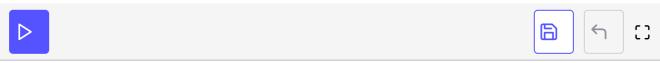
Replace

Along with searching, you can also alter the text if it matches your regex condition.

std::regex_replace replaces sequences in a text matching a text pattern. It
returns in the simple form std::regex_replace(text, regex, replString) its
result as 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;</pre>
  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::end]</pre>
  // 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;</pre>
  std::cout << std::endl;</pre>
}
```



std::replace

working on ranges. It enables you 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
you set the parameter to std::regex_constants::format_no_copy you will get
the part of the text matching the regular expression, the unmatched text is not
copied. If you set the parameter to std::regex_constants::format_first_only
std::regex_replace will only be applied once.