BÁO CÁO

Bùi Huỳnh Tây
MSSV: 24521589

25/03/2025

—

Cấu trúc dữ liệu và giải thuật -
IT003.P21.CTTN

—

GV Lý Thuyết: Duy Lê

GV Lý Thuyết: Nguyễn Thanh Sơn

GV Lý Thuyết: Phan Thế Duy

GV Lý Thuyết: Văn Thái Hùng

GV HDTH: Phan Minh Quân

GV HDTH: Trần Đình Khang

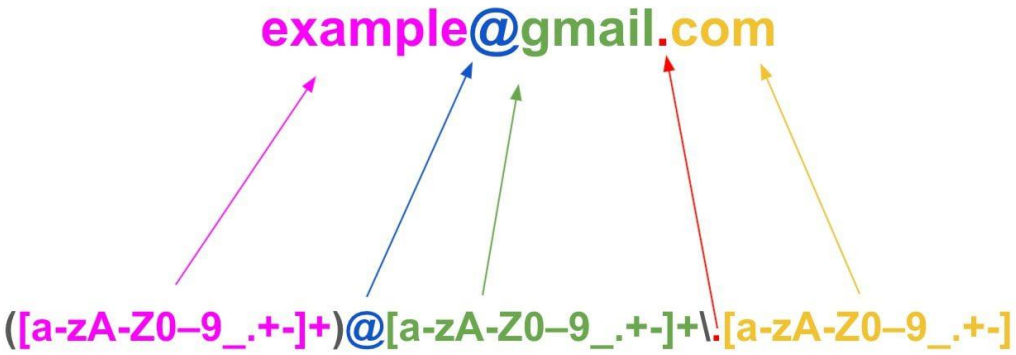
NỘI DUNG BÁO CÁO

Regular Expression (biểu thức chính quy), là các mẫu (pattern) dùng để tìm/thay thế (Find/Replace) và thao tác với chuỗi văn bản. Là một công cụ cực mạnh cho xử lí chuỗi trong Php, JavaScript…

Ví dụ: Khi kiểm tra tính hợp lệ của email hoặc số điện thoại thì ta có thể dùng là regex (viết tắt của Regular Expression)

Để xác định xem một chuỗi có phải là địa chỉ email hợp lệ không, chúng ta có thể sử dụng Regex để kiểm tra xem chuỗi đó có đúng định dạng email hay không, như là có chứa ký tự @, theo sau là tên miền hợp lệ. Một mẫu Regex đơn giản cho email có thể là: /^[^s@]+@[^s@]+.[^s@]+$/

Mẫu này kiểm tra rằng chuỗi bắt đầu với một hoặc nhiều ký tự không phải khoảng trắng hoặc @, sau đó là ký tự @, tiếp theo là một hoặc nhiều ký tự không phải khoảng trắng hoặc @, rồi đến dấu . và cuối cùng là một hoặc nhiều ký tự không phải khoảng trắng hoặc @.



Các cú pháp cơ bản Regular Expression

Dưới đây là bảng tóm tắt các cú pháp cơ bản của Regular Expression (Regex):

Ký hiệu	Ý Nghĩa	Ví Dụ	Giải Thích
.	Bất kỳ ký tự nào	a.c	Tìm các chuỗi như abc, a1c, nhưng không phải ac
d	Chữ số (0-9)	d{3}	Tìm ba chữ số liên tiếp, ví dụ: 123
D	Không phải chữ số	D{2}	Tìm hai ký tự không phải chữ số

Ký hiệu	Ý Nghĩa	Ví Dụ	Giải Thích
w	Ký tự chữ và số (a-z, A-Z, 0-9, _)	w+	Tìm một hoặc nhiều ký tự chữ và số
W	Ký tự không phải chữ và số	W+	Tìm một hoặc nhiều ký tự không phải chữ và số
s	Ký tự khoảng trắng (space, tab, line break)	s+	Tìm một hoặc nhiều ký tự khoảng trắng
S	Ký tự không phải khoảng trắng	S+	Tìm một hoặc nhiều ký tự không phải khoảng trắng
^	Bắt đầu chuỗi	^abc	Tìm chuỗi bắt đầu bằng abc
$	Kết thúc chuỗi	abc$	Tìm chuỗi kết thúc bằng abc
*	Lặp lại 0 hoặc nhiều lần	a*	Tìm 0 hoặc nhiều ký tự ‘a’
+	Lặp lại 1 hoặc nhiều lần	a+	Tìm 1 hoặc nhiều ký tự ‘a’
?	Tùy chọn (0 hoặc 1 lần)	a?	Tìm 0 hoặc 1 ký tự ‘a’
{n}	Lặp lại chính xác n lần	a{3}	Tìm ba ký tự ‘a’ liên tiếp
{n,}	Lặp lại ít nhất n lần	a{2,}	Tìm ít nhất hai ký tự ‘a’
{n,m}	Lặp lại từ n đến m lần	a{2,4}	Tìm từ hai đến bốn ký tự ‘a’
[]	Nhóm ký tự	[abc]	Tìm một ký tự trong nhóm ‘a’, ‘b’, hoặc ‘c’
[^]	Ký tự không nằm trong nhóm	[^a-c]	Tìm ký tự không phải ‘a’, ‘b’, hoặc ‘c’
`	`	Hoặc	`a
()	Nhóm ký tự	(abc)+	Tìm nhóm ‘abc’ lặp lại một hoặc nhiều lần

