Міністерство освіти і науки України
Київський національний університет імені Т.Г. Шевченка

Регулярні вирази.
Бібліотека <regex>

Виконала студентка 2 курсу,
Механіко-математичного
факультету,
спеціальність: комп’ютерна
математика,
підгрупа: 2
Рубан Дарія Сергіївна

Київ – 2021

Поняття про регулярні вирази

Регулярний вираз - це вираз, який містить послідовність символів, що визначають певний шаблон пошуку. Такі шаблони використовують в алгоритмах пошуку підрядків, а також широко використовуються для перевірки формату вхідних даних, наприклад, адреса електроної пошти має містити знак ‘@’ між двома стрінгами, можна задати формат цих стрінгів – лише літери, цифри, знаки ‘-’. ‘_’, ‘.’ не на першрму місці, правий від ‘@’ рядок містить ‘.’, але не скраю – все це можна виразити за допомогою регулярного виразу. Кожен символ у регулярному виразі - це або символ із буквальним значенням, або “метасимвол”, що має особливе значення.

Наприклад, регулярний вираз 'a[a-z]' може мати значення 'aa', 'ab', 'ax' і т.д. Тут перша літера ‘a’ має буквальне значення. У виразі '[a-z]' метасимволи '[', ']', '-' мають спеціальне значення: квадратні дужки означають “будь-який один із символів в дужках”, дефіс позначає діапазон, тобто вираз '[a-z]' - це співпадіння з будь-якою малою літерою від a до z. Даний приклад простий. Регулярні вирази можуть бути дуже складними.

Розглянемо детальніше, як можна позначити діапазон символів з допомогою регулярного виразу.

Діапазони

[d-n] → **k** *або* **j** *або* **e** *etc, але не* **a** *або* **z** *etc*

- діапазон малих літер від d до n. Це буде відповідати рівно одному малому символу.

[dn] → **d** *або* **n**

- діапазон з кількох конкретних символів

[A-Za-z0-9] → **F** *або* **h** *або* **8**

- діапазон, що містить один окремий символ великого регістру або нижнього регістру або цифру від 0 до 9.

Якщо потрібно вказати символи, які не входять у зазначений набір, то використовують символ ’^’ всередині квадратних дужок:

[^0-9] → d *або* **n** *і т.п., але не* **0, 3, ...**

- будь-який символ, крім цифр.

Екранування

Більшість символів у регулярному виразі представляють самі себе, за винятком спеціальних символів (метасимволів): [ ] \ / ^ $. | ? * + ( ) { }

Дужки [] у наведених вище виразах мають особливе значення. Але якщо потрібно включити дужку як частину виразу, необхідно вдатися до спеціального прийому, який називають екранування спецсимвола або escaping (уникнення). Для цього перед спецсимволом, який потрібно сприймати буквально (а не як мета символ), ставлять ‘\’.

Отже, ‘[’ - метасимол, ‘\[’ - символ “квадратна дужка”.

\[[0-9]\] → [3]

Цей вираз вказує на відкриваючу дужку, цифру в діапазоні від 0 до 9 та закриваючу дужку.

Квантифіктори

Наведені раніше приклади діапазонів відповідають лише одному символу або літералу. Якщо ми хочемо зіставити більше одного символу, ми зазвичай вказуємо квантифікатор разом із шаблоном, роблячи його повторюваним.

Квантифікатор	Кількість повторень	Вираз	Відповідність
*	Нуль або більше	colou*r	color, colour, colouur і т.д.
		[a-z] *	Порожній рядок, c, asdfqwerty
		(Xyz) *	Порожній рядок, Xyz, XyzXyz, XyzXyzXyz
+	Одне або більше	colou+r	colour, colouur і т.д. (але не color)

		[a-z] +	c, asdfqwerty
		(Xyz) +	Xyz, XyzXyz, XyzXyzXyz
?	Нуль або одне	colou?r	Color, colour
		[a-z] ?	Порожній рядок, x, y, z, etc
		(Xyz) ?	Порожній рядок, Xyz

Групування

Кілька символів можуть бути згруповані в межах регулярного виразу, для того щоб оперувати ним як єдиним об’єктом. Для цього використовуються круглі дужки ( ).

