

C++ programok egységtesztelése googletest segítségével (GKxB_INTM006)

Dr. Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

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

2019. július 31.

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.

Lefedettség: 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

Komponenteszt

- fehér dobozos teszt
- egységteszt
 - bemenet → kimenet vizsgálata
 - nem lehet mellékhatása
 - 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: mielőbbi 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 terhet 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 $c_{i,j} = a_{i,1} \cdot b_{1,j} + a_{i,2} \cdot b_{2,j} + \dots + a_{i,n} \cdot b_{n,j} = \sum_{k=1}^n a_{i,k} \cdot b_{k,j}$

01/matrix01.h

```
1 #include<vector>
2 #include<iostream>
3 namespace sizeMatrix {
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        }
14    }
```

01/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 };
```

01/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 }
```

01/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.));
```


01/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 }

```

01/example01.cpp

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

01/example01.cpp

```
14     szMatrix :: Matrix<int> m1(v1 );
15     szMatrix :: Matrix<int> m2(v2 );
16     szMatrix :: Matrix<int> 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!

01/matrix01test.cpp

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

01/matrix01test.cpp

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

01/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 }

```

01/CMakeLists.txt

```
1  cmake_minimum_required(VERSION 2.6)

14 # Locate GTest
15 find_package(GTest REQUIRED)
16 include_directories(${GTEST_INCLUDE_DIRS})
17
18 # Link runTests with what we want to test
19 # and the GTest and pthread library
20 add_executable(runTests matrix01test.cpp)
21 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.

Sajnos a googletest nevezéktana következetlen:

googletest	ISTQB
teszt (test)	teszteset
teszteset (test case)	tesztkészlet

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.)

02/matrix02.h (02/matrix02test.cpp, 02/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.)

03/matrix03.h (03/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.));
```

03/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.

Készítsünk lebegőpontos számokból álló mátrixokat, majd teszteljük a szorzást ismét!

04/matrix04test.cpp (04/matrix04.h, 04/CMakeLists.txt)

```
31 TEST(MulTest, rounding) {
32     std::vector<std::vector<double>> left = {
33         {sqrt(2.), 0.},
34         {0., 1./3.}
35     };
36     std::vector<std::vector<double>> right;
37     right.resize(2, std::vector<double>(2, 1.));
38     std::vector<std::vector<double>> expected = {
39         {1.414213562, 1.414213562},
40         {0.333333333, 0.333333333}
41     };
```

04/matrix04test.cpp

```
42     szMatrix::Matrix<double> m1(left);
43     szMatrix::Matrix<double> m2(right);
44     szMatrix::Matrix<double> multiplied = m1.mul(m2);
45     ASSERT_EQ(expected.size(), multiplied.getRowCount());
46     ASSERT_EQ(expected[0].size(), multiplied.getColCount());
47     for(unsigned row=0; row<expected.size(); row++) {
48         for(unsigned col=0; col<expected[row].size(); col++) {
49             EXPECT_EQ(expected[row][col], multiplied.get(row, col));
50         }
51     }
52 }
```

Kimenet

```
...
[ RUN      ] MulTest.rounding
/home/wajzy/Dokumentumok/gknb_intm006/GKxB_INTM006/04/matrix04test.cpp:49: Failure
Value of: multiplied.get(row, col)
  Actual: 1.41421
Expected: expected[row][col]
Which is: 1.41421
...
/home/wajzy/Dokumentumok/gknb_intm006/GKxB_INTM006/04/matrix04test.cpp:49: Failure
Value of: multiplied.get(row, col)
  Actual: 0.333333
Expected: expected[row][col]
Which is: 0.333333
[ FAILED   ] MulTest.rounding (0 ms)
...
```

A kerekítési hibák érzékelhetetlenek a kimeneten és a teszt sikertelen.

Próbálkozzunk a beépített, lebegőpontos számokat összehasonlító makrókkal!

