Регулярные выражения

Регулярные выражения - формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании метасимволов.

Материал взят из стати Реализация механизма обработки регулярных выражений на языке C++

Ссылки

1. Online RE
2. RE to NFA

Принципы

В первую очередь, рассмотрим несколько принципов поиска по шаблону с помощью `PB`.

Для указания шаблона вы должны использовать форму записи, которую может обработать компьютер. Эта форма записи, или язык, в нашем случае - синтаксис `PB`.

Язык `PB` состоит из литералов и метасимволов. Литералы - это обычный текст, являющийся частью шаблона. Для описания отношений между литералами применяются следующие метасимволы (от имеющих наивысший приоритет к имеющим наименьший):

1. Квантификаторы (`closure`): строка одинаковых литералов, длина которой может меняться, либо литерал, появление которого не обязательно. (Одна из важнейших составляющих поиска по шаблону.)
2. Конкатенация (`concatenation`): Если в шаблоне два литерала, идущих друг за другом, то и в тексте будет осуществлён поиск символов, идущих друг за другом.
3. Объединение (конструкция выбора, `alteration`): Один из литералов, между которыми осуществляется выбор должен находиться в тексте, сравниваемом с шаблоном.
   В дополнение к вышеперечисленному, использование скобок позволяет влиять на последовательность обработки элементов `PB`.

В большинстве реализаций обработчиков `PB`, квантификаторы включают:

1. звёздочку (`*`), означающую повторение литерала нуль или более раз
2. плюс (`+`), означающий повторение литерала один или более раз
3. знак вопроса (`?`), означающий, что наличие того или иного литерала возможно, но не является обязательным условием совпадения шаблона с текстом
   Примеры: `A*` совпадает с пустой строкой, `"A"`, `"AA"`, `"AAA"` и т. д., `A+` совпадает с `"A"`, `"AA"`, `"AAA"` и т. д. `A?` совпадает с пустой строкой или `"A"`.

Для задания конкатенации нет необходимости использовать метасимволы. Строка, содержащая идущие друг за другом символы, является конкатенацией. Регулярное выражение `ABC`, например, совпадёт со строкой `"ABC"`.

Объединение (конструкция выбора, `alteration`) задаётся с помощью символа `"|"` между двумя `PB`. `A|B` совпадает либо с `"A"`, либо с `"B"`.

Синтаксис

Обычные символы

Большинство символов в регулярном выражении представляют сами себя за исключением специальных символов **\ / ^ $. | ? * + ( ) { }**, которые могут быть экранированы символом `\` (обратная косая черта) для представления самих себя в качестве символов текста. Можно экранировать целую последовательность символов, заключив её между `\Q` и `\E`.

Пример	Соответствие
`a\.?`	a. или a
`a\\\\b`	a\b
`a\[F\]`	a[F]
`\Q+-*/\E`	±*/

Аналогично могут быть представлены другие специальные символы (набор символов, требующих экранирования, может отличаться в зависимости от конкретной реализации). Часть символов, которые в той или иной реализации не требуют экранирования (например, угловые скобки `< >`), могут быть экранированы из соображений удобочитаемости.

Любой символ

Метасимвол `.` (точка) означает один любой символ, но в некоторых реализациях исключая символ новой строки.

Вместо символа `.` можно использовать `[\s\S]` (все пробельные и непробельные символы, включая символ новой строки).

Символьные классы

Набор символов в квадратных скобках `[ ]` именуется символьным классом и позволяет указать интерпретатору регулярных выражений, что на данном месте в строке может стоять один из перечисленных символов. В частности, `[абв]` задаёт возможность появления в тексте одного из трёх указанных символов, а `[1234567890]` задаёт соответствие одной из цифр. Возможно указание диапазонов символов: например, `[А-Яа-я]` соответствует всем буквам русского алфавита, за исключением букв `«Ё»` и `«ё»`.

Если требуется указать символы, которые не входят в указанный набор, то используют символ `^` внутри квадратных скобок, например

`[^0-9]` означает любой символ, кроме цифр.

Некоторые символьные классы можно заменить специальными метасимволами:

Символ	Эквивалент	Соответствие
`\d`	`[0-9]`	Цифровой символ
`\D`	`[^0-9]`	Нецифровой символ
`\s`	`[ \f\n\r\t\v]`	Пробельный символ
`\S`	`[^ \f\n\r\t\v]`	Непробельный символ
`\w`	`[[:word:]]`	Буквенный или цифровой символ или знак подчёркивания
`\W`	`[^[:word:]]`	Любой символ, кроме буквенного или цифрового символа или знака подчёркивания

Позиция внутри строки

Следующие символы позволяют спозиционировать регулярное выражение относительно элементов текста: начала и конца строки, границ слова.

