Centrum Wiskunde & Informatica

Testing your code with Catch

WISM454 Laboratory Class Scientific Computing, Jan-Willem Buurlage

May 21, 2019

Testing

- When developing your library, you continuously test if your code works
- If you make a change, how do you ensure that the behaviour is still as intended?
- One way to try and get some stability, is by using unit tests
- An example of a test framework for C++ is Catch2

Example

```
#define CATCH_CONFIG_MAIN
#include "catch2/catch.hpp"

#include "ranges.hpp"

TEST_CASE("Basic range", "[range]")
{
    auto xs = std::vector<int>();
    for (auto x : range(4)) {
        xs.push_back(x);
    }
    REQUIRE(xs == std::vector<int>{0, 1, 2, 3});
}
```

Example output

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
test_range is a Catch v2.7.2 host application.
Run with -? for options

================================================================================
All tests passed (1 assertions in 1 test case)
```

Adding more tests

```
TEST_CASE("Range with stop", "[range]")
{
    auto xs = std::vector<int>();
    for (auto x : range(4, 8)) {
        xs.push_back(x);
    }
    REQUIRE(xs == std::vector<int>{4, 5, 6, 7});
}
```

Multiple requirements in test case

```
TEST_CASE("Range with step", "[range]")
{
    auto xs = std::vector<int>();
    for (auto x : range(4, 8, 2)) {
        xs.push_back(x);
    }
    REQUIRE(xs == std::vector<int>{4, 6});

    xs.clear();
    for (auto x : range(8, 4, -1)) {
        xs.push_back(x);
    }
    REQUIRE(xs == std::vector<int>{8, 7, 6, 5});
}
```

Benchmarks

```
TEST_CASE("Benchmark", "[benchmark]") {
    auto xs = std::vector<std::array<int, 3>>(1000000);
    xs.reserve(1000000);
    BENCHMARK("Iterate a large grid") {
        for (auto [x, y, z] : grid<3>(0, 100, 1)) {
            xs.push_back({x, y, z});
        }
    }
    REQUIRE(xs[xs.size() - 1] == std::array<int, 3>{99, 99, 99})

    auto ys = std::vector<std::array<int, 3>>(1000000);
    ys.reserve(1000000);
    BENCHMARK("Iterate a large grid with enumerate") {
        for (auto [i, v] : enumerate(grid<3>(0, 100, 1))) {
            ...
```

Benchmark output

```
Benchmark
-------------------------------------------------------------------
<filename>
...................................................................

benchmark name           iters   elapsed ns      average
-------------------------------------------------------------------
Iterate a large range    1       2889999         2.89 ms

===================================================================
```

Conclusion

- While you develop your code, instead of testing it manually write a unit test!
- Whenever you introduce a new feature (or fix an existing bug), test to see if you break on of your previous unit tests.
- When moving to a different implementation, the benchmark numbers can give you an idea of the performance impact of the change.
- Adding unit tests to your GA library is strongly recommended