Catch2 Example

- want to build program that uses Catch2 library to run test cases for simple header-only library
- code written in C++
- uses Catch2 library (version 3 or greater)
- files in project:
  - `CMakeLists.txt`
  - `test_app.cpp`
  - `lib.hpp`
- project has:
  - executable target `test_app`

Catch2 Example: Source Code

test_app.cpp

```
1  #include <catch2/catch_all.hpp>
2  #include "lib.hpp"
3
4  TEST_CASE("get_meaning_of_life", "[hg2g]") {
5      auto answer = hg2g::get_meaning_of_life();
6      REQUIRE(answer == 42);
7  }
```

lib.hpp

```
1  #ifndef LIB_HPP
2  #define LIB_HPP
3  namespace hg2g {
4  constexpr int get_meaning_of_life() {return 42;}
5  }
6  #endif
```

Catch2 Example: CMakeLists.txt

CMakeLists.txt

```
1  cmake_minimum_required(VERSION 3.20 FATAL_ERROR)
2  project(cmake_catch2 LANGUAGES CXX)
3
4  set(CMAKE_CXX_STANDARD 20)
5  set(CMAKE_CXX_STANDARD_REQUIRED true)
6
7  # Find the Catch2 library (version 3 or greater).
8  find_package(Catch2 3 REQUIRED)
9
10 # Define a program target called test_app.
11 add_executable(test_app test_app.cpp)
12
13 # Specify that the test_app target uses the Catch2 library and needs
14 # this library to provide a main function (for running test cases).
15 target_link_libraries(test_app Catch2::Catch2WithMain)
```

Section 8.9

Catch2

Catch2

- Catch2 (originally known as Catch) is multiparadigm test framework for C++
- provides framework for performing unit testing
- also offers basic microbenchmarking functionality
- simple and easy to use
- open source; released under Boost Software License
- originally written by Phil Nash
- official Git repository: `http://github.com/catchorg/Catch2`

Using Catch2 With CMake

- use `find_package` command in CMake with package name `Catch2`
- specify at least version 3 (of Catch2)
- if package found, provides imported target `Catch2::Catch2WithMain` for Catch2 library with `main` function that can run test cases
- full example of how to use Catch2 with CMake can be found in section on CMake

Counter Class Example: counter.hpp

```
1  #include <limits>
2  #include <stdexcept>
3
4  class counter {
5  public:
6      using count_type = std::size_t;
7      static constexpr count_type max_count() {
8          return std::numeric_limits<count_type>::max();
9      }
10     counter(count_type count = 0) : count_(count) {}
11     count_type get_count() const {
12         return count_;
13     }
14     void increment() {
15         if (count_ == max_count()) {
16             throw std::overflow_error("counter overflow");
17         }
18         ++count_;
19     }
20 private:
21     count_type count_;
22 };
```

Counter Class Example: Test Code

```
1  #include <catch2/catch_all.hpp>
2  #include "counter.hpp"
3
4  TEST_CASE("constructor", "[counter][constructor]") {
5      counter x;
6      CHECK(x.get_count() == 0);
7      counter y(1);
8      CHECK(y.get_count() == 1);
9  }
10
11 TEST_CASE("maximum count", "[counter][method]") {
12     CHECK(counter::max_count() == std::numeric_limits<
13       counter::count_type>::max());
14 }
15
16 TEST_CASE("increment (no overflow)", "[counter][method]") {
17     counter x(0);
18     REQUIRE(x.get_count() == 0);
19     x.increment();
20     CHECK(x.get_count() == 1);
21 }
22
23 TEST_CASE("increment (overflow)", "[counter][method]") {
24     counter x(counter::max_count());
25     CHECK_THROWS_AS(x.increment(), std::overflow_error);
26 }
```

Section Example

```
1  #include <catch2/catch_all.hpp>
2  #include <cstddef>
3  #include <vector>
4
5  TEST_CASE("Check resize", "[vector]") {
6      constexpr std::size_t size = 128;
7      std::vector<int> x(size);
8      REQUIRE(x.size() == size);
9      REQUIRE(x.capacity() >= size);
10     SECTION("Increase size") {
11         std::size_t n = size * 16;
12         x.resize(n);
13         CHECK(x.size() == n);
14         CHECK(x.capacity() >= n);
15     }
16     SECTION("Decrease size") {
17         std::size_t n = size / 16;
18         x.resize(n);
19         CHECK(x.size() == n);
20         CHECK(x.capacity() >= n);
21     }
22     SECTION("Zero size") {
23         x.resize(0);
24         CHECK(x.size() == 0);
25         CHECK(x.capacity() >= size);
26     }
27 }
```

