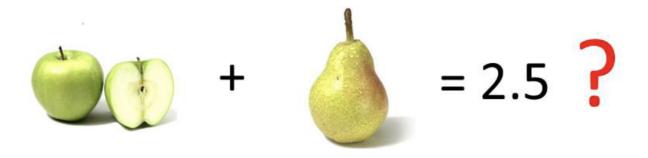
#### **User-Defined Literals**

In this lesson, we'll learn how to create user-defined literals.

#### WE'LL COVER THE FOLLOWING ^

- Syntax
- The magic
- Further information

User-defined literals are a unique feature in all mainstream programming languages. They empower us to combine values with units.



With C++11, it's possible to generate user-defined literals by adding a suffix to a built-in literal for integers, floating points, characters, and C strings.

## Syntax #

User-defined literals must obey the following syntax:

Usually, we use the suffix for a unit:

# The magic #

The C++ runtime maps the user-defined literal onto the corresponding literal operator. This literal operator has to be implemented by the programmer:

- 1\_m -> operator "" \_m(1){ ...
- "hello"\_i18n -> operator "" \_i18n("hello", 5)

Let's have a look at the user-defined literal <code>0101001000\_b</code> which represents a binary value. The compiler maps the user-defined literal <code>0101001000\_b</code> to the literal operator <code>operator"" \_b(long long int bin)</code>. A few special rules are still missing.

There has to be a space between the quotation marks ("") and the underscore with suffix (\_b). We have the binary value (0101001000) in the variable bin.

If the compiler doesn't find the corresponding literal operator, compilation will fail. In C++14, an alternative syntax for user-defined types was introduced. The syntax differs from the C++11 because it requires no space. Therefore, it is possible to use reserved keywords like \_c as a suffix; for example, we may define a literal of the form \_11\_C . The compiler will map \_11\_C to the literal operator""\_C(unsigned long long int) . The simple rule in C++14 is that we can use suffixes starting with an uppercase letter.

User-defined literals are an essential feature of modern C++ when writing safety-critical software. Why? Thanks to the automatic mapping of user-defined literals to literal operators we can implement type-safe arithmetic. The compiler ensures that it is impossible to add apples and pears.

### Further information #

• reserved keywords

In the next lesson, we'll delve further into built-in literals in C++14.