C++ programok egységtesztelése googletest segítségével (GKxB INTM006)

Dr. Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

https://github.com/wajzy/GKxB_INTM006.git 2019. augusztus 30.

Tesztelés célja: a hibákat megtalálni üzembe helyezés előtt Tesztelés alapelvei

- 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)
- Hibák csoportosulása (azokra a modulokra/bemenetekre kell tesztelni, amelyre 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.
- osztálvok.
- funkciók.
- modulok.

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

Egységteszt (unit test): a metódusok struktúra tesztie.



A tesztelés szintjei:

- 1 komponensteszt (egy komponens tesztelése)
 - a egységteszt
 - 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 Feilesztő cég
 - 4 Felhasználók

Komponensteszt

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



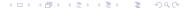
Integrációs teszt

- 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



- követelmény specifikációnak,
- funkcionális specifikációnak,
- rendszerterynek

Gyakran fekete dobozos, külső cég végzi (elfogulatlanság) Leendő futtatási környezet imitációja



- 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
- \blacksquare struktúrálható tesztek (teszt program \rightarrow teszt csomag \rightarrow teszteset)
- informatív
- leveszi a tesztelés technikai részének terhét a tesztelőről
- gvors (megosztott erőforrások)
- könnyen tanulható (xUnit architektúra)



Bevezetés

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.



Feladat

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

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



Az első teszprogram elkészítése

01/matrix01.h

```
#include < vector >
   #include < iostream >
   namespace szeMatrix {
4
   template < class T>
   class Matrix {
      protected:
        std::vector<std::vector<T>> mtx:
9
10
      public:
11
        Matrix(std::vector<std::vector<T>>> src) {
12
          mtx = src;
13
```

googletest

```
01/matrix01.h
   template < class T>
21
22
   void Matrix<T>::print() const {
23
      for(auto row : mtx) {
        for(auto elem : row) {
24
          std::cout << elem << '\t';
25
26
27
        std::cout << std::endl:
28
29
```

Az első teszprogram elkészítése

01/matrix01.h

```
template < class T>
31
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
      res.resize(i, std::vector\langle T \rangle(i, 0.));
43
```

01/matrix01.h

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

Az első teszprogram elkészítése

01/example01.cpp

```
#include < vector >
   #include"matrix01.h"
3
   int main() {
     std::vector < std::vector < int >> v1 = {
        {11, 12, 13, 14},
6
       {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

```
szeMatrix::Matrix<int> m1(v1);
szeMatrix::Matrix<int> m2(v2);
szeMatrix::Matrix<int> multiplied = m1.mul(m2);
multiplied.print();

return 0;
}
```

Kimenet

50	50	50
90	90	90
130	130	130

200

Készítsünk az example01.cpp alapján googletest alapú tesztprogramot!

01/matrix01test.cpp

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

Bevezetés

01/matrix01test.cpp



Az első teszprogram elkészítése

01/matrix01test.cpp

```
ASSERT EQ(expected.size(), multiplied.getRowCount());
21
     ASSERT EQ(expected[0].size(), multiplied.getColCount());
22
23
      for (unsigned row=0; row<expected.size(); row++) {</pre>
        for (unsigned col=0; col<expected[row]. size(); col++) {
24
          EXPECT EQ(expected[row][col], multiplied.get(row, col));
25
26
27
28
29
30
   int main(int argc, char **argv) {
        ::testing::InitGoogleTest(&argc, argv);
31
32
        return RUN ALL TESTS();
33
```

01/CMakeLists.txt

```
cmake minimum required (VERSION 2.6)
   # Locate GTest
   find package (GTest REQUIRED)
16
    include directories(${GTEST INCLUDE DIRS})
17
18
   # Link runTests with what we want to test
   # and the GTest and pthread library
   add executable (runTests matrix01test.cpp)
20
   target | ink | ibraries (run Tests $ { GTEST LIBRARIES } pthread)
21
```

Az első teszprogram elkészítése

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
\lceil = = = = = = \rceil 1 test from 1 test case ran. (0 ms total)
[ PASSED ] 1 test.
```

Bevezetés

Teszteset (test case)

"A set of preconditions, inputs, actions (where applicable), expected results and postconditions, developed based on test conditions." (meaningful, ld. matrix01test.cpp 5. sor)

Tesztkészlet (test suite)

"A set of test cases or test procedures to be executed in a specific test cycle." (MulTest, ld. matrix01test.cpp 5. sor)

Tesztprogram (test program)

Egy vagy több tesztkészletet foglal magába.