05/matrix05test.cpp (05/matrix05.h, 05/CMakeLists.txt)

```
47 for (unsigned row=0; row<expected.size(); row++) {  
48     for (unsigned col=0; col<expected[row].size(); col++) {  
49         //EXPECT_EQ(expected[row][col], multiplied.get(row, col));  
50         EXPECT_DOUBLE_EQ(expected[row][col], multiplied.get(row, col));  
51     }  
52 }
```

Kimenet

```
...
[ RUN      ] MulTest.rounding
/home/wajzy/Dokumentumok/gknb_intm006/GKxB_INTM006/05/matrix05test.cpp:50: Failure
Value of: multiplied.get(row, col)
  Actual: 1.4142135623730951
Expected: expected[row][col]
Which is: 1.414213562
...
/home/wajzy/Dokumentumok/gknb_intm006/GKxB_INTM006/05/matrix05test.cpp:50: Failure
Value of: multiplied.get(row, col)
  Actual: 0.33333333333333331
Expected: expected[row][col]
Which is: 0.333333333300000001
[  FAILED  ] MulTest.rounding (0 ms)
...
```

Most már látszik, hogy az értékek közötti különbség nagyobb, mint 4 ULP (Units in the Last Place), ezért tekinti őket a teszt különbözőnek.

Növeljük meg a számok közötti legnagyobb megengedett eltérést!

06/matrix06test.cpp (06/matrix06.h, 06/CMakeLists.txt)

```
47     for (unsigned row=0; row<expected.size(); row++) {
48         for (unsigned col=0; col<expected[row].size(); col++) {
49             //EXPECT_EQ(expected[row][col], multiplied.get(row, col));
50             //EXPECT_DOUBLE_EQ(expected[row][col], multiplied.get(row, col));
51             EXPECT_NEAR(expected[row][col], multiplied.get(row, col), 1e-9);
52         }
53     }
```

Kimenet

```
[=====] Running 2 tests from 1 test case.
[-----] Global test environment set-up.
[-----] 2 tests from MulTest
[ RUN      ] MulTest.meaningful
[          OK ] MulTest.meaningful (0 ms)
[ RUN      ] MulTest.rounding
[          OK ] MulTest.rounding (0 ms)
[-----] 2 tests from MulTest (1 ms total)

[-----] Global test environment tear-down
[=====] 2 tests from 1 test case ran. (1 ms total)
[ PASSED  ] 2 tests.
```

Lebegőpontos számokkal szemben támasztható követelmények

Végzetes hibákhoz	Nem végzetes hibákhoz	Követelmény
ASSERT_FLOAT_EQ(<i>val1</i> , <i>val2</i>);	EXPECT_FLOAT_EQ(<i>val1</i> , <i>val2</i>);	<i>float</i> típusú értékek 4 ULP-n belül
ASSERT_DOUBLE_EQ(<i>val1</i> , <i>val2</i>);	EXPECT_DOUBLE_EQ(<i>val1</i> , <i>val2</i>);	<i>double</i> típusú értékek 4 ULP-n belül
ASSERT_NEAR(<i>val1</i> , <i>val2</i> , <i>abs_error</i>);	EXPECT_NEAR(<i>val1</i> , <i>val2</i> , <i>abs_error</i>);	a két érték különbségének abszolút értéke nem nagyobb <i>abs_error</i> -nál

Próbáljuk meg a mátrixok elemenkénti összehasonlítása helyett a teljes mátrixokat összehasonlítani!

07/matrix07test.cpp (07/matrix07.h, 07/CMakeLists.txt)

```
31 TEST(MulTest, equality) {
32     std::vector<std::vector<double>> left = {
33         {11, 12, 13, 14},
34         {21, 22, 23, 24},
35         {31, 32, 33, 34}
36     };
37     std::vector<std::vector<double>> right;
38     right.resize(4, std::vector<double>(3, 1.));
39     std::vector<std::vector<double>> expected = {
40         {50, 50, 50},
41         {90, 90, 90},
42         {130, 130, 130}
43     };
```

07/matrix07test.cpp

```
44     szMatrix :: Matrix<double> m1( left );
45     szMatrix :: Matrix<double> m2( right );
46     szMatrix :: Matrix<double> mexp( expected );
47     szMatrix :: Matrix<double> multiplied = m1.mul(m2);
48     ASSERT_EQ(mexp.getRowCount(), multiplied.getRowCount());
49     ASSERT_EQ(mexp.getColCount(), multiplied.getColCount());
50     ASSERT_EQ(mexp, multiplied);
51 }
```

