

REGULAR EXPRESSIONS & PATTERN MATCHING ABBREVIATED

(Mark E. Donaldson)

A **regular expression** is a pattern (a template) to be matched against a string. Regular expressions are also referred to as **regex**. Matching a regular expression against a string either succeeds or fails. Some parts of the **pattern** match **single characters** in the string of a particular type, while other parts of the pattern may match **multiple characters**. These are called **grouping patterns**.

REGULAR EXPRESSION OPERATORS	
Operator	Description and Example
<p>m/ /</p> <p>or</p> <p>/ /</p>	<p>Match operator. The match operator consists of a slash, a regular expression, and another slash. Enclosing a string between slashes (the match operator) makes the string a regular expression. The <code>\$_</code> variable is then tested against the regular expression. If the regular expression matches, the match operator returns true. Otherwise, it returns false.</p> <p>Note: The "m" is optional if the <code>//</code> is used for the delimiter.</p> <p>if (/abc/)</p> <p>means if an "abc" are found together in succession, then do something.</p>
s/ / /	<p>Substitute operator. The substitute operator replaces part of the string that matches the regular expression with another string. The substitute operator consists of the letter s, a slash, a regular expression, a slash, a replacement string, and a final slash. The variable <code>\$_</code> is tested against the regular expression. If there is a match, the replacement string is substituted for the regular expression.</p> <p>if (s/abc/cba/)</p> <p>means if an "abc" are found together in succession, then replace them with a "cba".</p>
tr/ / /	<p>Transliterate operator. The transliterate operator consists of the letters tr, a slash, an old string, a slash, a new string, and a final slash. The transliterate operator modifies the contents of the <code>\$_</code> variable (just like the substitute operator), looking for characters of the old string within the <code>\$_</code> variable. All such characters found are replaced with the corresponding characters of the new string. Returns the number of characters changed: <code>\$count = changed characters</code>.</p>

	<p>tr/ab/ba/ then tr/a-z/ /</p> <p>means replace all "a"s with a "b", and all "b"s with an "a", and then count all lower case characters.</p>
/d	<p>Delete option. Used with "tr" to prevent replication. Any character that matches in the old string without a corresponding character in the new string is removed from the string.</p> <p>tr/a-z/ABCDE/d</p> <p>means all lower case characters "a" to "e" will be capitalized and the remaining lower case characters will be deleted.</p>
/c	<p>Complement option. Used with "tr" to complement the old string. Any character listed in the old string is removed from the set of all possible characters. The remaining characters form the resulting old string.</p> <p>tr/a-z/_/c then tr/a-z/ /cd</p> <p>means underscore all non-lowercase letters, and then delete all non-lower case letters.</p>
/s	<p>Squeeze option. Used with "tr" to squeeze multiple consecutive copies of the same resulting translated letter into one copy.</p> <p>tr/a-z/X/s</p> <p>means replace all words not capitalized with an X.</p>
/g	<p>Global option. Used with "m" or "s" to substitute as many times as possible on a line.</p>
/i	<p>Ignore Case option. Used with "m" or "s" to ignore case in the pattern to match.</p> <p>if (<STDIN> =~ /^Y/)</p> <p>means if input is a "y" or "Y" then do something.</p>
/\Q\E /	<p>Quoting Escape option.</p> <p>\$string = "[box]"; if (/AQ\$string\E/)</p> <p>means \$string turns into <code>\[box\]</code>, making the match look for a literal pair of enclosing brackets.</p>
=~	<p>Different Target operator. Takes a regular expression operator on the right side, but changes the target string or value</p>

	<p>on the left side to something other than \$_. if (<STDIN> =~ /^[yY]/) means if input is a “y” or “Y” then do something.</p>
--	--

SINGLE CHARACTER PATTERNS

Pattern	Description and Example
.	<p>Dot Character. Matches any single character except newline (\n). <i>/a./</i> means it matches any two letter sequence that starts with an “a” and is not “a\n”.</p>
[]	<p>Character Class. Represented by a pair of open and close square brackets and a list of characters between the brackets. One and only one of these characters must be present at the corresponding part of the string for the pattern to match. Ranges may be depicted with a dash (-), such as [a-z] or [1-9]. <i>/[abcde]/</i> means it matches a string containing any one of the first five letters of the lowercase alphabet.</p>
[^]	<p>Negated Character Class. Same as character class but has a leading up arrow (or caret ^) immediately after the left bracket. This character class matches any single character that is not in the list. <i>/[^abcde]/</i> means it matches a string containing any one of the lowercase alphabet member that is not one of the first five.</p>
\d	Digit Class. <i>[0-9]</i>
\w	Word Char Class. <i>[a-zA-Z1-9_]</i>
\s	Space Char Class. <i>[r\ t\n\f]</i>
\D	Not Digit Class. <i>[^0-9]</i>
\W	Not Word Char Class. <i>[^a-zA-Z1-9_]</i>
\S	Not Space Char Class. <i>[^\r\ t\n\f]</i>

GROUPING CHARACTER PATTERNS

Multipliers	Description and Example
*	<p>Asterisk. Zero or more of the immediately previous character or character class.</p>
+	<p>Plus Sign. One or more of the immediately previous character or character class.</p>

?	<p>Question Mark. Zero or one of the immediately previous character or character class.</p>
{, }	<p>General Multiplier. Consists of a pair of matching curly braces with one or two numbers separated by a comma inside. The immediately preceding character must be found within the indicated number of repetitions. If you leave off the second number, you indicate “that many or more”. If you leave off the comma, you indicate “exactly that many”. <i>/x{0, 5}/</i> <i>/x{5, }/</i> <i>/x{5}/</i> means 0 to 5 x’s, then 5 or more x’s, then exactly 5 x’s.</p>
Constructs	Description and Example
()	<p>Parenthesis As Memory. Open and closed parenthesis around any part of a pattern causes the part of the string matched by the pattern to be remembered so that it can be recalled or referenced later. <i>/a(.) b(*)\1\2/</i> means match, memorize, and recall (\1\2) an “a” followed by one or more “b”s.</p>
	<p>Alternation. Matches exactly one of the alternatives. Acts as a logical OR. <i>/a b c/</i> then <i>/man woman/</i> means match “a” or “b” or “c”, then match “man” or “woman”.</p>
Anchors	Description and Example
\$	<p>Dollar Sign Anchor. Matches a pattern at the end of a string. <i>/.txt\$/</i> means match “.txt” if it occurs at the end of a string.</p>
^	<p>Caret Anchor. Matches a pattern at the end of a string. <i>/^abc/</i> means match “abc” if it occurs at the beginning of a string.</p>
\b	Word Boundary Anchor.
\B	Not Word Boundary Anchor.