Sainos a googletest nevezéktana következetlen:

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



Assertion (≈ állítás, követelés) Ellenőrizzük valamely elvárásunk teljesülését → 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 tesztesetet (nincs értelme a folytatásnak; pl. ha két mátrix nem azonos méretű, nincs értelme az elemeiket összehasonlítgatni). 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
        for (int c=0: c<i: c++) {
46
          for(int item=0: item<k: item++) {</pre>
47
             // res[r][c] += mtx[r][item]*right.mtx[item][c];
48
49
50
51
```

Kimenet

```
[======] Running 1 test from 1 test case.
[-----] Global test environment set-up.
[-----] 1 test from MulTest
\Gamma 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
. . .
```

Az első teszprogram elkészítése

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 1 0 tests
  FAILED ] 1 test, listed below:
  FAILED
         ] MulTest.meaningful
```

1 FAILED TEST



Az első teszprogram elkészítése

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

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

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

03/matrix03test.cpp

```
ASSERT EQ(expected.size(), multiplied.getRowCount())
21
       << "A sorok szama elter! Elvart: " << expected.size()</pre>
22
23
       << ", kapott: " << multiplied.getRowCount();</pre>
      ASSERT EQ(expected [0]. size(), multiplied.getColCount())
24
        << "Az oszlopok szama elter! Elvart: " << expected[0]. size()</pre>
25
26
       << ". kapott: " << multiplied.getColCount();</pre>
      for(unsigned row=0; row<expected.size(); row++) {</pre>
27
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 << "]["</pre>
31
            << col << "] helven!":
32
33
```

Kimenet

```
[=======] Running 1 test from 1 test case.
[----] Global test environment set-up.
[-----] 1 test from MulTest
Γ RIIN
          ] 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 tesztesetet.



Végzetes hibákhoz Nem végzetes hibákhoz Követelménye ASSERT_TRUE(feltétel) EXPECT_TRUE(feltétel) feltétel igaz értékű EXPECT_FALSE(feltétel) feltétel hamis értékű

Relációs követelmények

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



Megjegyzések

- A feltüntetett operátoroknak definiáltnak kell lenniük val1 és val2 között. Lehetőségeink:
 - Felültöltjük az operátorokat.
 - 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.



Lebegőpontos számok kezelése

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;
      right resize (2, std :: vector < double > (2, 1.)):
37
38
      std::vector<std::vector<double>> expected = {
39
        {1.414213562, 1.414213562},
        {0.333333333 0.3333333333}
40
41
```

04/matrix04test.cpp

```
42
      szeMatrix :: Matrix < double > m1(left);
43
      szeMatrix :: Matrix < double > m2(right);
      szeMatrix :: Matrix < double > multiplied = m1.mul(m2) :
44
     ASSERT EQ(expected.size(), multiplied.getRowCount());
45
     ASSERT EQ(expected [0]. size(), multiplied.getColCount());
46
      for(unsigned row=0; row<expected.size(); row++) {</pre>
48
        for(unsigned col=0; col<expected[row].size(); col++) {</pre>
          EXPECT EQ(expected[row][col], multiplied.get(row, col));
49
50
52
```

googletest

Lebegőpontos számok kezelése

Kimenet

```
Γ 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)
  for (unsigned row=0; row<expected.size(); row++) {</pre>
    for (unsigned col=0; col < expected [row]. size (); <math>col++) {
       //EXPECT EQ(expected[row][col], multiplied.get(row, col));
      EXPECT DOUBLE EQ(expected [row][col], multiplied.get(row, col));
```

47

48

49

50 51 52

Kimenet

```
. . .
           ] MulTest.rounding
[ RUN
/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: 0.3333333333333333333
Expected: expected[row][col]
Which is: 0.33333333300000001
[ FAILED ] MulTest.rounding (0 ms)
. . .
```

Most már látszik, hogy az értékek közötti különbség nagyobb, mint 4 ULP (Units in the