Kimenet

```
wajzy@wajzy-notebook:~/Dokumentumok/gknb_intm006/GKxB_INTM006/07$ make
...
/home/wajzy/Dokumentumok/gknb_intm006/GKxB_INTM006/07/matrix07test.cpp:50:3:
  required from here
/usr/include/gtest/gtest.h:1325:16: error: no match for 'operator==' (operand
  types are 'const szMatrix::Matrix<double>' and
  'const szMatrix::Matrix<double>')
    if (expected == actual) {
                ^
...
```

Probléma: az 50. sor `ASSERT_EQ(mexp, multiplied);` utasítása feltételezi az `==` operátor felültöltését a `Matrix` osztályhoz.

08/matrix08.h (08/matrix08test.cpp, 08/CMakeLists.txt)

```
5  template<class T>
6  class Matrix {

10     public:

19         template<class U>
20         friend bool operator==(const Matrix<U> &m1, const Matrix<U> &m2);
21     };

58     template<class U>
59     bool operator==(const Matrix<U> &m1, const Matrix<U> &m2) {
60         return m1.mtx==m2.mtx;
61     }
```

Kimenet

```
wajzy@wajzy-notebook: ~/Dokumentumok/gknb_intm006/GKxB_INTM006/08$ make
[100%] Built target runTests
wajzy@wajzy-notebook: ~/Dokumentumok/gknb_intm006/GKxB_INTM006/08$ ./runTests
[=====] Running 3 tests from 1 test case.
[-----] Global test environment set-up.
[-----] 3 tests from MulTest
[ RUN      ] MulTest.meaningful
[          OK ] MulTest.meaningful (0 ms)
[ RUN      ] MulTest.equality
[          OK ] MulTest.equality (1 ms)
[ RUN      ] MulTest.rounding
[          OK ] MulTest.rounding (0 ms)
[-----] 3 tests from MulTest (1 ms total)

[-----] Global test environment tear-down
[=====] 3 tests from 1 test case ran. (1 ms total)
[ PASSED  ] 3 tests.
```


Teszteljük le a `print()` tagfüggvény kimenetét!

Függvény	Funkció
CaptureStdout()	Megkezdzi az stdout-ra írt tartalom rögzítését
GetCapturedStdout()	Lekérdezi a rögzített tartalmat és leállítja a rögzítést
CaptureStderr()	Megkezdzi az stderr-re írt tartalom rögzítését
GetCapturedStderr()	Lekérdezi a rögzített tartalmat és leállítja a rögzítést

Belső tagfüggvények, használatuk **nem javasolt** (googletest forráskód).

09/matrix09.cpp (09/matrix09.h, 09/CMakeLists.txt)

```
76 TEST(MulTest, print) {
77     std::vector<std::vector<double>>> right;
78     right.resize(2, std::vector<double>(2, 1.));
79     sizeMatrix::Matrix<double> m2(right);
80     const char* expected = "1\t1\t\n1\t1\t\n";
81     testing::internal::CaptureStdout();
82     m2.print();
83     std::string output = testing::internal::GetCapturedStdout();
84     ASSERT_EQ(expected, output.c_str());
85 }
```

Kimenet

```
...
[ RUN      ] MulTest.print
/home/wajzy/Dokumentumok/gknb_intm006/GKxB_INTM006/09/matrix09test.cpp:84: Failure
Value of: output.c_str()
  Actual: 0x1bb1f28
Expected: expected
Which is: 0x475e6a
[ FAILED   ] MulTest.print (0 ms)
...
```

Probléma: a C-stílusú karakterláncok **címeit** hasonlítja össze, nem az ott lévő tartalmat!

