

## Regular Expressions

by Dr. David J. Ritchie,  
Computer Club Advisor

Regular expressions (RE's) are patterns of letters and numbers. You can recognize (match) using regular expressions.

### Try it:

Write this program:

```
print "Enter: \n";
Sp = <STDIN>;
if ($p =~ m/ab. 12/) {
    print "Match! \n";
} else {
    print "No
Match! \n";
}
```

### Questions:

1. What entries match the pattern:  
`/ab.12/`
2. What if you use two dots instead?

## Match Info

. = match single character  
a = match "a"  
b = match "b", etc.  
^ = begin of string  
\$ = end of string  
previous character matches:  
\* = match 0 or more times  
+ = match 1 or more times  
? = match 0 or 1 time

Group: (az2) = match group "az2"  
[aeiou] = match one of (brackets).  
[a-z] = match a through z (brackets).

\d = digit= [0-9].

\D = non-digit= [^0-9]

\w = word char= [a-zA-Z0-9\_]

\W = non-word= [^a-zA-Z0-9\_]

\s = space char= [ \r\t\n\f]

\S = non-space= [^\r\t\n\f]

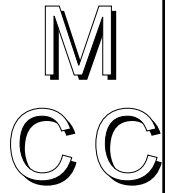
Issue 4

January 24, 2000

# Madison Computer Club News

Mrs. Gilmore

Dr. David J. Ritchie



## Students to play with Matches!

This week's newsletter is about making your program more intelligent in accepting answers. You do this by using regular expressions and "match". See the sidebar for more about regular expressions.

You can test for a match by using `m/pattern/`. This will test for a match with the regular expression given by `pattern`.

You can specify a lot of different patterns. There are also ways to specify special characters.

**Examples:** \r = return, \t=tab, \n= new line, \f=form feed ^ = match not what follows.

**Vertical Bar:** `red | blue` = match "red" or "blue".

## Substituting Characters

Matches allows a program to recognize letters and numbers.

forward slashes after the m.

By using the rules for constructing regular expressions, you can make the program recognize all kinds of text that is typed.

### A Piece of a Better Choose subroutine

```
if ($Response =~ m/left|L|l/) {
    $NeedResponse = 0;
    $ChosenPage = $Apage;
} elsif ($Response =~ m/right|R|r/) {
    $NeedResponse = 0;
    $ChosenPage = $Bpage;
} elsif ($Response =~ m/quit|Q|q/) {
    $NeedResponse = 0;
    $ChosenPage = -1;
} else {
    print "\nEnter Left or Right: ";
    $NeedResponse = 1;
}
```

If you need to allow more choices than simply right or left, you will have to extend the regular expression pattern.

Remember that in your program the person must type A, B, or Q to order to do choice A, B, or Quit.

Wouldn't it be better to allow the person to type left, L, or l to go left, right, R, or r, to go right, and Quit, Q, or q to quit?

The statements in the insert above test the contents of the \$Response variable for a match on several patterns.

The patterns are those possibilities listed between

### Take your time with regular expressions:

Regular expressions can be very difficult to understand. Don't expect yourself to master them right away.

Try to use them in simple ways at first. They can be very powerful and save you hours of writing other code!