Compile-Time Assertions

This compile-time analog to the classic runtime `assert` causes compilation to terminate with a user-supplied error message whenever its **constant-expression** argument evaluates to **`false`**.

Description

Assumptions are inherent in every program, whether we explicitly document them or not. A common way of validating certain assumptions at run time is to use the classic `assert` macro found in `<cassert>`. Such runtime assertions are not always ideal because (1) the program must already be built and running for them to even have a chance of being triggered and (2) executing a **redundant check** at run time typically[1] results in a slower program. Being able to validate an assertion at compile time avoids several drawbacks.

1. Validation occurs at compile time within a single translation unit and therefore doesn't need to wait until a complete program is linked and executed.
2. Compile-time assertions can exist in many more places than runtime assertions and are unrelated to program control flow.
3. No runtime code will be generated due to a **`static_assert`**, so program performance will not be impacted.

Syntax and semantics

We can use **static assertion declarations** to conditionally trigger controlled compilation failures depending on the truthfulness of a **constant expression**. Such declarations are introduced by the **`static_assert`** keyword, followed by a parenthesized list consisting of (1) ~~a constant Boolean expression~~ and (2) a mandatory **string literal** (see *Annoyances — Mandatory string literal* on page 123), which will be part of the compiler diagnostics if the compiler determines that the assertion fails to hold:

```
static_assert(true, "Never fires.");
static_assert(false, "Always fires.");
```

[1] It is not unheard of for a program having runtime assertions to run faster with them enabled than disabled. For example, asserting that a pointer is not null enables the optimizer to elide all code branches that can be reached only if that pointer were null.