Documentation

include/leap__year.hpp

`bool is_leap_year(int year)`

Calculates whether the parameter, `year`, was a leap year. There's a `#define` to make the test fail, `INTENTIONAL_FAILURE`. This causes the function to NOT throw an exception for years < 1752, which is an error because that was the year that the Gregorian calendar was adopted by the British Empire. (Although it was used by other European nations before that, so it's not an error in those countries…) But I basically just put it in to demonstrate the THROWS() macro.

include/factorial.hpp

`int factorial(int n)`

A simple version of the factorial function. Caveat: Does not perform error checks!

test/factorial-test.hpp

toy__test::suite factorial__suite

A sample test suite. Tests the factorial function in factorial.hpp

test/leap-year-test.hpp

toy__test::suite leap__year__suite

A sample test suite. Tests the leap year formula function in leap__year.hpp

test/toy__test.hpp

namespace toy__test

`class toy_test::test_case`

An element of a `suite`. Contains a name and an anonymous function. Usually created anonymously, as an element of an array within a `suite`.

`class toy_test::failure`

Holds information about a `test_case`'s failure. Not usually instantiated directly, but from the failure of an `ASSERT`.

`class toy_test::suite`

A container for `test_case`s. Holds a name and a `std::vector` of `test_case`s. Usually initialized using aggregate initialization.

`bool toy_test::suite::run()`

Function that executes the `test_case`s in a `suite`. It prints the results of the test run, including how many `test_case`s succeeded and failed.

`bool toy_test::run_suite(suite const& suite)`

Runs the contents of a `suite`. Returns `false` if there were any failures.

`bool toy_test::run_suites(std::initializer_list<suite const> const& suites)`

Runs the contents of a set of `suite`s, given as a list-initialized parameter. Returns `false` if there were any failures in any `suite`.

`macro ASSERT(condition)`

Tests a condition. Fails the `test_case` if the condition is `false`. Also aborts the `test_case` on failure.

`macro THROWS(expression, exception)`

Tests to ensure that a provided expression causes a particular exception, or one of its subtypes, is thrown. Fails the `test_case` and aborts it if the expected exception is not thrown.

test/demo-test.hpp

`namespace demo_test_internal`

Just a `namespace` to wrap the test `suite` and any necessary types and variables so they don't pollute the global `namespace`.

`toy_test::suite demo_test_internal::demo_suite`

This is the test `suite`. It has one `test_case` that passes, and one that fails.

`toy_test::suite demo_suite`

This is a variable alias to make the test `suite` available in the global `namespace`. The idea is that if you have `using` statements, `typedef`s, and variables inside the `suite`'s `namespace`, they don't pollute the global `namespace`. However, we can still address the `suite` itself using a global variable.