Представление	Позиция
`^`	Начало текста (или строки при модификаторе `?m`)

`$`	Конец текста (или строки при модификаторе `?m`)
`\b`	Граница слова
`\B`	Не граница слова
`\G`	Предыдущий успешный поиск

Обозначение группы

Круглые скобки используются для определения области действия и приоритета операций. Шаблон внутри группы обрабатывается как единое целое и может быть квантифицирован. Например, выражение `(тр[ау]м-?)*` найдёт последовательность вида `трам-трам-трумтрам-трум-трамтрум`

Перечисление

Вертикальная черта разделяет допустимые варианты. Например, `gray|grey` соответствует `gray` или `grey`. Следует помнить, что перебор вариантов выполняется слева направо, как они указаны.

Если требуется указать перечень вариантов внутри более сложного регулярного выражения, то его нужно заключить в группу. Например, `gray|grey` или `gr(a|e)y` описывают строку `gray` или `grey`. В случае с односимвольными альтернативами предпочтителен вариант `gr[ae]y`, так как сравнение с символьным классом выполняется проще, чем обработка группы с проверкой на все её возможные модификаторы и генерацией обратной связи.

Квантификация

Квантификатор после символа, символьного класса или группы определяет, сколько раз предшествующее выражение может встречаться. Следует учитывать, что квантификатор может относиться более чем к одному символу в регулярном выражении, только если это символьный класс или группа.

Представление	Число повторений	Эквивалент
`?`	Ноль или одно	`{0,1}`
`*`	Ноль или более	`{0,}`
`+`	Одно или более	`{1,}`

Представление	Число повторений
`{n}`	Ровно `n` раз
`{m,n}`	От `m` до `n` включительно
`{m,}`	Не менее `m`
`{,n}`	Не более `n`

Часто используется последовательность `.*` для обозначения любого количества любых символов между двумя частями регулярного выражения.

Жадность

Жадный	Ленивый
`*`	`*?`

`+`	`+?`
`{n,}`	`{n,}?`

Также общей проблемой как жадных, так и ленивых выражений являются точки возврата для перебора вариантов выражения. Точки ставятся после каждой итерации квантификатора. Если интерпретатор не нашёл соответствия после квантификатора, то он начинает возвращаться по всем установленным точкам, пересчитывая оттуда выражение по-другому.

Ревнивая жадность

В отличие от обычной (жадной) квантификации, ревнивая (`possessive`) квантификация не только старается найти максимально длинный вариант, но ещё и не позволяет алгоритму возвращаться к предыдущим шагам поиска для того, чтобы найти возможные соответствия для оставшейся части регулярного выражения.

Использование ревнивых квантификаторов увеличивает скорость поиска, особенно в тех случаях, когда строка не соответствует регулярному выражению. Кроме того, ревнивые квантификаторы могут быть использованы для исключения нежелательных совпадений.

Жадный	Ревнивый
`*`	`*+`
`?`	`?+`
`+`	`++`

`{n,}`	`{n,}+`

`ab(xa)*+a` над **abxaa** bxaa, но не abxa **abxaa**, так как буква `a` уже занята

