in **Subpatterns** on page IV-186), "succeeds" means matching the rest of the main pattern as well as the alternative in the subpattern.

## Match Option Settings

Character matching options can be changed from within the pattern by a sequence of Perl option letters enclosed between `(?` and `)`. The option letters are:

| Option | PCRE Name | Characters Matched |
|---|---|---|
| i | PCRE_CASELESS | Upper and lower case. |
| m | PCRE_MULTILINE | ^ matches start of string and just after a new line (\n). $ matches end of string and just before a new line. Without (?m), ^ and $ match only the start and end of the entire string. |
| s | PCRE_DOTALL | . matches all characters including newline. Without (?s), . does not match newlines. See **Matching Newlines** on page IV-184. |
| U | PCRE_UNGREEDY | Reverses the "greediness" of the quantifiers so that they are not greedy by default, but become greedy if followed by ?. |

For example, `(?i)` sets caseless matching.

You can unset these options by preceding the letter with a hyphen: (?-i) turns off caseless matching.

You can combine a setting and unsetting such as `(?i-U)`, which sets PCRE_CASELESS while unsetting PCRE_UNGREEDY.

When an option change occurs at top level (that is, not inside subpattern parentheses), the change applies to the remainder of the pattern that follows. If the change is placed right at the start of a pattern, PCRE extracts it into the global options.

An option change within a subpattern affects only that part of the current pattern that follows it, so

```
(a(?i)b)c
```

matches abc and aBc and no other strings.

Any changes made in one alternative do carry on into subsequent branches within the same subpattern. For example,

```
(a(?i)b|c)
```

matches "ab", "aB", "c", and "C", even though when matching "C" the first branch is abandoned before the option setting. This is because the effects of option settings happen at compile time.

The other PCRE 5-specific options, such as (?e), (?A), (?D), (?S), (?x) and (?X), are implemented but are not very useful with Igor's regular expression commands.

## Matching Newlines

Igor generally uses carriage-return (ASCII code 13) return to indicate a new line in text. This is represented in a literal string by "\r". For example:

```
Print "Hello\rGoodbye"
```

prints two lines of text to the history area.

However, in regular expressions, the newline character is linefeed (ASCII code 10). This is represented in a literal string by "\n". Carriage-return has no special status in regular expressions.

The match option setting `(?s)` changes the default behavior of dot so that it matches any character including newline. The setting can appear anywhere before the dot in the pattern. For example:

```
Function DemoNewline(includeNewline)
   Variable includeNewline

   String subject = "Hello\nGoodbye"   // \n represents newline

   String regExp
   if (includeNewline)
      regExp = "(?s)(.+)"     // One or more of any character
   else
      regExp = "(.+)"         // One or more of any character except newline
   endif

   String result
   SplitString /E=(regExp) subject, result
   Print result
End
```

The output from DemoNewline(0) is:

```
   Hello
```

The output from DemoNewline(1) is:

```
   Hello\nGoodbye
```

Here is a more realistic example:

```
Function DemoDotAll()
   String theString = ExampleHTMLString()

   // This regular expression attempts to extract items from
   // HTML code for an ordered list. It fails because the HTML
   // code contains newlines and, by default, . does not match newlines.
   String regExpFails="<li>(.*?)</li>.*<li>(.*?)</li>"
   String str1, str2
   SplitString/E=regExpFails theString, str1, str2
   Print V_flag, " subpatterns were matched using regExpFails"
   Printf "\"%s\", \"%s\"\r", str1, str2

   // This regular expression works because the "(?s)" match
   // option setting causes . to match newlines.
   String regExpWorks="(?s)<li>(.*?)</li>.*<li>(.*?)</li>"
   SplitString/E=regExpWorks theString, str1, str2
   Print V_flag, " subpatterns were matched using regExpWorks"
   Printf "\"%s\", \"%s\"\r", str1, str2
End
Function/S ExampleHTMLString()   // Returns HTML code containing newlines
   String theString = ""
   theString += "<ol>\n"
   theString += "\t<li>item 1</li>\n"
   theString += "\t<li>item 2</li>\n"
   theString += "<\ol>\n"
   return theString
End

•DemoDotAll()
  0   subpatterns were matched using regExpFails
  "", ""
  2 subpatterns were matched using regExpWorks
  "item 1", "item 2"
```