

# C++ programok egységtesztelése googletest segítségével (GKxB\_INTM006)

Dr. Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

[https://github.com/wajzy/GKxB\\_INTM006.git](https://github.com/wajzy/GKxB_INTM006.git)

2021. október 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.

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

Egységteszt (unit test): a metódusok struktúra tesztje.



## Kik végzik a tesztelést?

### 1-3 Fejlesztő cég

## 4 Felhasználók

## Komponentenstest

- 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





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

## Részletesen megvizsgáljuk: googletest



## 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.



```
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

```
31 template<class T>
32 Matrix<T> Matrix<T>::mul(Matrix<T> right) const {
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

```
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)
```









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
```

03/matrix03.h (03/CMakeLists.txt)

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

## 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 }

```







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>





## 04/matrix04test.cpp

```
42     sizeMatrix::Matrix<double> m1(left);
43     sizeMatrix::Matrix<double> m2(right);
44     sizeMatrix::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.

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     }
```





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

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

# 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.
```





```
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.



# 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.
```



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

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 }
```

```
...
[ 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!



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

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ≡ ≡ ↺ 🔍 ↻

10/matrix10.h

```
3 #include <sstream>
```

```
7 class Matrix {
```

```
11 public:
```

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

24 } ;

10/matrix10.h

```

36 template<class T>
37 std::string Matrix<T>::toString() const {
38     std::stringstream ss;
39     for(auto row : mtx) {
40         for(auto 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() const {
50     return toString().c_str();
51 }

```





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";

```











## 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      ] MulTest.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(int r=0; r<rows; r++) {
99         for(int 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.







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

```
6 #include<stdexcept>
7 namespace szMatrix {
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
18 };
19
20 }
```



## 13/matrix13.h

```

41 template<class T>
42 Matrix<T>::Matrix( std::vector<std::vector<T>> src ) {
43     bool firstRow = true;
44     unsigned numCols;
45     for( auto 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 }

```

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 kivétel objektumban tárolt adatok (üzenet, egyedi hibakód, stb.) ezzel a módszerrel nem ellenőrizhetők.

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     }
```



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>predicate</i> -nek megfelelőre 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).

*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.)

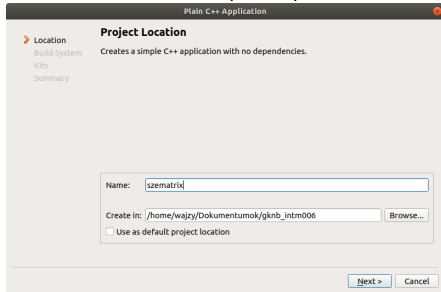
## Megjegyzések

- 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.



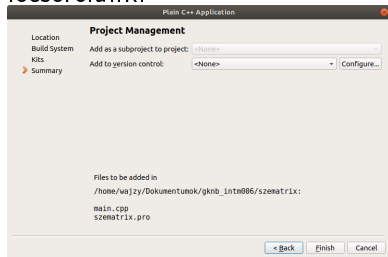


- 4 A dialógusablakban adjuk meg a projekt nevét a *Name* mezőben (szematrix), és válasszunk egy mappát, ahová a projektet elhelyezhetjük! Végül lépünk a következő oldalra (*Next*)!



- 6** A készleteknél (*Kit Selection*) hagyjunk mindent változatlanul!

A dialógusablak alján látszik, hogy létrejön a *qmake* projektleíró fájlja (*szematrix.pro*) illetve egy helykitöltő forrásszöveg (*main.cpp*), amit hamarosan lecserélünk.



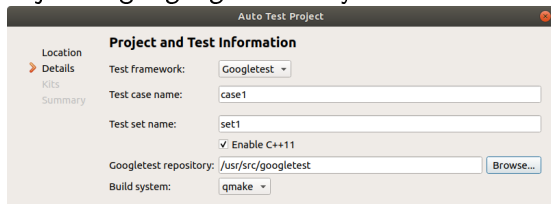
- ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ▶ ↺ 🔍 ↻





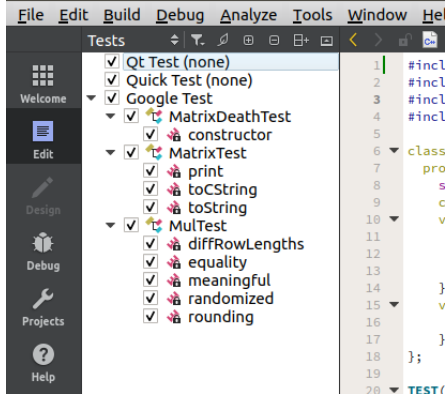
- 13** Készítsük el a teszt projektet! A *File* → *New File or Project...* menüvel megnyíló ablakban most válasszuk az *Other Project*-et, majd az *Auto Test Project*-et!
- 14** A megnyíló dialógusablak *Name* mezőjébe gépeljük az új projekt nevét: `matrixteszt`!

**15** A következő fülön kell kiválasztani a tesztelés eszközt, ami legyen *Googletest*! Bár most erre semmi szükségünk, kénytelenek vagyunk megadni egy helykitöltő teszteteset-nevet (*Test case name*, pl. case1) és tesztkészlet-nevet (*Test set name*, pl. set1). Engedélyezzük a C++11-kompatibilis fordítást a jelölőnégyzettel, majd adjuk meg a *googletest* könyvtárát! Őrizzük meg a *qmake* összeállító rendszert!

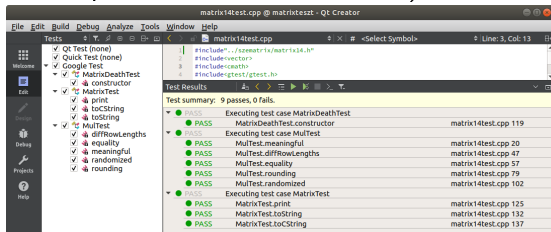


- ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

- Futtatandó tesztesetek kiválasztása: *Tests* nézetben.



- Kiválasztott/összes teszteset futtatása: *Test Results* kimeneti ablaktábla (*Window* → *Output Panes* → *Test Results*) zöld háromszögeivel



Széchenyi István Egyetem, Győr



## Tesztelésről általában

Ficsor Lajos, Kovács László, Kuser 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



További szakmai anyagok, érdeklődőknek

## Tesztelési módszerek

### Statikus kódellenőrzés

#### V&V általában