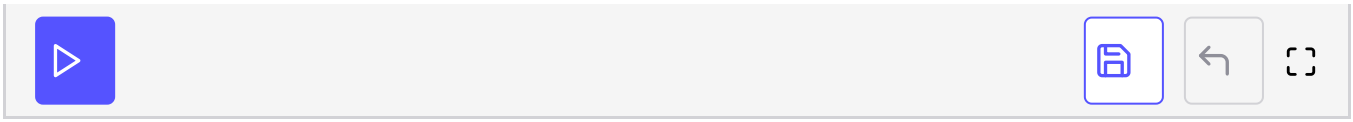
Replace

Along with searching, we can also alter the text if it matches our regex condition.

`std::regex_replace` replaces sequences in a text that match a text pattern. It returns its result in the simple form `std::regex_replace(text, regex, replString)` as a string. The function replaces an occurrence of `regex` in `text` with `replString`.

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

int main(){

  std::cout << std::endl;

  std::string future{"Future"};
  int len= sizeof(future);

  std::string unofficialStandardName{"The unofficial name of the new C++ standard is C++0x."}
  std::cout << std::setw(len) << std::left << "Past: " << unofficialStandardName << std::endl

  // replace C++0x with C++11
  std::regex rgxCpp(R"(C\+\+0x)");
  std::string newCppName{"C++11"};

  std::string newStandardName{std::regex_replace(unofficialStandardName, rgxCpp, newCppName)}
  // replace unofficial with official
  std::regex rgxOff{"unofficial"};
  std::string makeOfficial{"official"};

  std::string officialName{std::regex_replace(newStandardName, rgxOff, makeOfficial)};
  std::cout << std::setw(len) << std::left  << "Now: " << officialName << std::endl;

  std::cout << std::endl;

}
```



std::replace

In addition to the simple version, C++ has a version of `std::regex_replace`

working on ranges. It enables us to push the modified string directly into another string:

```
typedef basic_regex<char> regex;
std::string str2;
std::regex_replace(std::back_inserter(str2), text.begin(), text.end(), regex,replString);
```

All variants of `std::regex_replace` have an additional optional parameter. If we set the parameter to `std::regex_constants::format_no_copy` we will get the part of the text that matches the regular expression, the unmatched text is not copied. If we set the parameter to `std::regex_constants::format_first_only`, the `std::regex_replace` will only be applied once.

In the next lesson, we'll solve an exercise related to `std::regex_replace`.