C-stílusú karakterláncokkal szemben támasztható követelmények

Végzetes hibákhoz	Nem végzetes hibákhoz	Követelmény
ASSERT_STREQ(<i>str1</i> , <i>str2</i>);	EXPECT_STREQ(<i>str1</i> , <i>str2</i>);	A két C-stílusú karakterlánc tartalma azonos
ASSERT_STRNE(<i>str1</i> , <i>str2</i>);	EXPECT_STRNE(<i>str1</i> , <i>str2</i>);	A két C-stílusú karakterlánc tartalma eltérő
ASSERT_STRCASEEQ(<i>str1</i> , <i>str2</i>);	EXPECT_STRCASEEQ(<i>str1</i> , <i>str2</i>);	A két C-stílusú karakterlánc tartalma a kis- és nagybetűk eltérésétől eltekintve azonos
ASSERT_STRCASENE(<i>str1</i> , <i>str2</i>);	EXPECT_STRCASENE(<i>str1</i> , <i>str2</i>);	A két C-stílusú karakterlánc tartalma a kis- és nagybetűk eltérését figyelmen kívül hagyva is eltérő

Javítsuk a tesztet és készítsünk további, hasonló tagfüggvényt (tesztekkel)!

10/matrix10test.cpp (10/CMakeLists.txt)

```
76 TEST(MulTest, print) {
77     std::vector<std::vector<double>> right;
78     right.resize(2, std::vector<double>(2, 1.));
79     szeMatrix::Matrix<double> m2(right);
80     const char* expected = "1\t1\t\n1\t1\t\n";
81     testing::internal::CaptureStdout();
82     m2.print();
83     std::string output = testing::internal::GetCapturedStdout();
84     //ASSERT_EQ(expected, output.c_str());
85     ASSERT_STREQ(expected, output.c_str());
86 }
```

10/matrix10.h

```
3  #include <sstream>

7  class Matrix {

11     public:

16         void print();
17         std::string toString();
18         const char* toCString();

24 };
```

10/matrix10.h

```

36 template<class T>
37 std::string Matrix<T>::toString() {
38     std::stringstream ss;
39     for(std::vector<T> row : mtx) {
40         for(T elem : row) {
41             ss << elem << '\t';
42         }
43         ss << std::endl;
44     }
45     return ss.str();
46 }
47
48 template<class T>
49 const char* Matrix<T>::toCString() {
50     return toString().c_str();
51 }

```

10/matrix10test.cpp

```

88 TEST(MulTest, toString) {
89     std::vector<std::vector<double>> right;
90     right.resize(2, std::vector<double>(2, 1.));
91     szMatrix::Matrix<double> m2(right);
92     std::string expected = "1\t1\t\n1\t1\t\n";
93     ASSERT_EQ(expected, m2.toString());
94 }
95
96 TEST(MulTest, toCString) {
97     std::vector<std::vector<double>> right;
98     right.resize(2, std::vector<double>(2, 1.));
99     szMatrix::Matrix<double> m2(right);
100     const char* expected = "1\t1\t\n1\t1\t\n";
101     ASSERT_STREQ(expected, m2.toCString());
102 }

```


Vegyük észre, hogy a tesztünkben egyre többször ismétlődnek részek:

10/matrix10test.cpp

```

76 TEST(MulTest, print) {
77     std::vector<std::vector<double>> right;
78     right.resize(2, std::vector<double>(2, 1.));
79     szMatrix::Matrix<double> m2(right);
80     const char* expected = "1\t1\t\n1\t1\t\n";

88 TEST(MulTest, toString) {
89     std::vector<std::vector<double>> right;
90     right.resize(2, std::vector<double>(2, 1.));
91     szMatrix::Matrix<double> m2(right);
92     std::string expected = "1\t1\t\n1\t1\t\n";

96 TEST(MulTest, toCString) {
97     std::vector<std::vector<double>> right;
98     right.resize(2, std::vector<double>(2, 1.));
99     szMatrix::Matrix<double> m2(right);
100    const char* expected = "1\t1\t\n1\t1\t\n";

```

Megoldás: teszt **fixture**-ök (\approx alkatrész) használata

- 1 Származtassunk le egy osztályt a `::testing::Test`-ből! Ha az `Osztaly`-t szeretnénk tesztelni, legyen a neve `OsztalyTest`!
- 2 Deklaráljuk a többször használt tagokat! Legyenek védettek, hogy a leszármazottakból is használhatók legyenek!
- 3 A tagokat inicializáljuk az alapértelmezett konstruktorban vagy a (felüldefiniált) `SetUp()` tagfüggvényben!
- 4 Ha szükséges, készítsünk destruktort vagy (felüldefiniált) `TearDown()` tagfüggvényt az erőforrások felszabadítására!
- 5 Ha szükséges, írjunk függvényeket, amiket több teszt eset is hívhat!

