The Search Result

Whenever we verify whether a piece of text satisfies our regular expression, we have to store the results somewhere. std::match_results allows us to do just that.

WE'LL COVER THE FOLLOWING ^

std::sub_match

The object of type <code>std::match_results</code> is the result of a <code>std::regex_match</code> or <code>std::regex_search</code>. <code>std::match_results</code> is a sequential container having at least one capture group of a <code>std::sub_match</code> object. The <code>std::sub_match</code> objects are sequences of characters.

i What is a capture group?

Capture groups allow it to further analyse the search result in a regular expression. They are defined by a pair of parentheses (). The regular expression ((a+)(b+)(c+)) has four capture groups: ((a+)(b+)(c+)), (a+), (b+) and (c+) The total result is the 0th capture group.

C++ has four types of synonyms of type std::match_results:

```
typedef match_results<const char*> cmatch;
typedef match_results<const wchar_t*> wcmatch;
typedef match_results<string::const_iterator> smatch;
typedef match_results<wstring::const_iterator> wsmatch;
```

The search result std::smatch has a powerful interface.

Method	Description
<pre>smatch.size()</pre>	Returns the number of capture

groups. Returns if the search result has a smatch.empty() capture group. Returns the ith capture group. smatch[i] Returns the length of the ith smatch.length(i) capture group. Returns the position of the ith smatch.position(i) capture group. Returns the ith capture group as smatch.str(i) string. Returns the string before and after smatch.prefix() and the capture group. smatch.suffix() Returns the begin and end iterator smatch.begin() and smatch.end() for the capture groups. Formats std::smatch objects for smatch.format(...) the output.

Interface of std::smatch

The following program shows the output of the first four capture groups for different regular expressions.

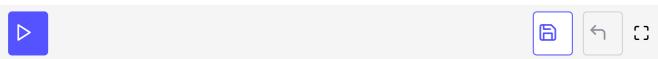
```
#include <regex>

#include <iomanip>
#include <iostream>
#include <string>

void showCaptureGroups(const std::string& regEx, const std::string& text){

// regular expression holder
std::regex rgx(regEx);
```

```
// result holder
  std::smatch smatch;
  // result evaluation
  if (std::regex_search(text, smatch, rgx)){
    std::cout << std::setw(10) << regEx << std::setw(30) << text << std::setw(30) << smatch[0]
}
int main(){
  std::cout << std::endl;</pre>
  std::cout << std::setw(10) << "reg Expr" << std::setw(30) << "text" << std::setw(30) << "sm
  showCaptureGroups("abc+", "abccccc");
  showCaptureGroups("(a+)(b+)(c+)", "aaabccc");
  showCaptureGroups("((a+)(b+)(c+))", "aaabccc");
  showCaptureGroups("(ab)(abc)+", "ababcabc");
  std::cout << std::endl;</pre>
}
```



Capture groups

std::sub_match

The capture groups are of type std::sub_match. As with std::match_results C++ defines the following four type synonyms.

```
typedef sub_match<const char*> csub_match;
                                                                                        6
typedef sub_match<const wchar_t*> wcsub_match;
typedef sub_match<string::const_iterator> ssub_match;
typedef sub_match<wstring::const_iterator> wssub_match;
```

You can further analyze the capture group cap.

Method	Description
<pre>cap.matched()</pre>	Indicates if this match was successful.

Returns the begin and end iterator of the character sequence.

Returns the length of the capture group.

Returns the capture group as string.

Compares the current capture group with another capture group.

The `std::sub_match` object

Here is a code snippet showing the interplay between the search result std::match_results and its capture groups std::sub_match.

```
#include <iostream>
#include <string>
#include <regex>
int main()
    // Simple regular expression matching
    std::string fnames[] = {"foo.txt", "bar.txt", "baz.dat", "zoidberg"};
    std::regex txt_regex("[a-z]+\\.txt");
    for (const auto &fname : fnames) {
        std::cout << fname << ": " << std::regex_match(fname, txt_regex) << '\n';</pre>
    }
    // Extraction of a sub-match
    std::regex base_regex("([a-z]+)\\.txt");
    std::smatch base_match;
    for (const auto &fname : fnames) {
        if (std::regex_match(fname, base_match, base_regex)) {
            // The first sub_match is the whole string; the next
            // sub_match is the first parenthesized expression.
            if (base_match.size() == 2) {
                std::ssub_match base_sub_match = base_match[1];
                std::string base = base_sub_match.str();
                std::cout << fname << " has a base of " << base << '\n';</pre>
            }
        }
    }
    // Extraction of several sub-matches
```

```
std::regex pieces_regex("([a-z]+)\\.([a-z]+)");
std::smatch pieces_match;

for (const auto &fname : fnames) {
    if (std::regex_match(fname, pieces_match, pieces_regex)) {
        std::cout << fname << '\n';
        for (size_t i = 0; i < pieces_match.size(); ++i) {
            std::ssub_match sub_match = pieces_match[i];
            std::string piece = sub_match.str();
            std::cout << " submatch " << i << ": " << piece << '\n';
        }
    }
}</pre>
```







[]

std::sub_match