Getting to Know the Standard Library Session 1

Alex Voronov

Getting to Know the Standard Library

C++ London Uni

1. Introduction to unit testing with Catch2
2. Basic containers
   - std::vector
   - std::string
3. Lambda functions and std::function
4. Associative containers
   - std::map and std::unordered_map
   - std::set and std::unordered_set
   - Associative containers with custom types
   - Set algorithms
5. Overview of algorithms in the standard library

Problem

Write a function that given a vector of integer numbers returns how many of them are positive

C++ London Uni

```cpp
size_t count_positive(const std::vector<int> &numbers) {
    size_t count = 0u;
    for (auto number : numbers) {
        if (number > 0) {
            ++count;
        }
    }

    return count;
}
```

```cpp
size_t count_positive(const std::vector<int> &numbers) {
    size_t count = 0u;
    for (auto number : numbers) {
        if (number > 0) {
            ++count;
        }
    }

    return count;
}

int main() {
    std::cout << count_positive({1, 2, 3, 4, 5}) << std::endl;
    std::cout << count_positive({1, 0, 0, 0, 1}) << std::endl;
    std::cout << count_positive({-1, -1, 0, 0, 42, 27}) << std::endl;
    return 0;
}
```

Catch2 – Unit Testing Framework

https://github.com/catchorg/Catch2

Download *catch.hpp* from the project page and include it in your .cpp file

```cpp
#define CATCH_CONFIG_MAIN
#include "catch.hpp"

#include <vector>

size_t count_positive(const std::vector<int> &numbers) {
    size_t count = 0u;
    for (auto number : numbers) {
        if (number > 0) {
            ++count;
        }
    }

    return count;
}

TEST_CASE("Count positive numbers") {
    CHECK(count_positive({1, 2, 3, 4, 5}) == 5);
    CHECK(count_positive({1, 0, 0, 0, 1}) == 2);
    CHECK(count_positive({-1, -1, 0, 0, 42, 27}) == 2);
}
```

Practice: the very first unit test

1. Download catch.hpp from the Catch2 project page
2. Add it to you project
3. Write a **TEST_CASE** that checks that 3 + 2 == 5 and run the test
4. Modify the expression to make the test fail and output the failure report

```
#define CATCH_CONFIG_MAIN

#include "catch.hpp"

TEST_CASE("My first test") {
    CHECK(3 + 2 == 5);
}
```

C++ London Uni

```
TEST_CASE("Example test") {
    std::vector<int> squares{0, 1, 4, 9, 16, 25};
    CHECK(squares.size() == 6);
    CHECK(squares[0] == 0);
    CHECK(squares[5] == 25);
}
```

```cpp
TEST_CASE("REQUIRE example (failure)") {
    std::vector<int> squares{0, 1, 4, 9, 16}; // Removed last element
    REQUIRE(squares.size() == 6);
    CHECK(squares[0] == 0);
    CHECK(squares[5] == 25);
}

TEST_CASE("REQUIRE example (success)") {
    std::vector<int> squares{0, 1, 4, 9, 16};
    REQUIRE(squares.size() == 5);
    CHECK(squares[0] == 0);
    CHECK(squares[4] == 16);
}
```

When **REQUIRE** fails, the failure is reported and the test execution stops.

When **CHECK** fails, the failure is reported but the test execution continues.

```
TEST_CASE("SECTION example") {
    std::vector<int> squares{0, 1, 4, 9, 16, 25};

    SECTION("Delete all") {
        squares.clear();
        CHECK(squares.size() == 0);
    }

    SECTION("Add an element") {
        squares.push_back(36);
        REQUIRE(squares.size() == 7);
        CHECK(squares[6] == 36);
    }
}
```

