CPPCON 2020 / HTTPS://BOOST-EXT.GITHUB.IO/UT/CPPCON-2020

MACRO-FREE TESTING WITH C++20

KRIS@JUSIAK.NET | @KRISJUSIAK | LINKEDIN.COM/IN/KRIS-JUSIAK

UT - https://github.com/boost-ext/ut

UT - https://github.com/boost-ext/ut

- C++20 (GCC-9+, CLANG-9.0+, APPLE CLANG-11.0.0+, MSVC-2019+*)

- (*) LIMITATIONS MAY APPLY

UT - https://github.com/boost-ext/ut

- C++20 (GCC-9+, CLANG-9.0+, APPLE CLANG-11.0.0+, MSVC-2019+*)
- SINGLE HEADER/MODULE (2K LOC) WITH NO EXTERNAL DEPENDENCIES

- (*) LIMITATIONS MAY APPLY

UT - https://github.com/boost-ext/ut

- C++20 (GCC-9+, CLANG-9.0+, APPLE CLANG-11.0.0+, MSVC-2019+*)
- SINGLE HEADER/MODULE (2K LOC) WITH NO EXTERNAL DEPENDENCIES
- MACRO-FREE (✔)

- (*) LIMITATIONS MAY APPLY

UT - https://github.com/boost-ext/ut

- C++20 (GCC-9+, CLANG-9.0+, APPLE CLANG-11.0.0+, MSVC-2019+*)
- SINGLE HEADER/MODULE (2K LOC) WITH NO EXTERNAL DEPENDENCIES
- MACRO-FREE (✔)
- FEATURES (ASSERTIONS, SUITES, TESTS, SECTIONS, BDD, MATCHERS, LOGGING, ...)

- (*) LIMITATIONS MAY APPLY

UT - https://github.com/boost-ext/ut

- C++20 (GCC-9+, CLANG-9.0+, APPLE CLANG-11.0.0+, MSVC-2019+*)
- SINGLE HEADER/MODULE (2K LOC) WITH NO EXTERNAL DEPENDENCIES
- MACRO-FREE (✔)
- FEATURES (ASSERTIONS, SUITES, TESTS, SECTIONS, BDD, MATCHERS, LOGGING, ...)

- (*) LIMITATIONS MAY APPLY
- UT IS NOT AN OFFICIAL BOOST LIBRARY

HELLO WORLD - HTTPS://GODBOLT.ORG/Z/Y43MXZ

```
1  import boost.ut;  / #include <boost/ut.hpp>
2
3  auto sum(auto... args){ return (args + ...); }
4
5  int main() {
6    using namespace boost::ut;
7
8    "sum"_test = [] {
9      sum(0) == 0_i;
10     sum(1, 2) == 3_i;
11     sum(1, 2) > 0_i and 41_i == sum(40, 2);
12   };
13 }
```

```
Running "sum"...
  sum.cpp:11:FAILED [(3 > 0 and 41 == 42)]
FAILED
================================================================================
tests:   1 | 1 failed
asserts: 3 | 2 passed | 1 failed
```

ASSERTIONS - HTTPS://GODBOLT.ORG/Z/JAFK8W

ASSERTIONS - HTTPS://GODBOLT.ORG/Z/JAFK8W

```
1_i == 2;                                   // Terse syntax
  -> assertions.cpp:1:FAILED [1 == 2]
```

ASSERTIONS - HTTPS://GODBOLT.ORG/Z/JAFK8W

```
1_i == 2;                                   // Terse syntax
  -> assertions.cpp:1:FAILED [1 == 2]

expect(2 == 1_i);                           // Expect syntax
  -> assertions.cpp:1:FAILED [2 == 1]
```

ASSERTIONS - HTTPS://GODBOLT.ORG/Z/JAFK8W

```
1_i == 2;                                   // Terse syntax
  -> assertions.cpp:1:FAILED [1 == 2]

expect(2 == 1_i);                           // Expect syntax
  -> assertions.cpp:1:FAILED [2 == 1]

expect(that % 1 == 2);                      // Matchers syntax
  -> assertions.cpp:1:FAILED [1 == 2]
```