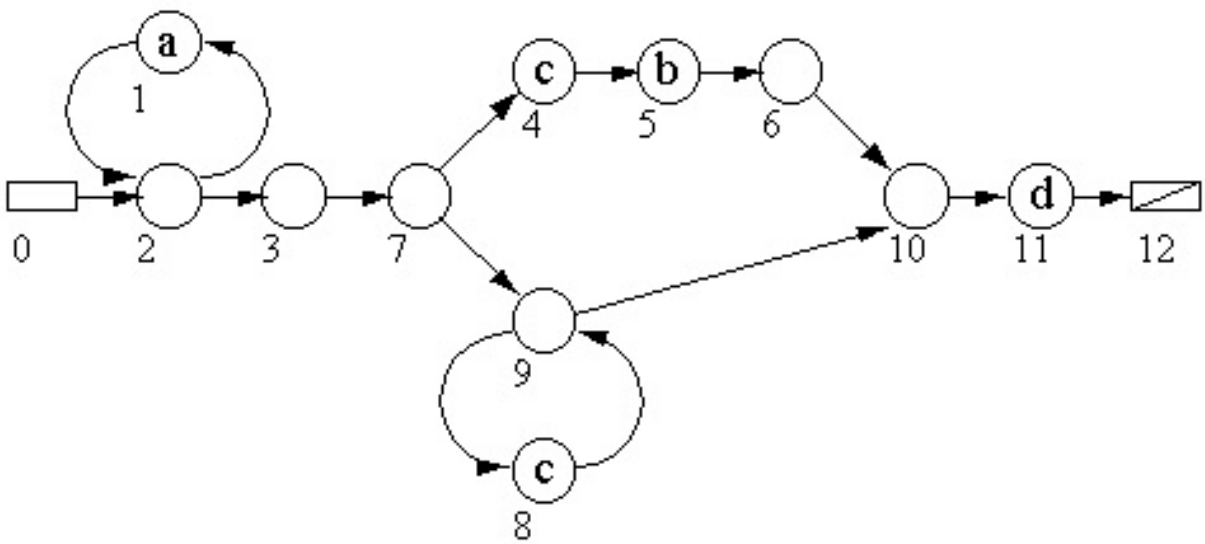
Автомат

Для поиска шаблона, заданного `PB`, вы не можете сравнивать каждый символ шаблона с каждым символом текста. Квантификаторы и объединение вызывают к жизни такое количество путей срабатывания сложных шаблонов, что использование “обычного” алгоритма невозможно. Должен быть применен более действенный подход. Лучший путь - построить автомат и смоделировать его работу. Для описания поискового шаблона, заданного регулярным выражением, вы можете использовать недетерминированный и детерминированный конечные автоматы.

Автомат может переходить в несколько состояний. Он может переходить из одного состояния в другое, в зависимости от события, которое поступило ему на вход, в нашем случае, - очередной входной символ. Здесь же видна разница между детерминированными и недетерминированными конечными автоматами. Детерминированный автомат может принять только одно состояние в ответ на определённый символ, а недетерминированный автомат может перейти в одно из нескольких состояний в ответ на один и тот же входной символ.

Оба типа автоматов допустимо использовать для задания любого `PB`. Каждый из этих типов автоматов имеет достоинства и недостатки. Для всех, кто хочет знать больше об этих автоматах в их связи с `PB`. В нашей реализации мы будем использовать недетерминированный автомат. Это наиболее востребованная стратегия реализации алгоритма обработки `PB`. Более того, сконструировать недетерминированный автомат по PB несколько проще, чем детерминированный.

На рисунке показана диаграмма переходов недетерминированного конечного автомата для шаблона `a*(cb|c*)d`. Она содержит все типы операций над регулярными множествами: итерацию, конкатенацию, объединение. Заметьте, что скобка, содержащая объединение, является литералом для операции конкатенации. Начальное состояние представлено прямоугольником в левой части рисунка. Допускающее состояние автомата показано в правой части рисунка в виде прямоугольника, перечёркнутого по диагонали.



Этот пример очень хорошо демонстрирует проблемы обработки шаблонов. В состоянии под номером `7` не ясно какое состояние будет следующим для входного символа `"c"`. Состояния `4` и `9` - возможные варианты. Автомат должен выбрать правильный путь.

Если на предмет совпадения с шаблоном должна быть исследована строка, содержащая текст `"aaccd"`, автомат стартует из состояния номер `0` - начального состояния. Следующее состояние, номер `2`, является нулевым. Иными словами, для перехода автомата в это состояние не нужны символы, совпадающие с шаблоном.

Первый символ на входе - `"a"`. Автомат переходит в состояние под номером `1`, это единственный путь. После успешного совпадения части шаблона с символом `"a"` будет прочитан очередной символ и автомат снова перейдёт в состояние `2`. Для следующего входного символа, тоже `"a"`, крайние два перехода повторяются. После этого, единственный допустимый путь - переход в состояние под номером `3` и `7`.

Мы оказались в состоянии, которое может вызвать проблемы. Следующий символ на входе - `"c"`. Здесь мы видим силу автомата. Он может догадаться, что правильный путь будет через состояние номер `9`, но не `4`. Это - душа недетерминированной стратегии: возможные решения находятся. Они не описываются алгоритмом, работающим “шаг за шагом”.

Разумеется, в реальном мире программирования мы должны

перебирать все возможные пути. Однако, о практической стороне дела поговорим несколько позже.