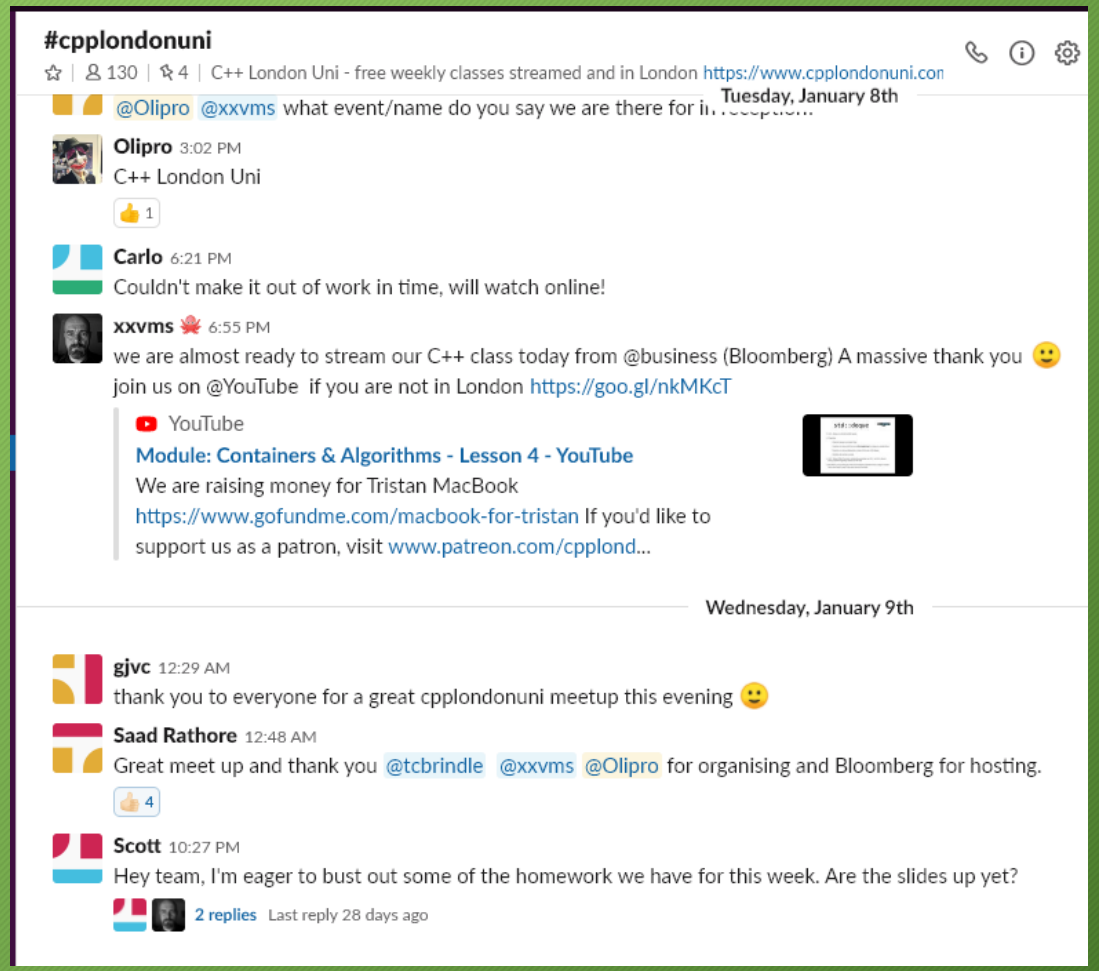
SECTIONS on the same level are run independently. For each SECTION the test code is executed from the beginning of the test.

Summary: unit testing with Catch2

- Catch2 is a single-header unit-test framework, available at https://github.com/catchorg/Catch2
- Prefer writing simple unit tests instead of printing to `std::cout`
- Use **`CHECK`** for simple independent conditions, and **`REQUIRE`** when it doesn't make sense to continue after failure
- Use **`SECTION`**s for independent tests with common setup

Feedback

- We'd love to hear from you!
- The easiest way is via the *CPPLang* Slack organisation. Our chatroom is #cpplondonuni
- If you already use Slack, don't worry, it supports multiple workgroups!
- Go to https://slack.cpp.al to register.



Thank You!

As usual, we will be going to the pub! Support us @ https://patreon.com/CPPLondonUni