ASSERTIONS - HTTPS://GODBOLT.ORG/Z/JAFK8W

```
1_i == 2;                                   // Terse syntax
```
```
  -> assertions.cpp:1:FAILED [1 == 2]
```

```
expect(2 == 1_i);                           // Expect syntax
```
```
  -> assertions.cpp:1:FAILED [2 == 1]
```

```
expect(that % 1 == 2);                      // Matchers syntax
```
```
  -> assertions.cpp:1:FAILED [1 == 2]
```

```
std::vector v{1l, 2l, 3l};
(4_ul == std::size(v)) >> fatal;            // Fatal assertion
v[3] == 4_l;                                // Not executed
```
```
  -> assertions.cpp:2:FAILED [4 == 3]
```

ASSERTIONS - HTTPS://GODBOLT.ORG/Z/JAFK8W

```
1_i == 2;                                   // Terse syntax
```
```
  -> assertions.cpp:1:FAILED [1 == 2]
```

```
expect(2 == 1_i);                           // Expect syntax
```
```
  -> assertions.cpp:1:FAILED [2 == 1]
```

```
expect(that % 1 == 2);                      // Matchers syntax
```
```
  -> assertions.cpp:1:FAILED [1 == 2]
```

```
std::vector v{1l, 2l, 3l};
(4_ul == std::size(v)) >> fatal;            // Fatal assertion
v[3] == 4_l;                                // Not executed
```
```
  -> assertions.cpp:2:FAILED [4 == 3]
```

```
41.10_d == 42.101 and "a" == "b"sv;     // Compound expression
                                        // with floating-point
```
```
  -> assertions.cpp:1:FAILED [42.1 == 42.101 and a == b]
```

SECTIONS - HTTPS://GODBOLT.ORG/Z/Y9M5VF

SECTIONS - HTTPS://GODBOLT.ORG/Z/Y9M5VF

```
"[vector]"_test = [] {

};
```

SECTIONS - HTTPS://GODBOLT.ORG/Z/Y9M5VF

```
"[vector]"_test = [] {
```

```
  // set up (1)
  std::vector<int> v(5);
  expect((5_ul == std::size(v)) >> fatal);
```

```
};
```

SECTIONS - HTTPS://GODBOLT.ORG/Z/Y9M5VF

```
"[vector]"_test = [] {

  // set up (1)
  std::vector<int> v(5);
  expect((5_ul == std::size(v)) >> fatal);

  should("resize bigger") = [v] { // section (2.1)
    mut(v).resize(10);
    expect(10_ul == std::size(v));
  };

};
```

SECTIONS - HTTPS://GODBOLT.ORG/Z/Y9M5VF

```
"[vector]"_test = [] {

  // set up (1)
  std::vector<int> v(5);
  expect((5_ul == std::size(v)) >> fatal);

  should("resize bigger") = [v] { // section (2.1)
    mut(v).resize(10);
    expect(10_ul == std::size(v));
  };

  expect((5_ul == std::size(v) >> fatal); // (3)

};
```

SECTIONS - HTTPS://GODBOLT.ORG/Z/Y9M5VF

```
"[vector]"_test = [] {

  // set up (1)
  std::vector<int> v(5);
  expect((5_ul == std::size(v)) >> fatal);

  should("resize bigger") = [v] { // section (2.1)
    mut(v).resize(10);
    expect(10_ul == std::size(v));
  };

  expect((5_ul == std::size(v) >> fatal); // (3)

  should("resize smaller") = [v] { // section (2.2)
    mut(v).resize(0);
    expect(0_ul == std::size(v));
  };



};
```

SECTIONS - HTTPS://GODBOLT.ORG/Z/Y9M5VF

```
"[vector]"_test = [] {

  // set up (1)
  std::vector<int> v(5);
  expect((5_ul == std::size(v)) >> fatal);

  should("resize bigger") = [v] { // section (2.1)
    mut(v).resize(10);
    expect(10_ul == std::size(v));
  };

  expect((5_ul == std::size(v) >> fatal); // (3)

  should("resize smaller") = [v] { // section (2.2)
    mut(v).resize(0);
    expect(0_ul == std::size(v));
  };

  // tear down (4)

};
```

SUITES - HTTPS://GODBOLT.ORG/Z/F3XJCJ

SUITES - HTTPS://GODBOLT.ORG/Z/F3XJCJ

```
suite errors = [] {
```

```
};
```

SUITES - HTTPS://GODBOLT.ORG/Z/F3XJCJ

```
suite errors = [] {
```

```
  "exception"_test = [] {
    expect(throws([] { throw 0; })) << "throws any exception";
  };

  "failure"_test = [] {
    expect(aborts([] { assert(false); }));
  };
```

```
};
```