- 6 A tesztesetek definiálásakor a TEST helyett használjuk a TEST_F makrót!
- 7 A tesztkészlet neve egyezzen meg a fixture osztály nevével (OsztalyTest)!

Megjegyzések

- Az osztálynak már a tesztesetek makrói előtt definiálnak kell lennie.
- Könnyű elgépelni a `SetUp()` és `TearDown()` függvények neveit, használjuk az `override` kulcsszót (C++11)!
- Minden egyes tesztesethez új példány készül a fixture-ből (nem „interferálnak” a tesztesetek), majd:
alapértelmezett konstruktor → `SetUp()` → `TEST_F` → `TearDown()` → destruktork.

Mikor és miért érdemes konstruktort/destruktort használni?

- A `const` minősítővel ellátott tagváltozó csak a konstruktort követő inicializátor listával inicializálható. Jó ötlet a véletlen módosítások meggátolására.
- Ha a `fixture` osztályból származtatunk, az ős(ök) konstruktorának/destruktorának hívása mindenképpen végbemegy a megfelelő sorrendben. A `SetUp()/TearDown()` esetében erre a programozónak kell ügyelnie.

Mikor és miért érdemes a SetUp()/TearDown() függvényeket használni?

- A C++ nem engedi meg virtuális függvények hívását a konstruktorokban és destruktorokban, mert elvileg így meghívható lehetne egy inicializálatlan objektum metódusa, és ezt túl körülményes ellenőrizni. (Ha megengedi, akkor is csak az aktuális objektum metódusát hívja.)
- A konstruktorban/destruktorban nem használhatóak az `ASSERT_*` makrók.
Megoldás:
 - 1 `SetUp()/TearDown()` használata
 - 2 Az egész tesztprogramot állítjuk le egy `abort()` hívással.
- Ha a leállási folyamat során kivételek keletkezhetnek, azt a destruktorban nem lehet megbízhatóan lekezelni (definiálatlan viselkedés, akár azonnali programleállással).

```
11/matrix11test.cpp (11/CMakeLists.txt, 11/matrix11.h)
```

```

6  class MatrixTest : public ::testing::Test {
7      protected:
8          sizeMatrix::Matrix<double>* mtx2by2;
9          const char* expectedStr = "1\t1\t\n1\t1\t\n";
10         void SetUp() override {
11             std::vector<std::vector<double>> vec2by2;
12             vec2by2.resize(2, std::vector<double>(2, 1.));
13             mtx2by2 = new sizeMatrix::Matrix<double>(vec2by2);
14         }
15         void TearDown() override {
16             delete mtx2by2;
17         }
18     };

```

11/matrix11test.cpp

```

90 TEST_F(MatrixTest, print) {
91     testing::internal::CaptureStdout();
92     mtx2by2->print();
93     std::string output = testing::internal::GetCapturedStdout();
94     ASSERT_STREQ(expectedStr, output.c_str());
95 }
96
97 TEST_F(MatrixTest, toString) {
98     std::string expected = expectedStr;
99     ASSERT_EQ(expected, mtx2by2->toString());
100 }
101
102 TEST_F(MatrixTest, toCString) {
103     ASSERT_STREQ(expectedStr, mtx2by2->toCString());
104 }

```

Kimenet

```
wajzy@lenovo:~/Dokumentumok/gknb_intm006/GKxB_INTM006/11$ ./runTests
```

```
[=====] Running 6 tests from 2 test cases.
```

```
[-----] Global test environment set-up.
```

```
[-----] 3 tests from MulTest
```

```
[ RUN      ] MulTest.meaningful
```

```
[ OK ] MultTest.meaningful (0 ms)
```

```
[ RUN      ] MulTest.equality
```

```
[ OK ] MulTest.equality (0 ms)
```

```
[ RUN      ] MultTest.rounding
```

```
[ OK ] MulTest.rounding (0 ms)
```

```
[-----] 3 tests from MulTest (0 ms total)
```

```
[-----] 3 tests from MatrixTest
```

```
[ RUN      ] MatrixTest.print
```

```
[ OK ] MatrixTest.print (0 ms)
```

```
[ RUN      ] MatrixTest.toString
```

```
[ OK ] MatrixTest.toString (0 ms)
```

```
[ RUN      ] MatrixTest.toCString
```

```
[ OK ] MatrixTest.toCString (0 ms)
```

```
[-----] 3 tests from MatrixTest (0 ms total)
```

```
[-----] Global test environment tear-down
```

```
[=====] 6 tests from 2 test cases ran. (1 ms total)
```

```
[ PASSED ] 6 tests.
```


