CS 3460

Introduction to Google Test

What is Google Test

- A unit testing framework for C++ projects
- Requires C++ 14 standard-compliant compiler (with Version 1.14.x)
- Framework is compiled with the tests and code being tested
- A large suite of assertions
  - plus a lot of other test types and parameterizations
- Works with a number of compilers and platforms
  - Windows
  - Linux
  - macOS X
  - others
- Designed for easy integration using CMake

CMake – Integration

- A few approaches
  - https://google.github.io/googletest/quickstart-cmake.html
  - https://stackoverflow.com/questions/38006584/how-to-clone-and-integrate-external-from-git-cmake-project-into-local-one
- Class Demonstration Technique
  - Using CMake to download gtest (via git)
  - Integrate *external* code into a project

CMake – FetchContent

- CMake `FetchContent_Declare` command
  - `GIT_REPOSITORY` : git URL of the project
  - `GIT_TAG` : git branch or tag to checkout

```
include(FetchContent)
FetchContent_Declare(
    googletest
    GIT_REPOSITORY    https://github.com/google/googletest.git
    GIT_TAG           v1.15.0
)

FetchContent_MakeAvailable(googletest)
```

CMake – `CMakeLists.txt` Integration

1. Organize variables holding names of the source files

```
set(HEADER_FILES utilities.hpp)
set(SOURCE_FILES utilities.cpp)
set(UNIT_TEST_FILES TestUtilities.cpp)
```

2. Create two executable targets

```
add_executable(GoogleTestIntro ${HEADER_FILES} ${SOURCE_FILES} main.cpp)
add_executable(UnitTestRunner  ${HEADER_FILES} ${SOURCE_FILES} ${UNIT_TEST_FILES})
```

3. Update the clang-format section

```
foreach(SOURCE_FILE ${HEADER_FILES} ${SOURCE_FILES} ${UNIT_TEST_FILES} main.cpp)
```

CMake – `CMakeLists.txt` Integration

4. Use `set` to disable a compiler setting (Google Test sets it)
   - Disable dynamically linked libraries
5. Use `target_link_libraries` to add shared libraries to the test runner target

   See next slide for these details

CMake – FetchContent

```
include(FetchContent)
FetchContent_Declare(
    googletest
    GIT_REPOSITORY    https://github.com/google/googletest.git
    GIT_TAG           v1.15.0
)
set(gtest_force_shared_crt ON CACHE BOOL "" FORCE)
FetchContent_MakeAvailable(googletest)

target_link_libraries(${UNIT_TEST_RUNNER} gtest_main)
```

`CMakeLists.txt` – Code Inspection

Terminology

- Test
  - One or more assertions to verify code behavior
  - If a test has failed assertions, the test fails
- Test Suite
  - One or more tests
- Test Program
  - The program used to execute the tests (test suites)
  - May contain more than one test suite

Testing Assertions

- Google Test provides a large number of macros used to make assertions about code's behavior
- Typically takes expected value and actual value, then tests for…
  - equality, greater than, less than
  - others

```
ASSERT_EQ(computed, 8);   // assert 'computed' contains the value 8
ASSERT_LE(computed, 8);   // assert 'computed' is less than or equal to 8

EXPECT_EQ(computed, 8);   // assert 'computed' contains the value 8
EXPECT_LE(computed, 8);   // assert 'computed' is less than or equal to 8
```

- `ASSERT_*` After first failure, no others are tested
- `EXPECT_*` All tests are run, even if other failures

Testing Assertions – Floating Point Types

- `ASSERT_*` and `EXPECT_*` aren't appropriate for floating point types
- Specific floating point assertions

```
ASSERT_FLOAT_EQ(computed, expected)
EXPECT_FLOAT_EQ(computed, expected)
```

  - These are true if the values are 'close' to each other.
    - Defined as within four Units in the Last Place (ULP)
    - Read the Google Test documentation for details
- You can define allowable error

```
ASSERT_NEAR(computed, expected, absError)
EXPECT_NEAR(computed, expected, absError)
```

Writing a Test

- We have a function we want to test
- Use the TEST macro to write a, well, test
  - First param is test suite
  - Second is test name

```
void swap(int& x, int& y)
{
     int temp = x;
     x = y;
     y = temp;
}
```

```
TEST(SwapTests, SwapTest)
{
    int a = 1;
    int b = 2;
    cs3460::swap(a, b);
    EXPECT_EQ(a, 2);
    EXPECT_EQ(b, 1);
}
```

Executing Tests

- Add the following to the main function of the test program

```
testing::InitGoogleTest(&argc, argv);

return RUN_ALL_TESTS();
```

- The Advanced Guide link in the Google Test Primer documentation has details on what is taking place with these two statements

Sin Tests – Code Demo