Dr. Hatwágner F. Miklós Széchenyi István Egyetem, Győr

47

48

49

50

51

52 53 Növeliük meg a számok közötti legnagyobb megengedett eltérést!

```
06/matrix06test.cpp (06/matrix06.h, 06/CMakeLists.txt)
  for (unsigned row=0; row<expected.size(); row++) {</pre>
    for (unsigned col=0; col < expected [row]. size (); col++) {
      //EXPECT EQ(expected[row][col], multiplied.get(row, col));
      //EXPECT DOUBLE EQ(expected [row][col], multiplied.get(row, col));
      EXPECT NEAR(expected [row][col], multiplied get(row, col), 1e-9);
```

Kimenet

```
[=======] Running 2 tests from 1 test case.
         -] Global test environment set-up.
         -1 2 tests from MulTest
[ RUN
          ] MulTest.meaningful
       OK ] MulTest.meaningful (0 ms)
[ RUN
          ] MulTest.rounding
       OK ] MulTest.rounding (0 ms)
          -1 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(val1, val2);	EXPECT_FLOAT_EQ(val1, val2);	float típusú értékek 4 ULP- n belül
ASSERT_DOUBLE_EQ(<i>val</i> 1, <i>val</i> 2);	EXPECT_DOUBLE_EQ(val1, val2);	<i>double</i> típusú értékek 4 ULP-n belül
ASSERT_NEAR(val1, val2, abs_error);	EXPECT_NEAR(val1, val2, abs_error);	a két érték különbségének abszolút értéke nem na- gyobb <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>> |eft = {
33
        {11, 12, 13, 14},
34
        {21, 22, 23, 24}.
35
        {31, 32, 33, 34}
36
37
      std::vector<std::vector<double>> right:
      right resize (4, std :: vector < double > (3, 1));
38
      std::vector<std::vector<double>> expected = {
39
        {50. 50. 50}.
40
41
        {90, 90, 90}.
        {130. 130. 130}
42
43
```

Bevezetés

07/matrix07test.cpp

```
szeMatrix::Matrix < double > m1(left);
szeMatrix::Matrix < double > m2(right);
szeMatrix::Matrix < double > mexp(expected);
szeMatrix::Matrix < double > multiplied = m1.mul(m2);
ASSERT_EQ(mexp.getRowCount(), multiplied.getRowCount());
ASSERT_EQ(mexp.getColCount(), multiplied.getColCount());
ASSERT_EQ(mexp, multiplied);
}
```

Kimenet

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)

```
template < class T>
 6
    class Matrix {
10
      public:
        template < class U>
19
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

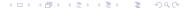
Operátor <u>felültöltés</u>

```
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
ΓRUN
          ] MulTest.meaningful
       OK ] MulTest.meaningful (0 ms)
Γ RUN
          ] MulTest.equality
       OK ] MulTest.equality (1 ms)
          ] MulTest.rounding
[ R.U.N
       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()	Megkezdi az stdout-ra írt tartalom rögzítését
${ t GetCapturedStdout()}$	Lekérdezi a rögzített tartalmat és leállítja a rögzítést
CaptureStderr()	Megkezdi az stderr-re írt tartalom rögzítését
<pre>GetCapturedStderr()</pre>	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) {
      std::vector<std::vector<double>> right;
77
      right resize (2, std :: vector < double > (2, 1.));
78
      szeMatrix :: Matrix < double > m2(right);
79
80
     const char* expected = "1\t1\t\n1\t1\t\n";
      testing::internal::CaptureStdout();
81
82
     m2.print();
83
      std :: string output = testing :: internal :: GetCapturedStdout();
     ASSERT EQ(expected, output.c str());
84
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(str1, str2);	EXPECT_STREQ(str1, str2);	A két C-stílusú karakter-
		lánc tartalma azonos
ASSERT_STRNE(<i>str1, str2</i>);	EXPECT_STRNE(<i>str1, str2</i>);	A két C-stílusú karakter-
		lánc tartalma eltérő
ASSERT_STRCASEEQ(str1, str2);	EXPECT_STRCASEEQ(str1, str2);	A két C-stílusú karakter-
		lánc tartalma a kis- és
		nagybetűk eltérésétől elte-
		kintve azonos
ASSERT STRCASENE(str1, str2);	EXPECT STRCASENE(str1, str2);	A két C-stílusú karakter-
		lánc tartalma a kis- és
		nagybetűk eltérését figyel-
		men kívül hagyva is eltérő



Javítsuk a tesztesetet és készítsünk további, hasonló tagfüggvényeket (tesztekkel)!

