`$reg` can also be interpolated into a larger regexp:

```
$x =~ /(abc)?$reg/;  # still matches
```

As with the matching operator, the regexp quote can use different delimiters, e.g., `qr!!`, `qr{}` and `qr~~`. The single quote delimiters `qr'` prevent any interpolation from taking place.

Pre-compiled regexps are useful for creating dynamic matches that don't need to be recompiled each time they are encountered. Using pre-compiled regexps, `simple_grep` program can be expanded into a program that matches multiple patterns:

```
% cat > multi_grep
#!/usr/bin/perl
# multi_grep - match any of <number> regexps
# usage: multi_grep <number> regexp1 regexp2 ... file1 file2 ...

$number = shift;
$regexp[$_] = shift foreach (0..$number-1);
@compiled = map qr/$_/, @regexp;
while ($line = <>) {
    foreach $pattern (@compiled) {
        if ($line =~ /$pattern/) {
            print $line;
            last;  # we matched, so move onto the next line
        }
    }
}
^D

% multi_grep 2 last for multi_grep
    $regexp[$_] = shift foreach (0..$number-1);
        foreach $pattern (@compiled) {
                last;
```

Storing pre-compiled regexps in an array `@compiled` allows us to simply loop through the regexps without any recompilation, thus gaining flexibility without sacrificing speed.

7.3.3 Embedding comments and modifiers in a regular expression

Starting with this section, we will be discussing Perl's set of **extended patterns**. These are extensions to the traditional regular expression syntax that provide powerful new tools for pattern matching. We have already seen extensions in the form of the minimal matching constructs `??`, `*?`, `+?`, `{n,m}?`, and `{n,}?`. The rest of the extensions below have the form `(?char...)`, where the `char` is a character that determines the type of extension.

The first extension is an embedded comment `(?#text)`. This embeds a comment into the regular expression without affecting its meaning. The comment should not have any closing parentheses in the text. An example is

```
/(?# Match an integer:)[+-]?\d+/;
```

This style of commenting has been largely superseded by the raw, freeform commenting that is allowed with the `//x` modifier.

The modifiers `//i`, `//m`, `//s`, and `//x` can also embedded in a regexp using `(?i)`, `(?m)`, `(?s)`, and `(?x)`. For instance,