Approximate Comparison Example

```
#include <cmath>
#include <catch2/catch_all.hpp>

TEST_CASE("check addition") {
    float x = 0.0f;
    for (int i = 0; i < 10; ++i) {x += 0.1f;}
    CHECK_NOFAIL(x == 1.0f);
      // condition may be false due to roundoff error
    CHECK_THAT(x, Catch::Matchers::WithinAbs(1.0f, 0.01f));
      // should pass (absolute tolerance 0.01)
    CHECK_THAT(x, Catch::Matchers::WithinRel(1.0f, 0.01f));
      // should pass (relative tolerance 1%)
    float y = std::nextafter(0.0f, 1.0f);
    CHECK_THAT(y, Catch::Matchers::WithinULP(0.0f, 1));
      // should pass (y is next representable value greater than 0.0f)
}
```

Type-Parameterized Test Example: stack.hpp

```
1  #include <cstddef>
2  #include <vector>
3
4  // Note: T is not allowed to be bool.
5  template <class T>
6  class Stack {
7  public:
8      bool empty() const {return s_.empty();}
9      std::size_t size() const {return s_.size();}
10     const T& top() const {return s_.back();}
11     void push(const T& x) {s_.push_back(x);}
12     void pop() {s_.pop_back();}
13 private:
14     std::vector<T> s_;
15 };
```

Type-Parameterized Test Example: Test Code

```
1  #include <catch2/catch_all.hpp>
2  #include <complex>
3  #include "stack.hpp"
4
5  TEMPLATE_TEST_CASE("Check default constructor", "[stack][constructor]",
6    int, double, std::complex<double>) {
7      Stack<TestType> s;
8      CHECK(s.empty());
9      CHECK(s.size() == 0);
10 }
11
12 TEMPLATE_TEST_CASE("Check push and pop", "[stack][modifier]",
13   int, double, std::complex<double>) {
14     std::size_t size = 0;
15     Stack<TestType> s;
16     REQUIRE(s.size() == size);
17     while (size < 3) {
18         ++size; s.push(TestType(size));
19         REQUIRE(s.size() == size);
20         REQUIRE(s.top() == TestType(size));
21     }
22     while (size > 0) {
23         s.pop(); --size;
24         REQUIRE(s.size() == size);
25     }
26 }
```

Test Fixture Example

```
1  #include <catch2/catch_all.hpp>
2  #include <deque>
3  #include <stdexcept>
4
5  class TestFixture {
6  public:
7      TestFixture() : q0(), q1{1, 2, 3} {
8      }
9  protected:
10     std::deque<int> q0;
11     std::deque<int> q1;
12 };
13
14 TEST_CASE_METHOD(TestFixture, "Check deque at", "[deque]") {
15     CHECK_THROWS_AS(q0.at(0), std::out_of_range);
16     CHECK(q1.at(0) == 1);
17     CHECK_THROWS_AS(q1.at(3), std::out_of_range);
18 }
19
20 TEST_CASE_METHOD(TestFixture, "Check deque clear", "[deque]") {
21     q0.clear();
22     CHECK(q0.size() == 0);
23     q1.clear();
24     CHECK(q1.size() == 0);
25 }
```

Benchmark Example

fibonacci.hpp

```
1   #ifndef FIBONACCI_HPP
2   #define FIBONACCI_HPP
3
4   #include <cstdint>
5
6   constexpr uint64_t fibonacci(uint64_t n) {
7       if (n >= 2) {return fibonacci(n - 1) + fibonacci(n - 2);}
8       else if (n == 1) {return 1;}
9       else {return 0;}
10  }
11
12  #endif
```

benchmark_fibonacci.cpp

```
1   #include <catch2/catch_test_macros.hpp>
2   #include <catch2/benchmark/catch_benchmark.hpp>
3   #include "fibonacci.hpp"
4
5   // Note: [!benchmark] is special tag indicating test case is benchmark
6   TEST_CASE("benchmark fibonacci", "[!benchmark]") {
7       REQUIRE(fibonacci(20) == 6'765);
8       BENCHMARK("fibonacci(20)") {
9           return fibonacci(20);
10      };
11  }
```

Talks I

1. Phil Nash. Modern C++ Testing with Catch2. CppCon, Bellevue, WA, USA, Sept. 24, 2018. Available online at https://youtu.be/0b5_XZrFQH0.
2. Phil Nash. Modern C++ Testing with Catch2. Meeting C++, Berlin, Germany, Nov. 9, 2017. Available online at https://youtu.be/3tIE6X5FjDE.
3. Phil Nash. Modern C++ Testing with Catch2. C++ Edinburgh, Edinburgh, UK, Aug. 14, 2017. Available online at https://youtu.be/grC0S6ZK59U.
4. Phil Nash. Test Driven C++ with Catch. CppCon, Bellevue, WA, USA, Sept. 22, 2015. Available online at https://youtu.be/gdzP3pAC6UI.
5. Phil Nash. Testdriven C++ with Catch. Meeting C++, Berlin, Germany, Dec. 5–6, 2014. Available online at https://youtu.be/C2LcIp56i-8.