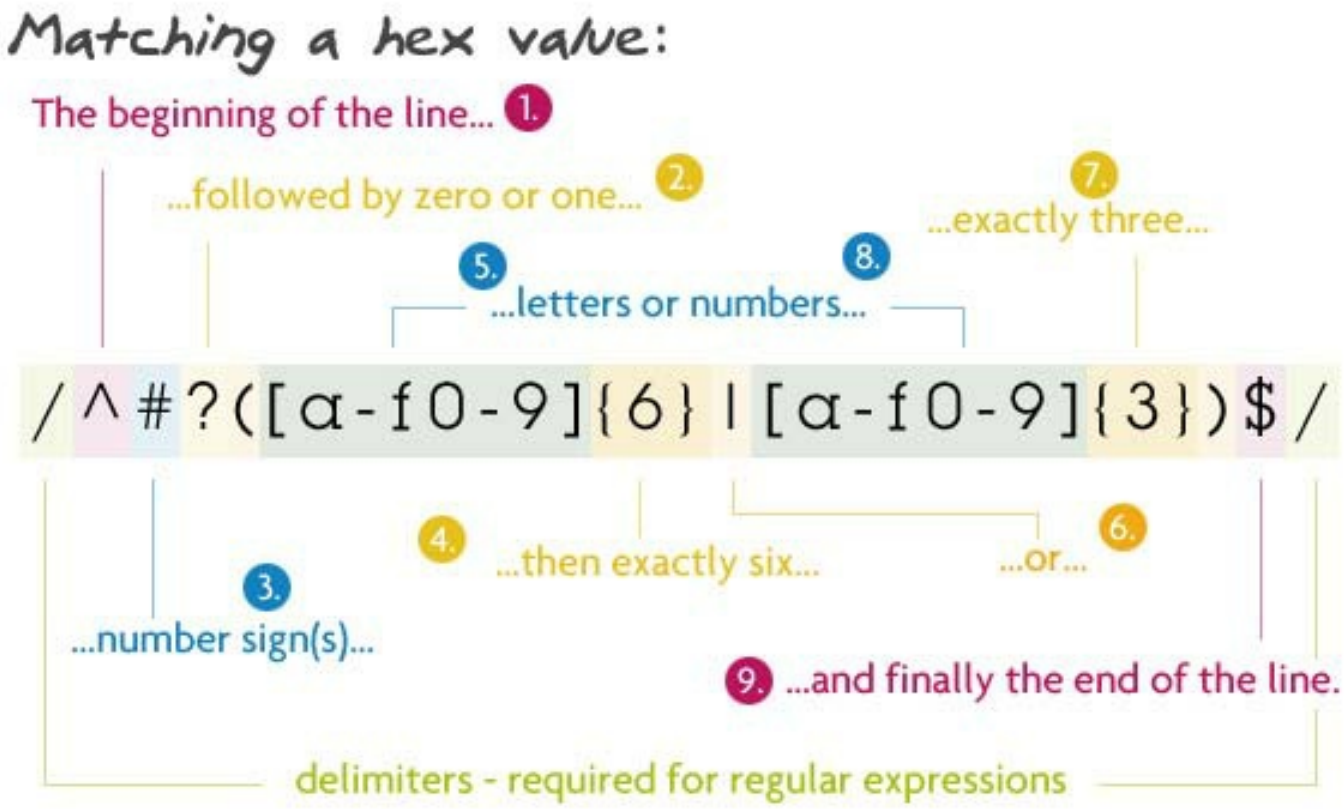
После достижения состояния под номером `9`, автомат осуществляет переход из `9` в `8` (первое `"c"` на входе), в `9`, в `8` (второе `"c"` на входе), `10` и `11` (совпадение с `"d"`) в допускающее состояние под номером `12`. Допускающее состояние достигнуто, следовательно, текст `"aaccd"` совпадает с шаблоном "`a*(cb|c*)d"`.

