Regular Expression Objects

Let's take a look at the various types and grammars which C++ provides for regex objects.

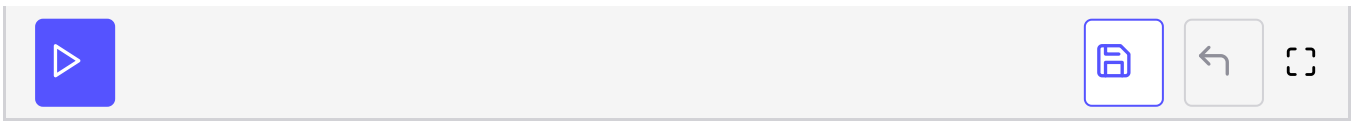
Objects of regular type expressions are instances of the class template `template <class charT, class traits= regex_traits <charT>> class basic_regex` parametrized by their character type and traits class. The traits class defines the interpretation of the properties of regular grammar. There are two type synonyms in C++:

```
typedef basic_regex<char> regex;
typedef basic_regex<wchar_t> wregex;
```

We can further customize the object of type regular expression. Therefore we can specify the used grammar or adapt the syntax. As said before, C++ supports the basic, extended, awk, grep, and egrep grammars. A regular expression qualified by the `std::regex_constants::icase` flag is case insensitive. If we want to adopt the syntax, we have to specify the grammar explicitly.

```
#include <iostream>
#include <regex>
#include <string>

int main(){

  std::cout << std::endl;

  std::string theQuestion="C++ or c++, that's the question.";

  // regular expression for c++
  std::string regExprStr(R"(c\+\+)");

  // regular expression object
  std::regex rgx(regExprStr);

  // search result holder
  std::smatch smatch;

  std::cout << theQuestion << std::endl;
```

```
  // looking for a partial match (case sensitive)
  if (std::regex_search(theQuestion, smatch, rgx)){


    std::cout << std::endl;
    std::cout << "The answer is case sensitive: " << smatch[0] <<  std::endl;

  }

  // regular expression object (case insensitive)
  std::regex rgxIn(regExprStr, std::regex_constants::ECMAScript|std::regex_constants::icase);

  // looking for a partial match (case sensitive)
  if (std::regex_search(theQuestion, smatch, rgxIn)){

    std::cout << std::endl;
    std::cout << "The answer is case insensitive: " << smatch[0] << std::endl;

  }

  std::cout << std::endl;

}
```



Specify the grammar

If we use the case-sensitive regular expression `rgx`, the result of the search in the text `theQuestion` is `c++`. That's not the case if our case-insensitive regular expression `rgxIn` is applied. Now we get the matching string `C++`.

In the next lesson, we'll discuss `std::match_results` and how it is used to verify a regular expression and to store the results.