(Xyz) + → Xyz, XyzXyz, XyzXyzXyz

- одне або більше повторень “Xyz”

Символьні класи

Зручними і лаконічними є спеціальні метасимволами, які позначають цілі символьні класи:

Символ	Опис
\d	Відповідає цифрі. Еквівалентно [0-9]
\D	Відповідає нецифровому символу. Еквівалентно [^0-9]
\s	Відповідає будь-якому пробільному символу. Еквівалентно [\f\n\r\t\v]
\S	Відповідає будь-якому непробільному символу. Еквівалентно [^ \f\n\r\t\v]
\w	Відповідає будь-якому літерному символу, цифровому й знаку підкреслення. Еквівалентно [[:word:]]
\W	Відповідає будь-якому символу, крім літерного символу, цифрового або підкреслення. Еквівалентно [^[:word:]]

Позиціонування

Наступні символи дозволяють позиціонувати регулярний вираз щодо елементів тексту: початку й кінця рядка, меж слова.

Представлення	Позиція	Приклад	Відповідність
^	Початок рядка	^a	aaa aaa
$	Кінець рядка	a$	aaa aaa
\b	Межа слова	a\b	aaa aaa

		\ba	aaa aaa
\B	Не межа слова	\Ba\B	aaa aaa
\G	Попередній успішний пошук	\Ga	aaa aaa (пошук зупинився на 4-й позиції — там, де не знайшлося a)

Регулярні вирази не є особливістю мови C++ чи якоїсь іншої, це загальний підхід до вирішення таких задач при роботі з текстом. Більшість мов програмування мають або вбудовану підтримку регулярних виразів, або мають відповідні бібліотеки. Починаючи з C ++ 11, C ++ забезпечує підтримку регулярних виразів за допомогою стандартної бібліотеки <regex>.

Регулярні вирази в C ++

Бібліотека <regex> регулярних виразів реалізує клас, який представляє регулярні вирази. Майже всі операції з регулярними виразами зводяться до операцій з наступними об'єктами:

Target sequence (цільова послідовність) - послідовність символів, у якій здійснюється пошук по шаблону. Це може бути діапазон, заданий двома ітераторами, рядок символів, що закінчується нуль-символом, або std::string.

Pattern (шаблон) — власне сам регулярний вираз. Він визначає умови, які мають виконатись, для того щоб певний рядок розглядався як збіг. Це об’єкт типу std::basic_regex, створений із рядка зі спеціальним синтаксисом.

Matched array (масив збігів). Інформацію про збіги можна отримати як об’єкт типу std::match_results.

Replacement string (рядок заміни) - це рядок, який визначає, як замінити збіги.

Функції для пошуку збігів

regex_match ()

Функція використовується для встановлення відповідності заданому шаблону. Повертає true, якщо регулярний вираз відповідає рядку. В іншому випадку функція повертає false.

regex_search ()

Функція використовується для спроби знайти збіг із шаблоном в будь-якій частині рядка. Функція шукає перше входження шаблона у рядок і повертає відповідний фрагмент рядка.

Отже, *regex_match()* буде успішно співставляти регулярний вираз з усією послідовністю символів, тоді як *regex_search()* буде шукати підрядок вхідного рядка, що відповідає регулярному виразу.

regex_replace()

Функція використовується для заміни рядка (підрядка), що відповідає регулярному виразу, вказаним у параметрах функції рядком.

Висновок

Regex використовується в пошукових системах для пошуку шаблонів, пошуку та заміни діалогових вікон таких програм, як текстові процесори та текстові редактори. Regex також використовується в утилітах UNIX, таких як sed, awk, а також для лексичного аналізу програми.

Ми бачили функції, які використовуються для узгодження, пошуку та заміни шаблонів у цьому посібнику. Використовуючи ці функції, ми в основному можемо розробити ефективний додаток, який реалізує бажану функціональність за допомогою регулярного виразу.

Regex дозволяє ефективно перевіряти вхідні дані, швидко та зручно шукати та замінювати рядок іншим рядком. Регулярні вирази дозволяють це зробити всього кількома рядками коду на C ++. Але слід розуміти, що реализація regex досить повільна, вона вимагає інтерпретації регулярних виразів і створення структури даних під час виконання програми, потребує динамічного виділення пам’яти. Тому потрібно уважно і з обережністю використовувати regex в циклах.

Джерела

1. https://en.cppreference.com/w/cpp/regex
2. https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D0%B8%D0%B9_%D0%B2%D0%B8%D1%80%D0%B0%D0%B7
3. https://www.regular-expressions.info/stdregex.html
4. https://regexr.com/
5. https://coding.tools/regex-tester
6. https://replit.com/