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

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

```
10/matrix10.h
```

```
3 #include <sstream >
   class Matrix {
11
      public:
        void print() const;
16
        std::string toString() const;
17
18
        const char* toCString() const;
24
```

10/matrix10.h

```
36
    template < class T>
37
    std::string Matrix<T>::toString() const {
38
      std::stringstream ss;
      for(auto row : mtx) {
39
40
        for(auto elem : row) {
          ss << e|em << '\t':
41
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
```

10/matrix10test.cpp

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

Vegyük észre, hogy a tesztünkben egyre többször ismétlődnek részek:

```
10/matrix10test.cpp
```

```
TEST(MulTest, print) {
76
       std::vector<std::vector<double>> right:
77
78
       right resize (2, std::vector < double > (2, 1.));
79
       szeMatrix:: Matrix < double > m2(right):
       const char* expected = "1\t1\t1\t1\t1\t1
80
88
     TEST(MulTest, toString) {
89
       std::vector<std::vector<double>> right:
       right resize (2, std::vector < double > (2, 1.));
90
       szeMatrix:: Matrix < double > m2(right):
91
92
       96
     TEST(MulTest to CString) {
       std::vector<std::vector<double>> right:
97
98
       right resize (2. \text{ std} :: \text{vector} < \text{double} > (2. 1.)):
99
       szeMatrix:: Matrix < double > m2(right);
100
       const char* expected = "1\t1\t\n1\t1\t\n":
```

Megoldás: teszt fixture-ök (≈alkatrész) használata

- Származtassunk le egy osztályt a ::testing::Test-ből! Ha az Osztaly-t szeretnénk tesztelni, legyen a neve OsztalyTest!
- Deklaráljuk a többször használt tagokat! Legyenek védettek, hogy a leszármazottakból is használhatók legyenek!
- 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 teszteset is hívhat!



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

Megjegyzések

- Az osztálynak már a tesztesetek makrói előtt definiáltnak 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:
 - $\mathsf{alap\'ertelmezett} \ \mathsf{konstruktor} \to \mathsf{SetUp}() \to \mathsf{TEST_F} \to \mathsf{TearDown}() \to \mathsf{destruktor}.$



Fivtures

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 kelethezhetnek, azt a destruktorban nem lehet megbízhatóan lekezelni (definiálatlan viselkedés, akár azonnali programleállással).



Fixtures

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

```
class MatrixTest : public :: testing :: Test {
      protected:
 8
        szeMatrix:: Matrix < double >* mtx2by2;
        const char* expectedStr = "1\t1\t1\t1\t1\t1\t1
        void SetUp() override {
10
          std::vector<std::vector<double>> vec2by2;
11
12
          vec2by2.resize(2, std::vector < double > (2, 1.));
13
          mtx2bv2 = new szeMatrix:: Matrix < double > (vec2bv2):
14
15
        void TearDown() override {
16
          delete mtx2bv2:
17
18
    };
```

11/matrix11test.cpp

```
90
     TEST F(MatrixTest, print) {
91
       testing :: internal :: CaptureStdout();
92
       mtx2by2 \rightarrow print();
93
       std::string output = testing::internal::GetCapturedStdout();
94
      ASSERT STREQ(expectedStr. output.c str()):
95
96
97
    TEST F(MatrixTest, toString) {
       std: string expected = expectedStr;
98
      ASSERT EQ(expected, mtx2by2->toString());
99
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 ] MulTest.meaningful (0 ms)
Γ RIIN
          ] MulTest.equality
       OK ] MulTest.equality (0 ms)
[ RUN
          1 MulTest.rounding
       OK 1 MulTest.rounding (0 ms)
[----] 3 tests from MulTest (0 ms total)
[-----] 3 tests from MatrixTest
[ RUN
          ] MatrixTest.print
       OK ] MatrixTest.print (0 ms)
Γ RIIN
          ] MatrixTest.toString
       OK ] MatrixTest.toString (0 ms)
[ RUN
          1 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:
        Matrix (int rows, int cols, T min, T max);
14
```

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;
      mtx.resize(rows, std::vector<T>(cols));
34
      for (int r=0; r< rows; r++) {
35
36
        for (int c=0; c<co|s; c++) {
37
          mtx[r][c] = 0.2 + min+(T) dist(rng)/rng.max()*(max-min); // BAD
          // mtx[r][c] = min+(T) dist(rng)/rng max()*(max-min); // GOOD
38
39
40
41
```

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



12/matrix12test.cpp