SUITES - HTTPS://GODBOLT.ORG/Z/F3XJCJ

```
suite errors = [] {
```

```
  "exception"_test = [] {
    expect(throws([] { throw 0; })) << "throws any exception";
  };

  "failure"_test = [] {
    expect(aborts([] { assert(false); }));
  };
```

```
};
```

```
int main() { }
```

```
 -> All tests passed (2 asserts in 2 tests)
```

PARAMETERIZED - HTTPS://GODBOLT.ORG/Z/6FHTPQ

PARAMETERIZED - HTTPS://GODBOLT.ORG/Z/6FHTPQ

```
for (auto i : std::vector{1, 2, 3}) {
  test("args " + std::to_string(i)) = [i] {
    expect(arg > 0_i) << "all values greater than 0";
  };
}
```

PARAMETERIZED - HTTPS://GODBOLT.ORG/Z/6FHTPQ

```
for (auto i : std::vector{1, 2, 3}) {
  test("args " + std::to_string(i)) = [i] {
    expect(arg > 0_i) << "all values greater than 0";
  };
}
```

```
-> All tests passed (3 asserts in 3 tests)
```

PARAMETERIZED - HTTPS://GODBOLT.ORG/Z/6FHTPQ

```
for (auto i : std::vector{1, 2, 3}) {
  test("args " + std::to_string(i)) = [i] {
    expect(arg > 0_i) << "all values greater than 0";
  };
}
```

```
-> All tests passed (3 asserts in 3 tests)
```

```
"args and types"_test =
  []<class TArg>(TArg arg) {
    expect(std::is_integral_v<TArg>);
    expect(type<TArg> == type<int> or type<TArg> == type<bool>);
  }
```

PARAMETERIZED - HTTPS://GODBOLT.ORG/Z/6FHTPQ

```
for (auto i : std::vector{1, 2, 3}) {
  test("args " + std::to_string(i)) = [i] {
    expect(arg > 0_i) << "all values greater than 0";
  };
}
```

```
-> All tests passed (3 asserts in 3 tests)
```

```
"args and types"_test =
  []<class TArg>(TArg arg) {
    expect(std::is_integral_v<TArg>);
    expect(type<TArg> == type<int> or type<TArg> == type<bool>);
  }
```

```
| std::tuple{true, 42};
 -> All tests passed (4 asserts in 2 tests)
```

SPEC - HTTPS://GODBOLT.ORG/Z/6JKKZT

SPEC - HTTPS://GODBOLT.ORG/Z/6JKKZT

```
describe("vector") = [] {




};
```

SPEC - HTTPS://GODBOLT.ORG/Z/6JKKZT

```
describe("vector") = [] {
```

```
  std::vector<int> v(5);
  expect((5_ul == std::size(v)) >> fatal);
```

```
};
```

SPEC - HTTPS://GODBOLT.ORG/Z/6JKKZT

```
describe("vector") = [] {
```

```
  std::vector<int> v(5);
  expect((5_ul == std::size(v)) >> fatal);
```

```
  it("should resize bigger") = [v] {
    mut(v).resize(10);
    expect(10_ul == std::size(v));
  };
```

```
};
```

SPEC - HTTPS://GODBOLT.ORG/Z/6JKKZT

```
describe("vector") = [] {

  std::vector<int> v(5);
  expect((5_ul == std::size(v)) >> fatal);

  it("should resize bigger") = [v] {
    mut(v).resize(10);
    expect(10_ul == std::size(v));
  };

};
```

```
-> All tests passed (2 asserts in 1 tests)
```

BEHAVIOR DRIVEN DEVELOPMENT (BDD) - HTTPS://GODBOLT.ORG/Z/4MD03K

BEHAVIOR DRIVEN DEVELOPMENT (BDD) - HTTPS://GODBOLT.ORG/Z/4MD03K

```
feature("vector") = [] {
  scenario("size") = [] {
```

```
  };
};
```

BEHAVIOR DRIVEN DEVELOPMENT (BDD) - HTTPS://GODBOLT.ORG/Z/4MDO3K

```
feature("vector") = [] {
  scenario("size") = [] {
```

```
    given("I have a vector") = [] {
      std::vector<int> v(5);
      expect((5_ul == std::size(v)) >> fatal);
```

```
  };
};
```