Ký hiệu	Ý Nghĩa	Ví Dụ	Giải Thích
	Escape ký tự đặc biệt	.	Tìm ký tự '.' (giá trị chính xác của dấu chấm)

Ứng dụng Regex

- Tìm kiếm: Xác định và tìm kiếm các mẫu cụ thể trong văn bản.
- Thay thế: Thay thế các phần của văn bản dựa trên mẫu.
- Xác thực: Kiểm tra xem dữ liệu có phù hợp với định dạng yêu cầu không, chẳng hạn như email, số điện thoại.
- Tách dữ liệu: Phân tích và tách dữ liệu từ văn bản dựa trên mẫu.

Cách sử dụng Regex trong C++

1. Các lớp chính trong <regex>

- std::regex: Biểu diễn một biểu thức chính quy.
- std::regex_match: Kiểm tra xem toàn bộ chuỗi có khớp với mẫu không.
- std::regex_search: Tìm kiếm mẫu trong chuỗi (khớp một phần).
- std::regex_replace: Thay thế các chuỗi con khớp với mẫu.
- std::smatch: Lưu trữ kết quả khớp từ std::string.
- std::cmatch: Lưu trữ kết quả khớp từ mảng const char*.

2. Ví dụ minh họa

a. Kiểm tra chuỗi có khớp mẫu không (regex_match)

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

int main() {
    std::string text = "hello123";
    std::regex pattern("^[a-z]+\\d+$"); // Mẫu: chữ thường + số

    if (std::regex_match(text, pattern)) {
        std::cout << "Chuỗi khớp mẫu!" << std::endl;
    } else {
        std::cout << "Chuỗi không khớp!" << std::endl;
    }

    return 0;
}
```

b. Tìm kiếm chuỗi con (regex_search)

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

int main() {
    std::string text = "Email: user@example.com, Phone: 123-456-7890";
    std::regex email_pattern(R"(\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\b)");

    std::smatch matches;
    if (std::regex_search(text, matches, email_pattern)) {
        std::cout << "Email tìm thấy: " << matches[0] << std::endl;
    }

    return 0;
}
```

c. Thay thế chuỗi (regex_replace)

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

int main() {
    std::string text = "Ngày 15/06/2023";
    std::regex date_pattern(R"((\d{2})/(\d{2})/(\d{4}))");
    std::string new_text = std::regex_replace(text, date_pattern, "$3-$2-$1"); // Định dạng lại ngày

    std::cout << "Chuỗi sau khi thay thế: " << new_text << std::endl;
    return 0;
}
```

Kết luận

Regex là một công cụ cực kỳ hữu ích trong lập trình, giúp xử lý văn bản một cách hiệu quả và tiết kiệm thời gian. Tuy nhiên, cần sử dụng một cách hợp lý để tránh phức tạp hóa vấn đề không cần thiết.

Lưu ý

- Sử dụng các công cụ kiểm tra Regex trực tuyến như Regex101 để test pattern trước khi tích hợp vào code.
- Viết comment giải thích rõ ràng cho các biểu thức phức tạp để dễ bảo trì sau này.
- Ưu tiên các phương pháp đơn giản nếu Regex khiến code trở nên khó hiểu.

Bài thu hoạch này đã trình bày tổng quan về Regular Expression.

Người thực hiện: [BÙI HUỲNH TÂY]
Ngày nộp: [28/03/2025]

Tài liệu tham khảo:

- Regular-Expressions.info
- Python re Documentation
- MDN Web Docs - Regular Expressions