Примеры типичных регулярных выражений

Материал взят 8 Regular expressions you should know

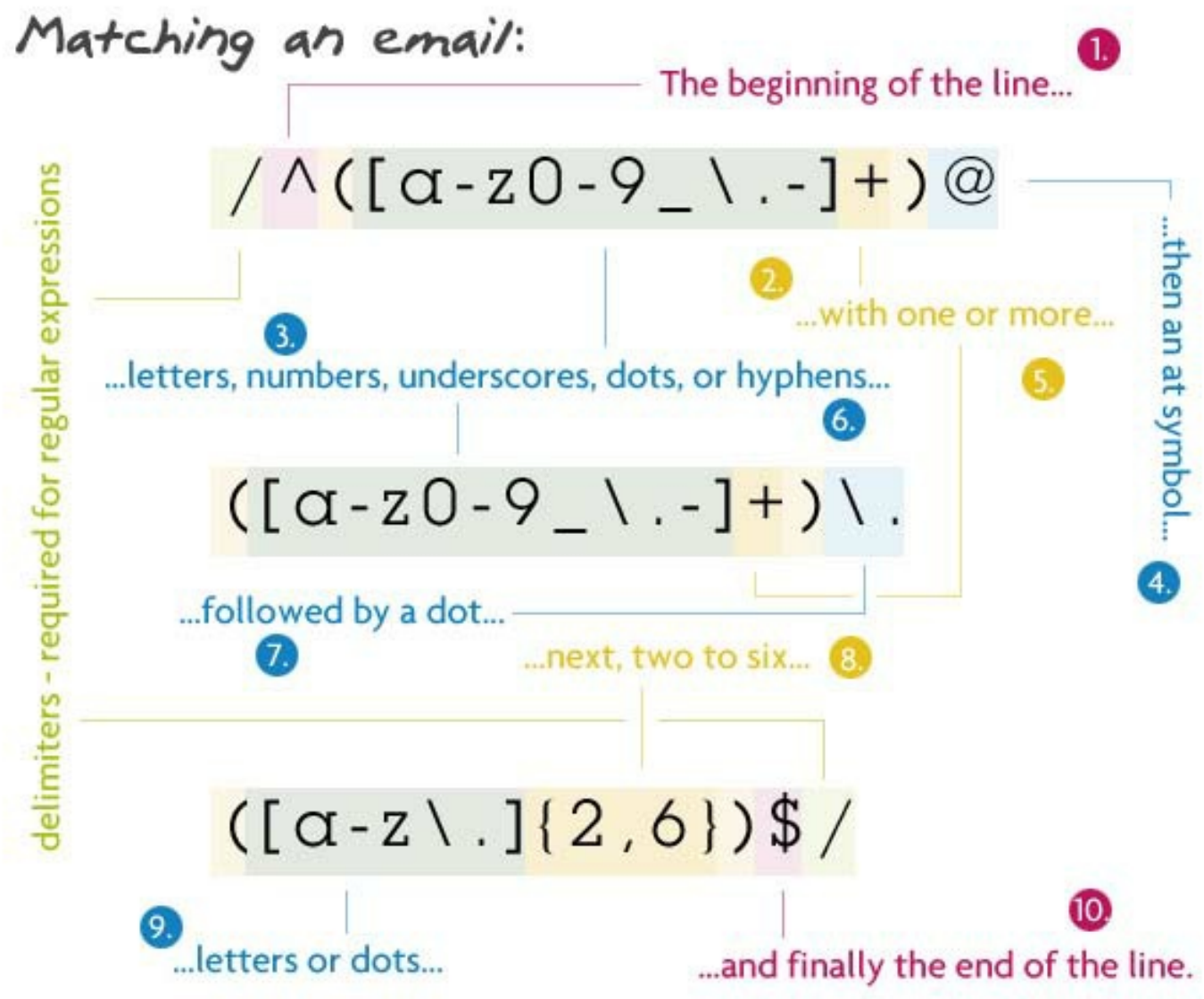
Matching a Hex Value

Pattern `/^#?([a-f0-9]{6}|[a-f0-9]{3})$/`



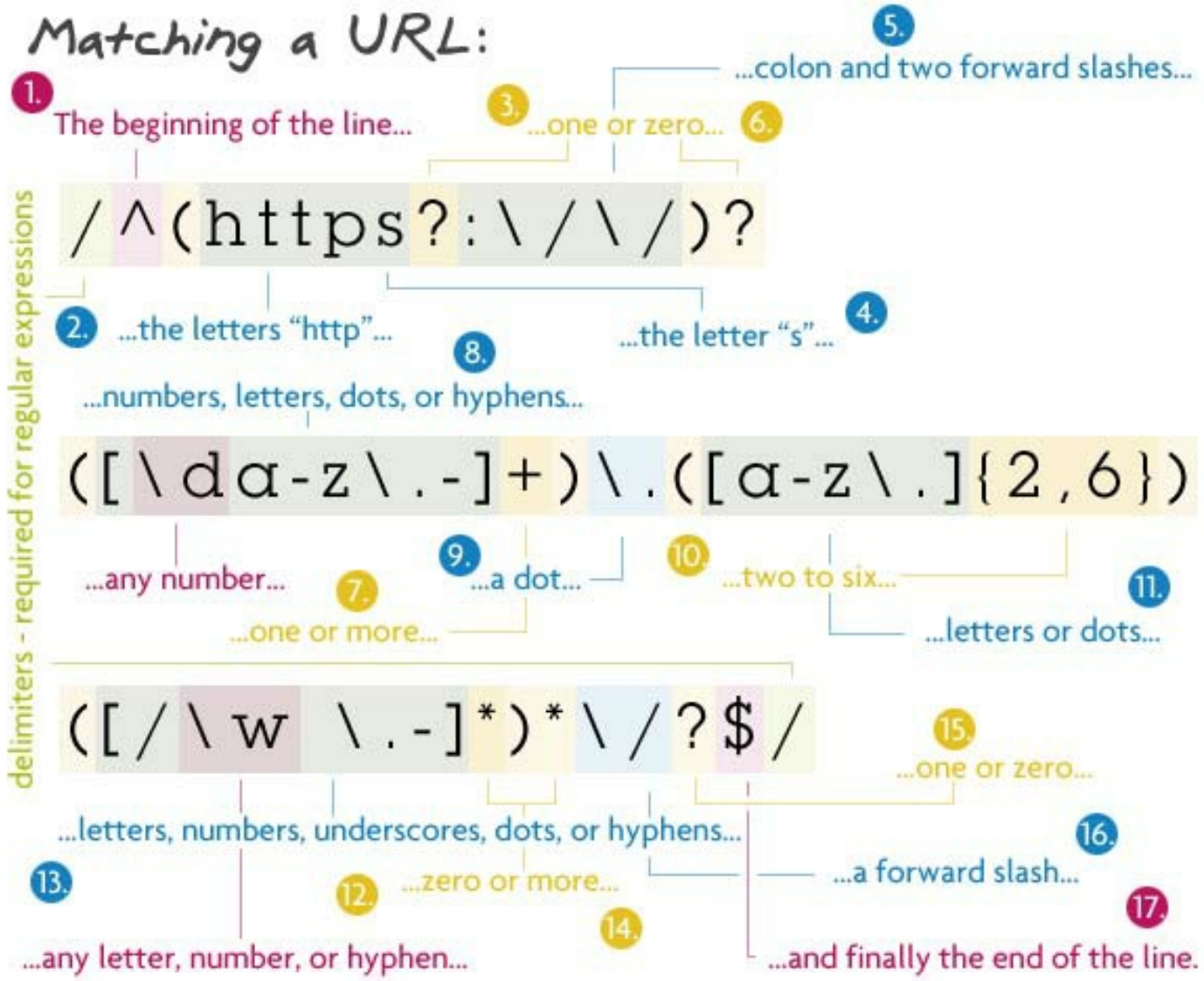
Matching an Email

Pattern `/^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/`



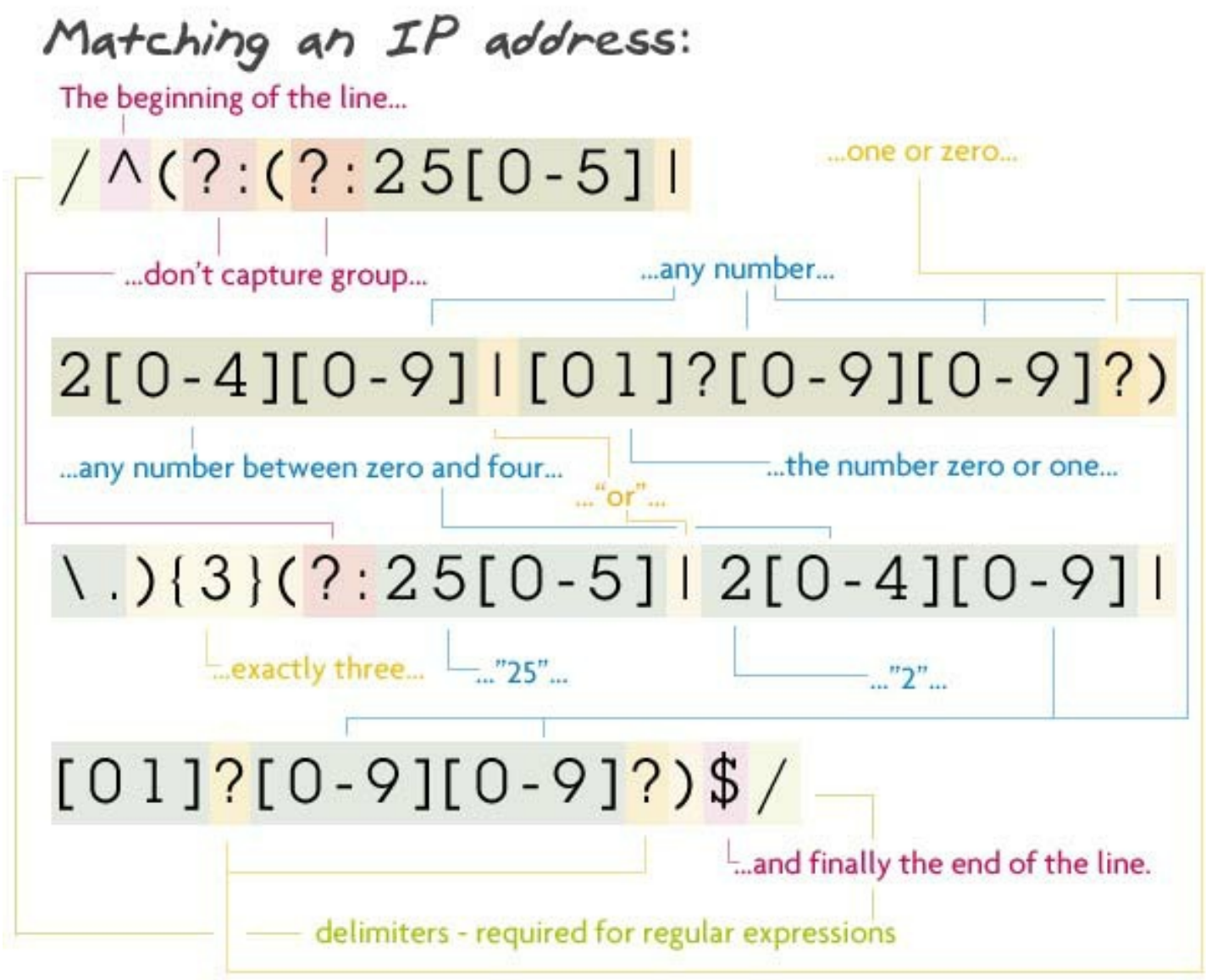
Matching a URL