BEHAVIOR DRIVEN DEVELOPMENT (BDD) - HTTPS://GODBOLT.ORG/Z/4MD03K

```
feature("vector") = [] {
  scenario("size") = [] {
```

```
    given("I have a vector") = [] {
      std::vector<int> v(5);
      expect((5_ul == std::size(v)) >> fatal);
```

```
      when("I resize bigger") = [v] {
        mut(v).resize(10);
```

```
      };
```

```
  };
};
```

BEHAVIOR DRIVEN DEVELOPMENT (BDD) - HTTPS://GODBOLT.ORG/Z/4MDO3K

```
feature("vector") = [] {
  scenario("size") = [] {

    given("I have a vector") = [] {
      std::vector<int> v(5);
      expect((5_ul == std::size(v)) >> fatal);

      when("I resize bigger") = [v] {
        mut(v).resize(10);

        then("The size should increase") = [v] {
          expect(10_ul == std::size(v));
        };

      };

  };
};
```

BEHAVIOR DRIVEN DEVELOPMENT (BDD) - HTTPS://GODBOLT.ORG/Z/4MDO3K

```
feature("vector") = [] {
  scenario("size") = [] {

    given("I have a vector") = [] {
      std::vector<int> v(5);
      expect((5_ul == std::size(v)) >> fatal);

      when("I resize bigger") = [v] {
        mut(v).resize(10);

        then("The size should increase") = [v] {
          expect(10_ul == std::size(v));
        };

      };

  };
};
```

```
-> All tests passed (2 asserts in 1 tests)
```

GHERKIN - HTTPS://GODBOLT.ORG/Z/JB1D8P

GHERKIN - HTTPS://GODBOLT.ORG/Z/JB1D8P

VECTOR.FEATURE

GHERKIN - HTTPS://GODBOLT.ORG/Z/JB1D8P

VECTOR.FEATURE

```
Feature: Vector
  Scenario: Resize
    Given I have a vector
    When I resize bigger
    Then The size should increase
```

GHERKIN - HTTPS://GODBOLT.ORG/Z/JB1D8P

VECTOR.FEATURE

```
Feature: Vector
  Scenario: Resize
    Given I have a vector
    When I resize bigger
    Then The size should increase
```

VECTOR.CPP

GHERKIN - HTTPS://GODBOLT.ORG/Z/JB1D8P

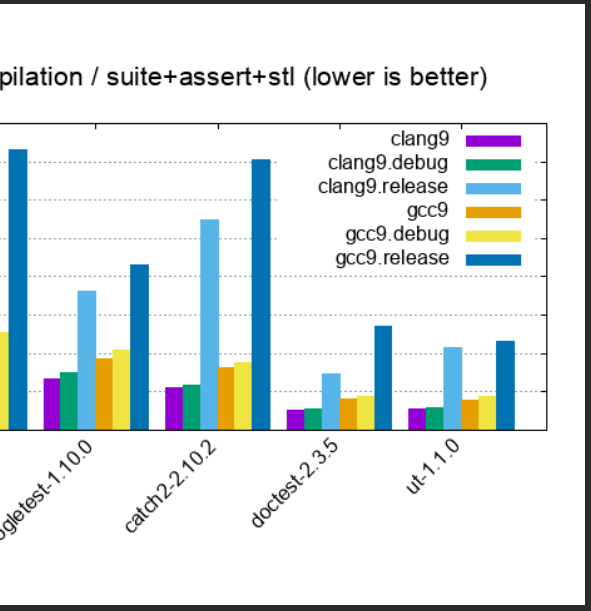
VECTOR.FEATURE

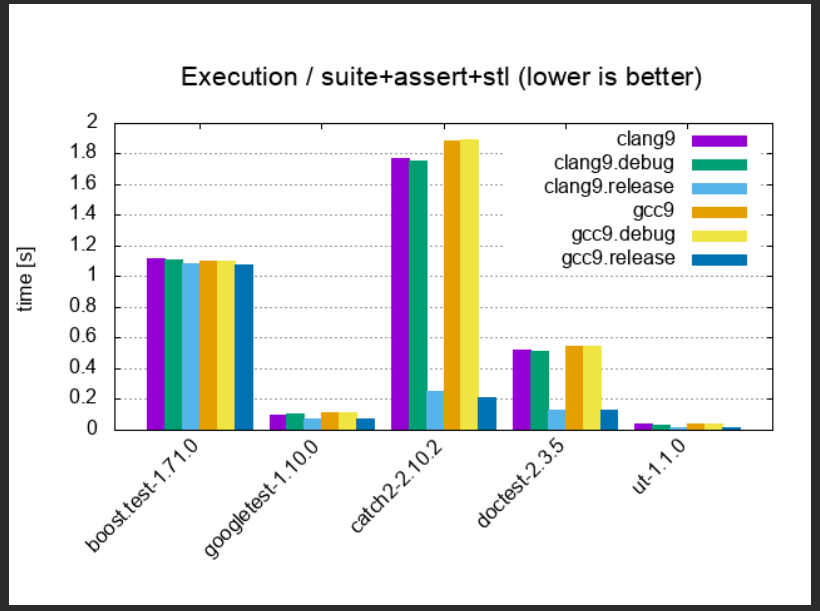
```
Feature: Vector
  Scenario: Resize
    Given I have a vector
    When I resize bigger
    Then The size should increase
```

