Les expressions régulières

http://www.informit.com/articles/article.aspx?p=2079020

Création et initialisation

Fichier d'en-tête : #include <regex>

Initialisation : std::regex pattern { “bla bla bla” }; Utilisation des raw string

Fonctionnalités principales

Sera détaillé par la suite.

- match : vérifier qu'une chaîne donnée correspond à un motif
- recherche : trouve une sous-chaîne correspondant à un motif dans une chaîne
- remplacement : recherche une sous-chaîne et la remplace

Parcourir : regex_iterator et regex_token_iterator. Sera vu plus tard ?

Introduction au langage des regex

Définitions :

une expression régulière est un motif représenté par une chaîne

- séquence cible (*Target sequence*) : chaine sur laquelle est appliqué la recherche
- motif (*pattern*) : séquence de caractères représentant ce que vous rechercher

- correspondance (*match*) : sous-chaîne de la séquence cible qui correspond au motif

Syntaxe

Plusieurs syntaxes possible, par défaut ECMAScript. Autre : basic posix, extended posix, awk, grep, egrep

```
std::regex("meow", std::regex::ECMAScript|std::regex::icase)
```

Vérifier qu'une chaîne correspond à un motif. Par exemple, vérifier un code postal, une date, un nombre, etc.

```
std::regex pattern { R"(\d{5})" }; -> 5 chiffres
```

Caractères spéciaux : ^ $ \ . * + ? ( ) [ ] { } | ont un sens spécifique. Pour utiliser le caractère "normal" correspondant à un caractère spécial, précéder de \ : \^ \$
\. * \+ \? \(\) \[\] \{ \} \|

Caractères génériques :

- . 1 caractère quelconque
- * 0 ou plus
- + 1 ou plus
- ? optionnel (0 ou 1)

Ancres début et fin :

- ^ début : “^abc” = doit commencer par “abc”
- $ fin : "abc$” = doit terminer par “abc”

Comptage :

- {n} exactement n fois
- {n,} au moins n fois
- {n,m} entre n et m fois

Groupe et classes [] ()

Classes de caractères (et abréviation)

main.cpp

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

void eval_regex(std::string const& target, std::string const
& pattern_str) {
    std::regex pattern { pattern_str };
    std::cout << R")" << pattern_str << R" in )" <<
target << R" = )" << std::boolalpha << std::regex_match(
target, pattern) << std::endl;
}

int main()
{
    eval_regex("", R"(.)");
    eval_regex("a", R"(.)");
    eval_regex("abc", R"(.)");
    std::cout << std::endl;

    eval_regex("", R"(.*)");
    eval_regex("a", R"(.*)");
    eval_regex("abc", R"(.*)");
    std::cout << std::endl;

    eval_regex("", R"(.+)");
    eval_regex("a", R"(.+)");
    eval_regex("abc", R"(.+)");
    std::cout << std::endl;

    eval_regex("abc", R"(^abc.*)");
    eval_regex("abcdef", R"(^abc.*)");
    eval_regex("defabc", R"(^abc.*)");
}
```

http://stormimon.developpez.com/dotnet/expressions-regulieres/

Chapitre précédent	Sommaire principal	Chapitre suivant