Extended Literals

- Unicode string literals help declare UTF strings
- Raw string literals provide an easy way to quote long strings with various characters otherwise forbidden in strings (unquoted)
  - The delimiter can be customized, e.g. `R"$(...)$"`

```
auto utf8string  = u8"Hello";  // same as "Hello"
auto utf16string = u"Hello";   // same as L"Hello"
auto utf32string = U"Hello";


auto raw = R"(I can put " here and also \)";
```

User-Defined Literals

- Custom operators that accept several built in types (including C strings) and produce custom values
- Must begin with _ to avoid conflicts with standard-defined literals

```
unsigned long long operator "" _kb(unsigned long long v)
{
  return v * 1024;
}

std::cout << 3_kb << std::endl; // prints 3072
```

Standard-Defined Literals

C++ 14

- The standard library defines some literals in the `std::literals` inline namespace
  - Chrono literals for time durations
  - Complex literals for complex numbers
  - String literals for `std::basic_string<>`

```
auto break_time = 5min; // std::chrono::minutes

auto c = 0.5 + 1.0i;    // std::complex<double>

auto message = L"Hi there"s;   // std::wstring
```

Binary Literals, Digit Separators

C++ 14

- Yay, binary literals!

```
auto bitmask = 0b1001101110;
```

- Yay, digit separators (anywhere you'd like)!

```
auto billion = 1'000'000'000;
auto bitmask = 0b1101'0010'0011;
```