Egészítsük ki a `Matrix` osztályt olyan konstruktorral, ami egy `rows` sorból és `cols` oszlopból álló mátrixot véletlenszerűen feltölt `min` és `max` közé eső értékekkel!

12/matrix12.h (12/CMakeLists.txt)

```
8  template<class T>
9  class Matrix {
13     public:
14         Matrix(int rows, int cols, T min, T max);
27 };

```

12/matrix12.h

```

29 template<class T>
30 Matrix<T>::Matrix(int rows, int cols, T min, T max) {
31     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
32     std::mt19937 rng(seed);
33     std::uniform_int_distribution<uint32_t> dist;
34     mtx.resize(rows, std::vector<T>(cols));
35     for(int r=0; r<rows; r++) {
36         for(int c=0; c<cols; c++) {
37             mtx[r][c] = 0.2 + min+(T)dist(rng)/rng.max()*(max-min); // BAD
38             // mtx[r][c] = min+(T)dist(rng)/rng.max()*(max-min); // GOOD
39         }
40     }
41 }

```

A **BAD** sor kizárólag tesztelési célokat szolgál, hogy néha intervallumon kívüli értékek kerüljenek a mátrixba.

12/matrix12test.cpp

```

90 TEST(MulTest, randomized) {
91     int rows = 2;
92     int cols = 3;
93     double min = -3.;
94     double max = +3.;
95     szMatrix::Matrix<double> mtxRnd(rows, cols, min, max);
96     ASSERT_EQ(rows, mtxRnd.getRowCount());
97     ASSERT_EQ(cols, mtxRnd.getColCount());
98     for(unsigned r=0; r<rows; r++) {
99         for(unsigned c=0; c<cols; c++) {
100             double val = mtxRnd.get(r, c);
101             EXPECT_GE(max, val);
102             EXPECT_LE(min, val);
103         }
104     }
105 }

```

- Tesztkészlet N-szeri ismétlése. Negatív értékre az örökkévalóságig ismétél.
--gtest_repeat=N
- Leállítás az első olyan tesztkészlet iterációnál, ami hibát talált. Debuggerből futtatva a teszteket a memória tartalma ellenőrizhető.
--gtest_break_on_failure
- Tesztesetek szűrése: csak akkor fut le egy teszteset, ha létezik olyan pozitív, de nem létezik olyan negatív minta, amire illeszkedik. A negatív minták elhagyhatóak. A pozitív mintákat a negatívaktól - választja el. A * tetszőleges karakterláncra illeszkedik, a ? egy tetszőleges karaktert helyettesít.
--gtest_filter=poz1:poz2:...:pozN-neg1:neg2:...:negN
- Tesztkészletek és -esetek listázása
--gtest_list_tests

Egyes beállítások környezeti változókon keresztül is módosíthatóak.

Milyen teszteseteink vannak?

```
./runTests --gtest_list_tests
```

MulTest.

meaningful

equality

rounding

randomized

MatrixTest.

```
print
```

toString

toCString

- Minden tesztkészlet összes tesztesetének futtatása
`./runTests`
`./runTests --gtest_filter=*`
- Csak a MulTest tesztkészlet futtatása
`./runTests --gtest_filter=MulTest.*`
- Az összes **r** betűt tartalmazó teszt futtatása, kivéve a **String**-et tartalmazókat és `MulTest.rounding`-ot, azaz `randomized` és `print` futtatása
`./runTests --gtest_filter=*r*-*String:MulTest.rounding`
- Csak a `randomized` futtatása 100-szor
`./runTests --gtest_filter=MulTest.randomized --gtest_repeat=100`