Pattern `/^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?$/`



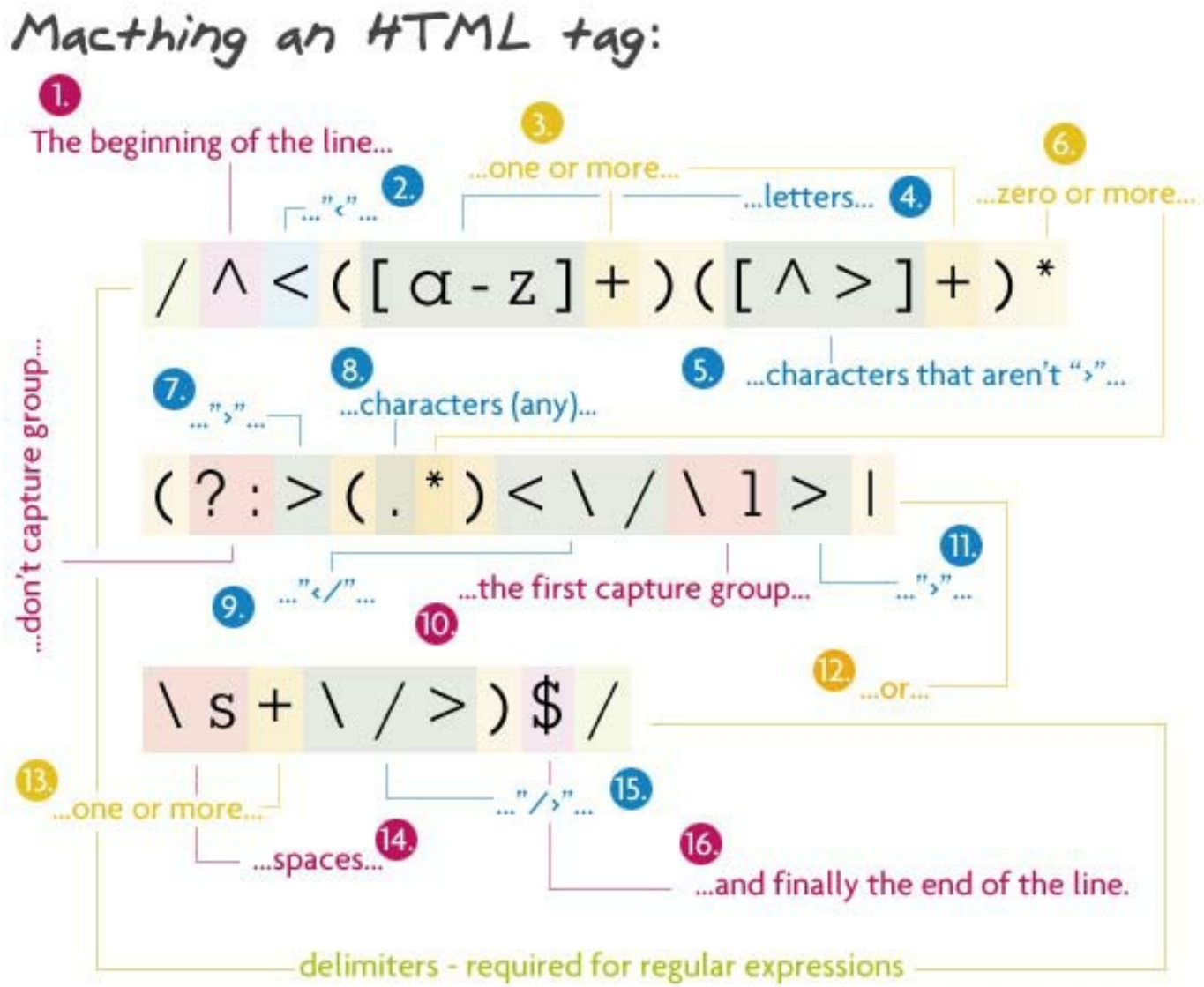
Matching an IP Address

Pattern `/^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/`



Matching an HTML Tag

Pattern `/^<([a-z]+)([^<]+)*(?:>(.*)<\/\1>|\s+\/>)$/`


Macthing an HTML tag:
1.
The beginning of the line...
2.
..."<"...
3.
...one or more...
4.
...letters...
5.
...characters that aren't ">"...
6.
...zero or more...
/ ^ < ([a - z] +) ([^ >] +) *
7.
..."<"...
8.
...characters (any)...
(? : > (. *) < \ / \ 1 > |
...don't capture group...
9.
..."</"...
10.
...the first capture group...
11.
..."<"...
12.
...or...
\ s + \ / >) $ /
13.
...one or more...
14.
...spaces...
15.
..."/>"...
16.
...and finally the end of the line.
delimiters - required for regular expressions

