```

example01.cpp

```
1  #include <vector>
2  #include "matrix01.h"
3
4  int main() {
5      std::vector<std::vector<double>> v1 = {
6          {11, 12, 13, 14},
7          {21, 22, 23, 24},
8          {31, 32, 33, 34}
9      };
10
11      std::vector<std::vector<double>> v2;
12      v2.resize(4, std::vector<double>(3, 1.));
```

example01.cpp

```
14     szeMatrix :: Matrix<double> m1(v1 );
15     szeMatrix :: Matrix<double> m2(v2 );
16     szeMatrix :: Matrix<double> multiplied = m1.mul(m2);
17     multiplied.print ();
18
19     return 0;
20 }
```

Kimenet

50	50	50
90	90	90
130	130	130

Készítsünk az `example01.cpp` alapján googletest alapú tesztprogramot!

`matrix01test.cpp`

```
1 #include "matrix01.h"
2 #include <vector>
3 #include <gtest/gtest.h>
4
5 TEST(MulTest, meaningful) {
6     std::vector<std::vector<double>> left = {
7         {11, 12, 13, 14},
8         {21, 22, 23, 24},
9         {31, 32, 33, 34}
10    };
11    std::vector<std::vector<double>> right;
12    right.resize(4, std::vector<double>(3, 1.));
```

matrix01test.cpp

```
13     std::vector<std::vector<double>> expected = {  
14         {50, 50, 50},  
15         {90, 90, 90},  
16         {130, 130, 130}  
17     };  
18     szeMatrix::Matrix<double> m1( left );  
19     szeMatrix::Matrix<double> m2( right );  
20     szeMatrix::Matrix<double> multiplied = m1.mul(m2);
```

matrix01test.cpp

```
21  ASSERT_EQ(expected.size(), multiplied.getRowCount());
22  ASSERT_EQ(expected[0].size(), multiplied.getColCount());
23  for(unsigned row=0; row<expected.size(); row++) {
24      for(unsigned col=0; col<expected[row].size(); col++) {
25          EXPECT_EQ(expected[row][col], multiplied.get(row, col));
26      }
27  }
28  }
29
30  int main(int argc, char **argv) {
31      ::testing::InitGoogleTest(&argc, argv);
32      return RUN_ALL_TESTS();
33  }
```

CMakeLists.txt

```
1  cmake_minimum_required(VERSION 2.6)
2
3  # Locate GTest
4  find_package(GTest REQUIRED)
5  include_directories(${GTEST_INCLUDE_DIRS})
6
7  # Link runTests with what we want to test
8  # and the GTest and pthread library
9  add_executable(runTests matrix01test.cpp)
10 target_link_libraries(runTests ${GTEST_LIBRARIES} pthread)
```

```
cmake CMakeLists.txt
```

Összeállító (build) környezet beállítása.

```
make
```

Összeállítás indítása.

```
./runTests
```

Tesztprogram indítása.

Kimenet

```
[=====] Running 1 test from 1 test case.
[-----] Global test environment set-up.
[-----] 1 test from MulTest
[ RUN      ] MulTest.meaningful
[          OK ] MulTest.meaningful (0 ms)
[-----] 1 test from MulTest (0 ms total)

[-----] Global test environment tear-down
[=====] 1 test from 1 test case ran. (0 ms total)
[ PASSED   ] 1 test.
```


Teszteset (test case)

"A set of preconditions, inputs, actions (where applicable), expected results and postconditions, developed based on test conditions."

(meaningful, ld. `matrix01test.ccp` 5. sor)

Tesztkészlet (test suite)

"A set of test cases or test procedures to be executed in a specific test cycle."

(MulTest, ld. `matrix01test.ccp` 5. sor)

Tesztprogram (test program)

Egy vagy több tesztkészletet foglal magába.

Assertion (\approx állítás, követelés) Ellenőrizzük valamely elvárásunk teljesülését \rightarrow siker (success), nem végzetes hiba (nonfatal failure), végzetes hiba (fatal failure).

Makrók:

EXPECT_* nem végzetes hibát generál, ajánlott (több hiba jelezhető egyszerre)

ASSERT_* végzetes hibát generál, azonnal leállítja a tesztet (nincs értelme a folytatásnak; pl. ha két mátrix nem azonos méretű, nincs értelme az elemeiket összehasonlítani). **Erőforrások felszabadítása, takarítás is elmarad!**

Rontsuk el a kódot! („Elfelejtjük” összegezni a szorzatokat.)

matrix02.h (matrix02test.cpp, CMakeLists.txt)

```
45     for(int r=0; r<i; r++) { // Matrix multiplication
46         for(int c=0; c<j; c++) {
47             for(int item=0; item<k; item++) {
48                 // res[r][c] += mtx[r][item]*right.mtx[item][c];
49             }
50         }
51     }
```

Kimenet

```
[=====] Running 1 test from 1 test case.  
[-----] Global test environment set-up.  
[-----] 1 test from MulTest  
[ RUN      ] MulTest.meaningful  
/home/wajzy/Dokumentumok/gknb_intm006/GKxB_INTM006/02/matrix02test.cpp:25: Failure  
    Expected: expected[row][col]  
    Which is: 50  
To be equal to: multiplied.get(row, col)  
    Which is: 0  
...
```

Kimenet

```
...
/home/wajzy/Dokumentumok/gknb_intm006/GKxB_INTM006/02/matrix02test.cpp:25: Failure
    Expected: expected[row][col]
    Which is: 130
To be equal to: multiplied.get(row, col)
    Which is: 0
[ FAILED ] MulTest.meaningful (1 ms)
[-----] 1 test from MulTest (1 ms total)