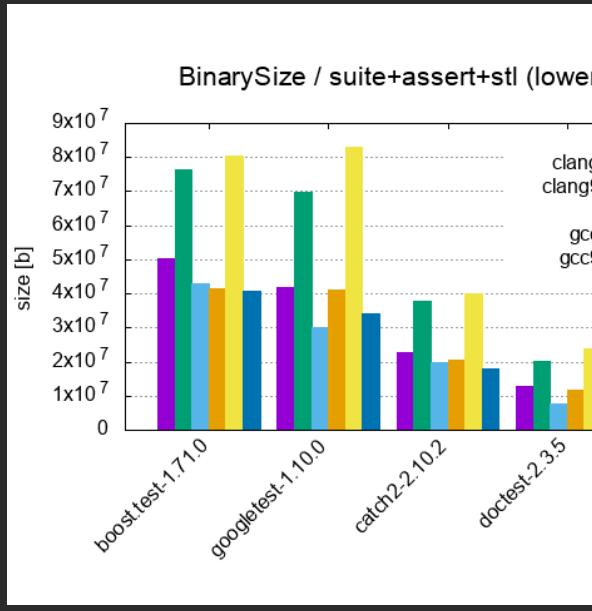
VECTOR.CPP

```
gherkin::steps steps = [](auto& s) {
 s.feature("Vector") = [&] {
  s.scenario("*") = [&]{
   s.given("I have a vector") = [&] {
    std::vector<int> v(5);
    s.when("I resize bigger") = [&] { v.resize(10); };
    s.then("The size should increase") = [&]{ 10_u == std::size(v); };
   };
  };
 };
};
```