```
90
     TEST(MulTest, randomized) {
91
       int rows = 2:
92
       int cols = 3:
       double min = -3:
93
94
       double max = +3;
95
       szeMatrix :: 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);
           EXPECT GE(max, val);
101
           EXPECT LE(min, val);
102
103
104
105
```

- Tesztkészlet N-szeri ismétlése. Negatív értékre az örökkévalóságig ismétel.
 - --gtest_repeat=N
- Leállás 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 fáilba 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/waizy/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)

```
#include < stdexcept >
    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):
    };
26
```

13/matrix13.h

```
41
    template < class T>
42
    Matrix<T>:: Matrix (std::vector<std::vector<T>> src) {
43
      bool firstRow = true;
      unsigned numCols;
44
      for(auto row : src) {
45
        if (firstRow) {
46
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) {
      std::vector<std::vector<int>> left = {
21
        {11, 12, 13, 14},
22
23
        {21, 22, 23, 24}.
        {31, 32, 33, 34}
24
25
26
      std::vector<std::vector<int>> right;
      right resize (4, std :: vector < int > (3, 1.));
27
28
      std::vector<std::vector<int>> expected = {
29
        {50, 50, 50}.
        {90, 90, 90}.
3.0
31
        {130, 130, 130}
32
```

13/matrix13test.cpp

```
ASSERT NO THROW({
33
        szeMatrix :: Matrix < int > m1(|eft);
34
35
        szeMatrix:: Matrix<int> m2(right);
36
        szeMatrix :: Matrix < int > multiplied = m1.mul(m2);
37
        ASSERT EQ(expected.size(), multiplied.getRowCount());
        ASSERT EQ(expected[0]. size(), multiplied.getColCount());
38
        for(unsigned row=0; row<expected.size(); row++) {</pre>
39
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) {
     std::vector<std::vector<int>> invalid = {
48
       {11},
49
       {21, 22},
50
       {31, 32, 33}
51
52
     ASSERT THROW(szeMatrix::Matrix<int> re(invalid),
53
54
        std::range error):
55
```

Kivételek és haláltesztek

Kivételek kiváltásával szemben támasztható követelmények

Végzetes hibákhoz	Nem végzetes hibákhoz	Köv et elmény
ASSERT_THROW(statement, exception_type);	EXPECT_THROW(statement, exception_type);	statement hatására ex-
		<i>ce ption_type</i> kivételnek kell keletkeznie
ASSERT_ANY_THROW(statement);	EXPECT_ANY_THROW(statement);	<i>statement</i> hatására vala- milyen kivételnek kell ke- letkeznie
ASSERT_NO_THROW(<i>statement</i>);	EXPECT_NO_THROW(statement);	statement hatására sem- milyen kivételnek sem sza- bad keletkeznie



Kivételek és haláltesztek

28

29

30

31

32

33

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!

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

Kivételek és haláltesztek

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

```
14/matrix14test.cpp
```

```
TEST(MatrixDeathTest, constructor) {

ASSERT_EXIT(szeMatrix::Matrix<double> mtxRnd(-1, 2, 1., 2.);,

::testing::ExitedWithCode(1),

"Row and column numbers must be non-negative.");