Шпаргалка

Регулярные выражения

Якоря

^	Начало строки +
\A	Начало текста +
$	Конец строки +
\Z	Конец текста +
\b	Граница слова +
\B	Не граница слова +
\<	Начало слова
\>	Конец слова

Символьные классы

\c	Управляющий символ
\s	Пробел
\S	Не пробел
\d	Цифра
\D	Не цифра
\w	Слово
\W	Не слово
\xhh	Шестнадцатиричный символ hh
\Oxxx	Восьмиричный символ xxx

Символьные классы POSIX

[:upper:]	Буквы в верхнем регистре
[:lower:]	Буквы в нижнем регистре
[:alpha:]	Все буквы
[:alnum:]	Буквы и цифры
[:digit:]	Цифры
[:xdigit:]	Шестнадцатиричные цифры
[:punct:]	Пунктуация
[:blank:]	Пробел и табуляция
[:space:]	Пустые символы
[:cntrl:]	Управляющие символы
[:graph:]	Печатные символы
[:print:]	Печатные символы и пробелы
[:word:]	Буквы, цифры и подчеркивание

Утверждения

?=	Вперед смотрящее +
?!	Отрицательное вперед смотрящее +
?<=	Назад смотрящее +
?!= или ?	Отрицательное назад смотрящее +
?>	Однократное подвыражение
?()	Условие [если, то]
?()|	Условие [если, то, а иначе]
?#	Комментарий

Примечание *Отмеченное + работает в большинстве языков программирования.*

Образцы шаблонов

([A-Za-z0-9-]+)	Буквы, числа и знаки переноса
(\d{1,2}\/\d{1,2}\/\d{4})	Дата (напр., 21/3/2006)
([^\s]+(?=\.(jpg|gif|png))\.\2)	Имя файла jpg, gif или png
(^[1-9]{1}$\|^[1-4]{1}[0-9]{1}$|^50$)	Любое число от 1 до 50 включительно
(#?([A-Fa-f0-9]){3}(([A-Fa-f0-9]){3})?)	Шестнадцатиричный код цвета
((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,15})	От 8 до 15 символов с минимум одной цифрой, одной заглавной и одной строчной буквой (полезно для паролей).
(\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,6})	Адрес email
(\<(/?[^\>]+)\>)	HTML теги

Примечание *Эти шаблоны предназначены для ознакомительных целей и основательно не проверялись. Используйте их с осторожностью и предварительно тестируйте.*

Кванторы

*	0 или больше +
*?	0 или больше, нежадный +
+	1 или больше +
+?	1 или больше, нежадный +
?	0 или 1 +
??	0 или 1, нежадный +
{3}	Ровно 3 +
{3,}	3 или больше +
{3,5}	3, 4 или 5 +
{3,5}?	3, 4 или 5, нежадный +

Специальные символы

\	Экранирующий символ +
\n	Новая строка +
\r	Возврат каретки +
\t	Табуляция +
\v	Вертикальная табуляция +
\f	Новая страница +
\a	Звуковой сигнал
[\b]	Возврат на один символ
\e	Escape-символ
\N{name}	Именованый символ

Подстановка строк

$n	n-ая непассивная группа
$2	«xyz» в /^(abc(xyz))$/
$1	«xyz» в /^(?:abc)(xyz)$/
$`	Перед найденной строкой
$'	После найденной строки
$+	Последняя найденная строка
$&	Найденная строка целиком
$_	Исходный текст целиком
$$	Символ «$»

Диапазоны

.	Любой символ, кроме переноса строки (\n) +
(a|b)	a или b +
(...)	Группа +
(?:...)	Пассивная группа +
[abc]	Диапазон (a или b или c) +
[^abc]	Не a, не b и не c +
[a-q]	Буква между a и q +
[A-Q]	Буква в верхнем регистре между A и Q +
[0-7]	Цифра между 0 и 7 +
\n	n-ая группа/подшаблон +

Примечание *Диапазоны включают граничные значения.*

Модификаторы шаблонов

g	Глобальный поиск
i	Регистронезависимый шаблон
m	Многострочный текст
s	Считать текст одной строкой
x	Разрешить комментарии и пробелы в шаблоне
e	Выполнение подстановки
U	Нежадный шаблон

Мета-символы (экранируются)

^	[	.
$	{	*
(	\	+
)	|	?
<	>	

Эта таблица доступна на www.exlab.net
Англоязычный оригинал на AddedBytes.com