BENCHMARKS - SUITE+ASSERT+STL

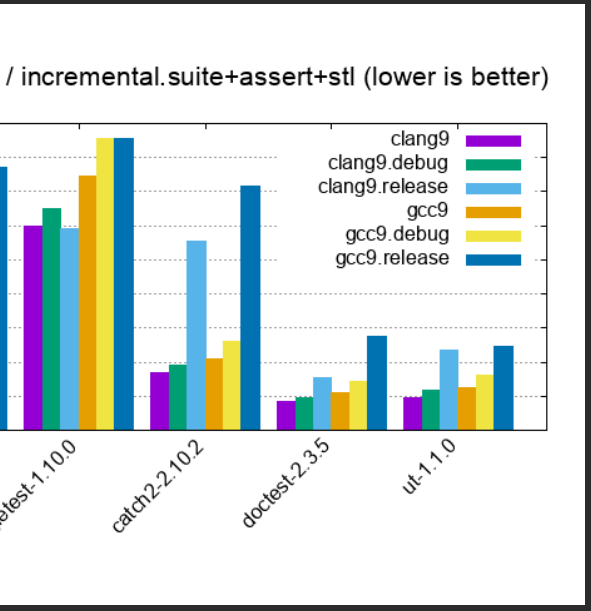


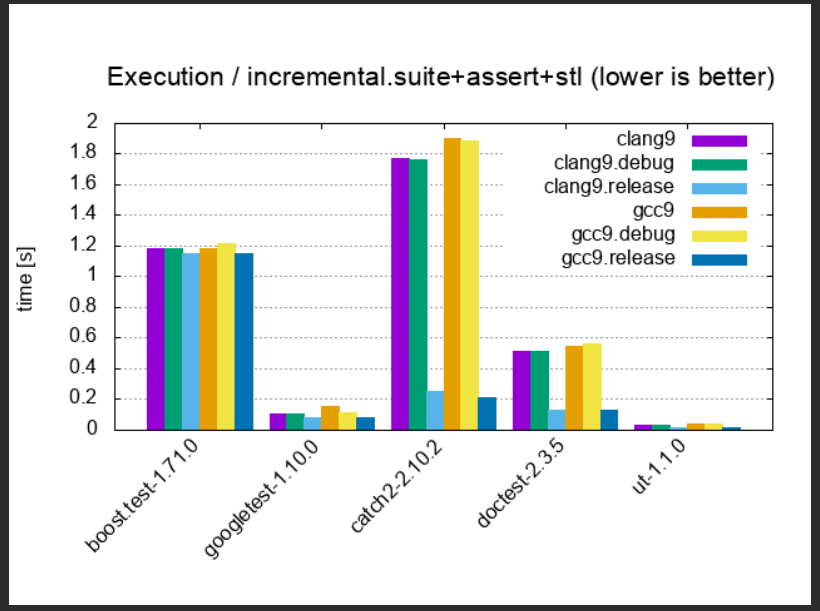


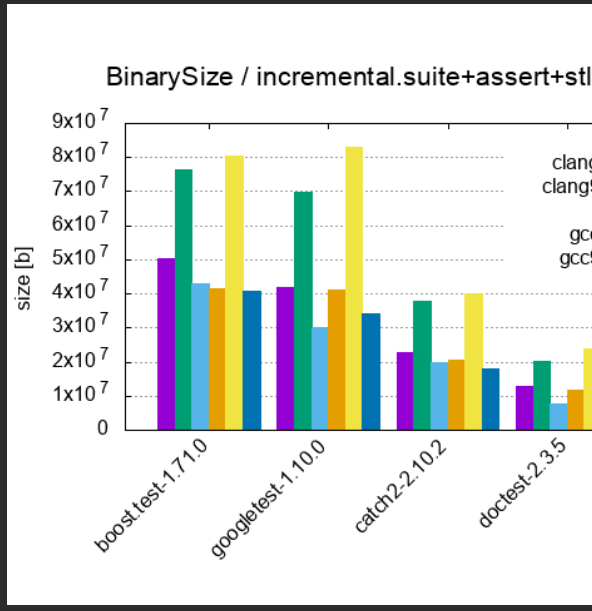


10'000 TESTS, 20'000 ASSERTS, 100 CPP FILES

BENCHMARKS - INCREMENTAL BUILD







1 CPP FILE CHANGE (1'000 TESTS, 20'000 ASSERTS, 100 CPP FILES)

HTTPS://GODBOLT.ORG/Z/Y43MXZ

IF YOU LIKED IT THEN YOU *"SHOULD HAVE PUT A"_ TEST ON IT"*

BEYONCE RULE

HTTPS://GITHUB.COM/BOOST-EXT/UT

CATCH2 - HTTPS://GODBOLT.ORG/Z/JFB7JK

CATCH2 - HTTPS://GODBOLT.ORG/Z/JFB7JK

```
#define REQUIRE(...)    ut::expect(that % __VA_ARGS__)
#define TEST_CASE(...) ut::test{"test", __VA_ARGS__} = [=]() mutable
#define SECTION(name)  ut::test{"section", name} = [=]() mutable
```

CATCH2 - HTTPS://GODBOLT.ORG/Z/JFB7JK

```
#define REQUIRE(...)    ut::expect(that % __VA_ARGS__)
#define TEST_CASE(...) ut::test{"test", __VA_ARGS__} = [=]() mutable
#define SECTION(name)  ut::test{"section", name} = [=]() mutable
```

```
TEST_CASE("vectors can be resized", "[vector]") {
  std::vector<int> v(5);

  SECTION("resize bigger") {
    v.resize(10);
    REQUIRE(10 == std::size(v));
  };
};
```

CATCH2 - HTTPS://GODBOLT.ORG/Z/JFB7JK

```
#define REQUIRE(...)    ut::expect(that % __VA_ARGS__)
#define TEST_CASE(...) ut::test{"test", __VA_ARGS__} = [=]() mutable
#define SECTION(name)  ut::test{"section", name} = [=]() mutable
```

```
TEST_CASE("vectors can be resized", "[vector]") {
  std::vector<int> v(5);

  SECTION("resize bigger") {
    v.resize(10);
    REQUIRE(10 == std::size(v));
  };
};
```

```
-> All tests passed (1 asserts in 1 tests)
```