- Teszteredmények fájlba mentése. Tesztismétlés esetén csak az utolsó iteráció eredményét tartalmazza. Alapértelmezett kimenet: `test_detail.xml` Ha **kimenet** egy mappa, mindig új nevet választ a felülírás elkerülésére.

```
--gtest_output=xml<:kimenet>
```

```
Pl. ./runTests --gtest_filter=MulTest.randomized --gtest_output=xml:egysegteszt.xml
```

12/egysegteszt.xml

```
-<testsuites tests="7" failures="1" disabled="0" errors="0" time="0.001" name="AllTests">
-  <testsuite name="MulTest" tests="4" failures="1" disabled="0" errors="0" time="0">
-    <testcase name="randomized" status="run" time="0" classname="MulTest">
-      <failure message="Expected: (max) >= (val), actual: 3 vs 3.10317" type="">
        /home/wajzy/Dokumentumok/gknb_intm006/GKxB_INTM006/12/matrix12test.cpp:101 Expected: (max) >= (val), actual: 3 vs 3.10317
      </failure>
    </testcase>
  </testsuite>
</testsuites>
```

- Az XML megjeleníthető különféle eszközökkel, pl. [Jenkins/xUnit](#)-tal

Egészítsük ki a konstruktort kivétel dobásával, ha az eredeti vektor sorai nem azonos elemszámúak!

13/matrix13.h (13/CMakeLists.txt)

```
6  #include<stdexcept>
7  namespace szeMatrix {
8
9  template<class T>
10 class Matrix {
11     protected:
12         std::vector<std::vector<T>> mtx;
13
14     public:
15         Matrix(int rows, int cols, T min, T max);
16         Matrix(std::vector<std::vector<T>> src);
17
26 };
```


13/matrix13.h

```
41 template<class T>
42 Matrix<T>::Matrix(std::vector<std::vector<T>> src) {
43     bool firstRow = true;
44     int numCols;
45     for(std::vector<T> row : src) {
46         if(firstRow) {
47             numCols = row.size();
48             firstRow = false;
49         } else {
50             if(numCols != row.size()) {
51                 throw std::range_error("Row lengths are different.");
52             }
53         }
54         mtx.push_back(row);
55     }
56 }
```

Módosítsuk és egészítsük ki tesztünket!

13/matrix13test.cpp

```
20 TEST(MulTest, meaningful) {
21     std::vector<std::vector<int>> left = {
22         {11, 12, 13, 14},
23         {21, 22, 23, 24},
24         {31, 32, 33, 34}
25     };
26     std::vector<std::vector<int>> right;
27     right.resize(4, std::vector<int>(3, 1.));
28     std::vector<std::vector<int>> expected = {
29         {50, 50, 50},
30         {90, 90, 90},
31         {130, 130, 130}
32     };
```

13/matrix13test.cpp

```

33 ASSERT_NO_THROW({
34     sizeMatrix::Matrix<int> m1(left);
35     sizeMatrix::Matrix<int> m2(right);
36     sizeMatrix::Matrix<int> multiplied = m1.mul(m2);
37     ASSERT_EQ(expected.size(), multiplied.getRowCount());
38     ASSERT_EQ(expected[0].size(), multiplied.getColCount());
39     for(unsigned row=0; row<expected.size(); row++) {
40         for(unsigned col=0; col<expected[row].size(); col++) {
41             EXPECT_EQ(expected[row][col], multiplied.get(row, col));
42         }
43     }
44 });
45 }

```

13/matrix13test.cpp

```
47 TEST(MulTest, diffRowLengths) {
48     std::vector<std::vector<int>> invalid = {
49         {11},
50         {21, 22},
51         {31, 32, 33}
52     };
53     ASSERT_THROW(szeMatrix::Matrix<int> re(invalid),
54         std::range_error);
55 }
```

