Introduction

Now we move on to regular expressions in C++. Our first step will be to understand the purpose of this library.

Regular expressions are a language for describing text patterns. They need the header `<regex>`.

Regular expressions are a powerful tool for the following tasks:

- Check if a text matches a text pattern: `std::regex_match`
- Search for a text pattern in a text: `std::regex_search`
- Replace a text pattern with a text: `std::regex_replace`
- Iterate through all text patterns in a text: `std::regex_iterator` and `std::regex_token_iterator`

C++ supports six different grammars for regular expressions. By default, the ECMAScript grammar is used. This one is the most powerful of the six grammars, and is quite similar to the grammar used in Perl 5. The other five grammars are the basic, extended, awk, grep, and egrep grammars.

> 🗝️ **Use raw strings**
> Use raw string literals in regular expressions. The regular expression for the text in C++ is quite ugly: `C\\+\\+`. We have to use two backslashes for each + sign. First, the + sign is a special character in a regular expression. Second, the backslash is a special character in a string. Therefore one backslash escapes the + sign, the other backslash escapes the backslash. By using a raw string literal, the second backslash is not necessary anymore, because the backslash is not interpreted in the string.

```
#include <regex>
//...
std::string regExpr("C\\+\\+");
std::string regExprRaw(R"(C\+\+)");
```

Dealing with regular expressions is typically done in three steps:

I. Define the regular expression:

```
std::string text="C++ or c++.";
std::string regExpr(R"(C\+\+)");
std::regex rgx(regExpr);
```

II. Store the result of the search:

```
std::smatch result;
std::regex_search(text, result, rgx);
```

III. Process the result:

```
std::cout << result[0] << std::endl;
```

In the next lesson, we'll learn how regex identifies the different types of text in C++.