

Recherche dans les chaînes

Rechercher un caractère

La recherche d'un caractère dans une chaîne peut également être réalisée en utilisant des fonctions membres et des fonctions libres (comme pour les fonctions `begin` et `end` dans le chapitre précédent). C'est le cas de la fonction de base pour faire une recherche : `find` (*trouver* en anglais). D'autres fonctions de recherche ne sont utilisable qu'en fonction libre :

- `rfind` recherche à partir de la fin (*reverse find*) ;
- `find_first_of` recherche la première occurrence d'un caractère ;
- `find_first_not_of` recherche la première absence de caractère ??? ;
- `find_last_of` recherche la dernière occurrence d'un caractère ??? ;
- `find_last_not_of` recherche la dernière absence d'un caractère ???.

D'autres fonctions ne sont utilisables que sous forme de fonction libre (ce sont les algorithmes de la bibliothèque standard) :

- `count` pour compter le nombre d'occurrence d'un caractère ;

Vérifier qu'une chaîne contient une sous-chaîne

```
string const s { "une petite phrase" };
cout << s.find("petite") << endl;
cout << s.find("totot") << endl;
cout << s.find('w') << endl;
```

```
cout << s.rfind("petite") << endl;

// réutilisation de pos
auto pos { s.find('e') }; // trouve le premier e
cout << pos << endl;
cout << s.find('e', pos) << endl;
```

find_first_of : 1 caractère dans une liste de caractères

find_first_not_of : l'inverse

find_last_of

find_last_not_of

parsing : extraire une information

parcourir toutes les correspondances

token : split une chaîne en sous chaînes

main.cpp

```
{
    std::regex pattern("(ab)cd(ef)");    // Find double
word.
    std::string replacement = "le premier groupe est $1
et le second groupe est $2";
    std::string target = "abcdef";
    std::string output_str = regex_replace(target,
pattern, replacement);
    std::cout << output_str << std::endl;
}
{
    std::regex pattern(R
"((\\d{2})[-/](\\d{2})[-/](\\d{4}))");
    std::smatch match;
    std::regex_search(std::string("12-03-2014"), match,
pattern);
    for (size_t i = 0; i < match.size(); ++i)
    {
        std::cout << i << ": " << match[i].str() << '\n';
    }
}
```

```

    }

    std::cout << "Traduction" << std::endl;
    {
        std::regex pattern("([a-zA-Z]+) \\1");
        std::string replacement = "$1";
        std::string target = "The cat cat bites the dog
dog.";
        std::string output_str = regex_replace(target,
pattern, replacement);
        std::cout << output_str << std::endl;
    }

    std::cout << tr("bla bla $1 bla bla", 123) << std::endl;
    std::cout << tr("bli bli bli bli $1", 123) << std::endl;
    std::cout << std::endl;

    std::cout << "Groupe" << std::endl;
    match("", R"((ab)*)");
    match("a", R"((ab)*)");
    match("b", R"((ab)*)");
    match("ab", R"((ab)*)");
    match("abc", R"((ab)*)");
    match("ababab", R"((ab)*)");
    std::cout << std::endl;

    match("cat", R"(c[a-z]*t)");
    std::cout << std::endl;

    std::cout << "Exemples de regex" << std::endl;
    std::cout << "Date" << std::endl;
    match("12-03-2014", R"(\d{2}[-/]\d{2}[-/]\d{4})");
    std::cout << "Time" << std::endl;
    match("15:17", R"(\d{2}:\d{2})");
    std::cout << std::endl;

    // vérifier qu'un identifiant C++ est valide
    // [a-zA-Z_][a-zA-Z_0-9]*

    // fichier windows

```

```

//[a-zA-Z][a-zA-Z_0-9]*\.[a-zA-Z0-9]+

std::string regex_str = "[a-z][a-z_0-9]*\\. [a-z0-9]+";
std::regex regl(regex_str, std::regex_constants::icase);
std::string str = "File names are readme.txt and
my.cmd.";
std::sregex_iterator it(str.begin(), str.end(), regl);
std::sregex_iterator it_end;
while(it != it_end) {
    std::cout << it->str() << std::endl;
    ++it;
}
}

```

Groups

le premier groupe est ab et le second groupe est ef

0: 12-03-2014

1: 12

2: 03

3: 2014

Traduction

The cat bites the dog.

bla bla 123 bla bla

bli bli bli bli 123

Groupe

"(ab)*" match with "" = true

"(ab)*" match with "a" = false

"(ab)*" match with "b" = false

"(ab)*" match with "ab" = true

"(ab)*" match with "abc" = false

"(ab)*" match with "ababab" = true

"c[a-z]*t" match with "cat" = true

Exemples de regex

Date

"\d{2}[/-]\d{2}[/-]\d{4}" match with "12-03-2014" = true

Time

"\d{2}:\d{2}" match with "15:17" = true

readme.txt

```
my.cmd
```

Manque : vorace

A déplacer dans le chapitre sur les recherches ?

```
main.cpp
```

```
// non-greedy (non vorace)
search("aaaaa", R("a{3}"));    // -> aaaaa = plus
longue correspondance
search("aaaaa", R("a{3}?"));    // -> aaa = plus courte
correspondance
std::cout << std::endl;
}
```

affiche :

```
search "a{3}" in "aaaaa" = true
search "a{3}?" in "aaaaa" = true
```

Chapitre précédent	Sommaire principal	Chapitre suivant
------------------------------------	------------------------------------	----------------------------------

[Cours, C++](#)