Kivételek kiváltásával szemben támasztható követelmények

Végzetes hibákhoz	Nem végzetes hibákhoz	Követelmény
ASSERT_THROW(<i>statement</i> , <i>exception_type</i>);	EXPECT_THROW(<i>statement</i> , <i>exception_type</i>);	<i>statement</i> hatására <i>exception_type</i> kivételnek kell keletkeznie
ASSERT_ANY_THROW(<i>statement</i>);	EXPECT_ANY_THROW(<i>statement</i>);	<i>statement</i> hatására valamilyen kivételnek kell keletkeznie
ASSERT_NO_THROW(<i>statement</i>);	EXPECT_NO_THROW(<i>statement</i>);	<i>statement</i> hatására semmilyen kivételnek sem szabad keletkeznie

A haláltesztek (Death Tests) azt ellenőrzik, hogy valamilyen körülmény hatására a program leáll-e. Egészítsük ki a konstruktort úgy, hogy negatív sor- vagy oszlopszám esetén 1 hibakóddal álljon le a program!

14/matrix14.h (14/CMakeLists.txt)

```
28 template<class T>
29 Matrix<T>::Matrix(int rows, int cols, T min, T max) {
30     if(rows<0 or cols<0) {
31         std::cerr << "Row and column numbers must be non-negative.";
32         exit(1);
33     }
```

Ellenőrizzük, hogy a program valóban leáll-e az elvárt módon!

14/matrix14test.cpp

```
119 TEST(MatrixDeathTest, constructor) {
120     ASSERT_EXIT(szeMatrix::Matrix<double> mtxRnd(-1, 2, 1., 2.);,
121                 ::testing::ExitedWithCode(1),
122                 "Row and column numbers must be non-negative.");
123 }
```

Halálteszteket támogató makrók

Végzetes hibákhoz	Nem végzetes hibákhoz	Követelmény
ASSERT_DEATH(<i>statement</i> , <i>matcher</i>);	EXPECT_DEATH(<i>statement</i> , <i>matcher</i>);	<i>statement</i> programleállást idéz elő <i>matcher</i> üzenettel
ASSERT_DEATH_IF_SUPPORTED(<i>statement</i> , <i>matcher</i>);	EXPECT_DEATH_IF_SUPPORTED(<i>statement</i> , <i>matcher</i>);	Csak akkor ellenőrzi, hogy <i>statement</i> programleállást idéz-e elő <i>matcher</i> üzenettel, ha a haláltesztek támogatottak
ASSERT_EXIT(<i>statement</i> , <i>predicate</i> , <i>matcher</i>);	EXPECT_EXIT(<i>statement</i> , <i>predicate</i> , <i>matcher</i>);	<i>statement</i> programleállást idéz elő <i>matcher</i> üzenettel, a kilépési kódot <i>matcher</i> -re állítja

Paraméterezés:

statement

A programleálláshoz vezető (egyszerű vagy összetett) utasítás.

predicate

Függvény vagy függvény objektum, ami `int` paramétert vár és `bool`-t szolgáltat:

- ```
■ ::testing::ExitedWithCode(exit_code)
```

Az elvárt kilépési kódot ellenőrzi.

- ```
■ ::testing::KilledBySignal(signal_number)
```

Ellenőrzi, hogy a programot az elvárt jelzés szakította-e félbe (Windows-on nem támogatott).

Paraméterezés folyt.:

matcher

A szabvány hibacsatornára írt, elvárt üzenet. Ellenőrizhető:

- 1 GMock illesztővel (*const std::string&*-t illeszt)
- 2 Perl-kompatibilis reguláris kifejezéssel (A „csupasz” karakterláncokat `ContainsRegex(str)`-rel értékeli ki.)

Megjegyzések

- A 0 kilépési kóddal leálló programot nem tekintik „halott” programnak. A leállítást általában `abort()`, `exit()` hívással vagy egy jelzéssel történik.
- A haláltesztek készletének neve `DeathTest`-re kell, hogy végződjön (részletek). Száلبiztos környezet szükséges lehet.

Tesztelésről általában

Ficsor Lajos, Kovács László, Kúspér Gábor, Krizsán Zoltán: Szoftvertesztelés
ISTQB CTFL Syllabus 2018
Szakkifejezések kereshető gyűjteménye

googletest

Hivatalos Google tutorial, bevezető

Hivatalos Google tutorial, fejlett technikák

googletest FAQ

Ubuntu-specifikus részletek

IBM tananyag a googletest-hez