}
```

Halálteszteket támogató makrók

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



Paraméterezés folyt.:

matcher

A szabvány hibacsatornára írt, elvárt üzenet. Ellenőrizhető:

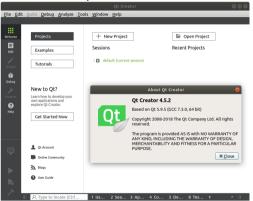
- **GMock illesztővel (const std::string&-t illeszt)**
- Perl-kompatibilis reguláris kifejezéssel (A "csupasz" karakterláncokat ContainsRegex(str)-rel értékelik ki.)

Megjegyzések

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



I Indítsuk el a Qt Creatort.





Bevezetés

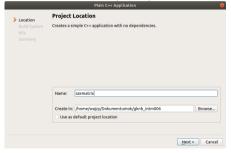
- ∠ Készítsünk új projektet az osztályunk működésének kipróbálásához (egyelőre googletest nélkül)! File → New File or Project...
- 3 A dialógusablakban jelöljük meg a *Non-Qt Project*-et majd a *Plain C++ Application*-t! Végül kattintsunk a *Choose...* gombra!

Choose a template:		All Templates -
Cheorie e tempate: Projects Application Library Other Project Import Project Import Project Import Project Modeling Qt CLSL General	Plain C Application Plain C++ Application	Creates a simple C++ application with no dependencies. Supported Platforms: Desk top
Java Python Nim		



Bevezetés

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





5 Az összeállító rendszer elvileg lehetne *cmake* is, de ezt az IDE nem támogatja teljeskörűen, ezért őrizzük meg az alapértelmezett *qmake*-et, majd lépjünk a következő oldalra!

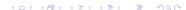


6 A készleteknél (*Kit Selection*) hagyjunk mindent változatlanul!



- A következő oldalon, az Add to version control...-nál lehetne beállítani a Git verziókezelőt, de mivel nincs hosszabb távú célunk a projekttel, nem élünk a lehetőséggel.
 - A dialógusablak alján látszik, hogy létrejön a *qmake* projektleíró fájlja (szematrix.pro) illetve egy helykitöltő forrásszöveg (main.cpp), amit hamarosan lecserélünk.

Location	Project Management		
Build System Ad	Add as a subproject to project:		
Kits Summary	Add to <u>v</u> ersion control:	<none></none>	Configure
	Files to be added in /home/wajzy/Dokumentum	ok/gknb_intm806/szematrix:	
	main.cpp szematrix.pro		



- Másoljuk a projekt mappájába a 01/example01.cpp és 14/matrix14.h fájlokat!
- 9 A Projects nézetben kattintsunk jobb gombbal a projekt nevén (szematrix), a kinyíló menüben pedig válasszuk az Add Existing Files... pontot! Adjuk hozzá a projekthez az előbb bemásolt két állományt!
- Mattintsunk jobb gombbal a main.cpp-n, majd válasszuk a Remove File... lehetőséget, azaz töröljük a generált állományt! Ne sajnáljuk a fájlt véglegesen törölni (Delete file permanently).
- Nyissuk meg az example01.cpp fájlt a nevére duplán kattintva, majd módosítsuk a második sort, hogy a matrix14.h-ra hivatkozzon!



 \blacksquare Ezután a program fordítható, futtatható ($Build \rightarrow Run$, vagy a zöld háromszögre kattintva).





- II 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!
- A megnyíló dialógusablak Name mezőjébe gépeljük az új projekt nevét: matrixteszt!



Bevezetés

IS A következő fülön kell kiválasztani a tesztelés eszközét, ami legyen *Googletest*! Bár most erre semmi szükségünk, kénytelenek vagyunk megadni egy helykitöltő teszteset-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!

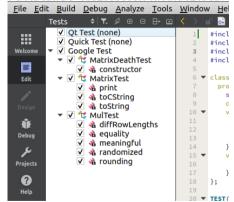
			Auto Test Project	
	Location	Project and Test	Information	
3	Details	Test framework:	Googletest 🔻	
Kits Summar		Test case name:	case1	
		Test set name:	set1	
			✓ Enable C++11	
		Googletest repository:	/usr/src/googletest	rowse
		Build system:	qmake 🔻	

Bevezetés

- 16 A Next, majd Finish gombokra kattintva létrejön, és aktívvá válik a teszt projekt.
- A projekt mappájába (matrixteszt) másoljuk be a 14/matrix14test.cpp fájlt! A Projects nézetben a projekten jobb gombbal kattintva válasszuk az Add Existing Files... pontot, és adjuk hozzá az átmásolt fájlt a projekthez!
- Nyissuk meg a fájlt a szerkesztőben, majd módosítsuk az első sort: #include"../szematrix/matrix14.h"
- 🔟 Távolítsuk el a projektből a main.cpp és tst_case1.h fájlokat (Del gomb)!
- 🔟 A tesztprogram a zöld gombbal fordítható, futtatható.



Futtatandó tesztesetek kiválasztása: Tests nézetben.





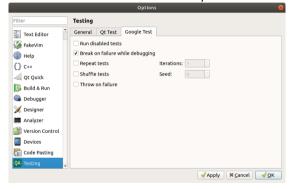
Kiválasztott/összes teszteset futtatása: Test Results kimeneti ablaktábla (Window ightarrow Output Panes ightarrow Test Results) zöld háromszögeivel





Qt Creator integráció

■ További beállítások: *Tools* → *Options...* → *Testing*



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 googletest FAQ Ubuntu-specifikus részletek IBM tananyag a googletest-hez



Tesztelési módszerek Statikus kódellenőrzés V&V általában