[-----] Global test environment tear-down
[=====] 1 test from 1 test case ran. (1 ms total)
[ PASSED ] 0 tests.
[ FAILED ] 1 test, listed below:
[ FAILED ] MulTest.meaningful

1 FAILED TEST
```

Most rontsuk el másképp a kódot! (Túl nagy lesz az eredmény mátrix.)

matrix03.h (CMakeLists.txt)

```
40 // Creating an empty result matrix
41 std::vector<std::vector<T>> res;
42 // Resizing and filling it with zeros
43 //res.resize(i, std::vector<T>(j, 0.));
44 res.resize(i*2, std::vector<T>(j, 0.));
```

matrix03test.cpp

```
21 ASSERT_EQ(expected.size(), multiplied.getRowCount())
22     << "A sorok szama elter! Elvart: " << expected.size()
23     << ", kapott: " << multiplied.getRowCount();
24 ASSERT_EQ(expected[0].size(), multiplied.getColCount())
25     << "Az oszlopok szama elter! Elvart: " << expected[0].size()
26     << ", kapott: " << multiplied.getColCount();
27 for(unsigned row=0; row<expected.size(); row++) {
28     for(unsigned col=0; col<expected[row].size(); col++) {
29         EXPECT_EQ(expected[row][col], multiplied.get(row, col))
30             << "Nem egyezik az elemek erteke a [" << row << "]["
31             << col << "] helyen!";
32     }
33 }
```

Kimenet

```
[=====] Running 1 test from 1 test case.
[-----] Global test environment set-up.
[-----] 1 test from MulTest
[ RUN      ] MulTest.meaningful
/home/wajzy/Dokumentumok/gknb_intm006/GKxB_INTM006/03/matrix03test.cpp:21: Failure
  Expected: expected.size()
    Which is: 3
To be equal to: multiplied.getRowCount()
    Which is: 6
A sorok szama elter! Elvart: 3, kapott: 6
[  FAILED  ] MulTest.meaningful (0 ms)
[-----] 1 test from MulTest (0 ms total)

[-----] Global test environment tear-down
[=====] 1 test from 1 test case ran. (0 ms total)
[  PASSED  ] 0 tests.
[  FAILED  ] 1 test, listed below:
[  FAILED  ] MulTest.meaningful

1 FAILED TEST
```

- Az ASSERT_EQ leállította a tesztet.
- Testreszabott hibaüzeneteket jelenítettünk meg.

Elemi követelmények

Végzetes hibákhoz	Nem végzetes hibákhoz	Követelmény
ASSERT_TRUE(<i>feltétel</i>)	EXPECT_TRUE(<i>feltétel</i>)	<i>feltétel</i> igaz értékű
ASSERT_FALSE(<i>feltétel</i>)	EXPECT_FALSE(<i>feltétel</i>)	<i>feltétel</i> hamis értékű

Relációs követelmények

Végzetes hibákhoz	Nem végzetes hibákhoz	Követelmény
ASSERT_EQ(<i>val1</i> , <i>val2</i>);	EXPECT_EQ(<i>val1</i> , <i>val2</i>);	<i>val1</i> == <i>val2</i>
ASSERT_NE(<i>val1</i> , <i>val2</i>);	EXPECT_NE(<i>val1</i> , <i>val2</i>);	<i>val1</i> != <i>val2</i>
ASSERT_LT(<i>val1</i> , <i>val2</i>);	EXPECT_LT(<i>val1</i> , <i>val2</i>);	<i>val1</i> < <i>val2</i>
ASSERT_LE(<i>val1</i> , <i>val2</i>);	EXPECT_LE(<i>val1</i> , <i>val2</i>);	<i>val1</i> <= <i>val2</i>
ASSERT_GT(<i>val1</i> , <i>val2</i>);	EXPECT_GT(<i>val1</i> , <i>val2</i>);	<i>val1</i> > <i>val2</i>
ASSERT_GE(<i>val1</i> , <i>val2</i>);	EXPECT_GE(<i>val1</i> , <i>val2</i>);	<i>val1</i> >= <i>val2</i>

Megjegyzések

- A feltüntetett operátoroknak definiálnak kell lenniük *val1* és *val2* között.
Lehetőségeink:
 - 1 Felültöltjük az operátorokat.
 - 2 Az `{ASSERT,EXPECT}_ {TRUE,FALSE}` makrókat használjuk, de ezek nem írják a kimenetre az elvárt/kapott értékeket.
- A paraméterek egyszer lesznek kiértékelve, de nem definiált sorrendben (mellékhatások).
- Az `{ASSERT,EXPECT}_EQ` makrók mutatók esetén a címeket hasonlítja össze, nem az ott lévő tartalmat! C-stílusú karakterláncok kezeléséhez külön makrók léteznek. (`string` objektumokkal nincs gond.)
- C++11 szabványnak megfelelő fordító esetén `NULL` helyett `nullptr`-t használjunk (utóbbi nem konvertálható implicit módon `int`-té)!
- Lebegőpontos számok összehasonlításakor kerekítési hibák adódhatnak.

Tesztelésről általában

Ficsor Lajos, Kovács László, Kusper Gábor, Krizsán Zoltán: Szoftvertesztelés

ISTQB CTFL Syllabus 2018

Szakkifejezések kereshető gyűjteménye

googletest

Hivatalos Google tutorial, bevezető

Hivatalos Google tutorial, fejlett technikák

Ubuntu-specifikus részletek

IBM tananyag a googletest-hez