Regular expressions

Each character matches itself, except: + ? . * ^ $ ( ) [ ] { } | \

A \ before a special character escapes its special meaning.

.	matches any single character except a newline
^	beginning of a line
$	end of a line
[abc]	matches any of the enclosed characters
[^abc]	matches any character that is not enclosed
[a-m]	matches any character in this range
(...)	groups a series of pattern elements into a single element
(...|...|...)	matches one of the alternatives

How many consecutive matches?

*	matches preceding pattern element zero or more times
+	matches preceding pattern element one or more times
?	matches preceding pattern element zero or one times
{N,M}	matches preceding pattern element between N and M times
{N}	matches preceding pattern element exactly N times
{N,}	matches preceding pattern element at least N times

Character classes

Abbrev.	Equiv. pattern	Matches
\d	[0-9]	a digit
\D	[^0-9]	a non-digit
\w	[a-zA-Z_0-9]	an alphanumeric character, or underscore
\W	[^a-zA-Z_0-9]	a non-alphnumeric character
\s	[\t\n\r\f]	a whitespace character
\S	[^ \t\n\r\f]	a non-whitespace character

match1.pl

```
@seqs = <DATA>;
foreach $a ( @seqs ) {
        chomp($a);
        print($a);
        if ( $a =~ /ACCCC[AG][AG][AG]GTGT/ ) {
                print("$a matches\n");
        }  else {
                print("$a doesn't match\n");
        }
}

__END__
ACCCCAAAGTGT
ACCCCGGGGTGT
ACCCCAGAGTGT
```

```
ACCCCAAAGTGT matches
ACCCCGGGGTGT matches
ACCCCAGAGTGT matches
```

date.pl

```
#!/usr/bin/perl

print "Enter date (YYYY-MM-DD): ";
$s = <STDIN>;
chomp($s);

if ( $s =~ /(\d\d\d\d)-(\d\d)-(\d\d)/ ) {
    print "Correctly formed date\n";
    print "Year is: $1\n";
    print "Month is: $2\n";
    print "Day is: $3\n";
}
```

```
Correctly formed date
Year is: 2012
Month is: 01
Day is: 23
```

Substitutions

Replace substring that matches the pattern:

```
$string =~ s/PATTERN/REPLACEMENT_STRING/;
```

Case-insensitive pattern matching:

```
$string =~ s/PATTERN/REPLACEMENT_STRING/i;
```

Replace all matches:

```
$string =~ s/PATTERN/REPLACEMENT_STRING/g;
```

Remove all substrings that match:

```
$string =~ s/PATTERN//g;
```

Translating characters

Translates all occurrences of the characters found in the search list with the corresponding character in the replacement list. It returns the number of characters replaced.

```
$string =~ tr/abc/123/;
```

substitution.pl

```
$str1 = "123 45  678   9";
$str2 = "123 45  678   9";
$str3 = "123 45  678   9";
$str4 = "123 45  678   9";
$str5 = "123 45  678   9";

$str1 =~ s/ //;
$str2 =~ tr/ /-/;
$c3 = $str3 =~ s/ //;
$c4 = $str4 =~ s/ //g;
$c5 = $str5 =~ tr/ //d;

print "$str1\n";           # 12345  678   9
print "$str2\n";           # 123-45--678---9
print "$str3 ($c3)\n";     # 12345  678   9 (1)
print "$str4 ($c4)\n";     # 123456789 (6)
print "$str5 ($c5)\n";     # 123456789 (6)
```

array.1

```
@num1 = (3,2,5,9,7,13,16);
@num2 = (3..7);
@num3 = (2..4,9);
@subjects = ("biology","chemistry","math");
@mixed = (3,0.5,"Israel",2.7,"China");
@empty = ();

print "@num1\n";         # 3 2 5 9 7 13 16
print "@num2\n";         # 3 4 5 6 7
print "@num3\n";         # 2 3 4 9
print "@subjects\n";     # biology chemistry math
print "@mixed\n";        # 3 0.5 Israel 2.7 China
print "@empty\n";        #

print "Last index: $#num1\n";         # Last index: 6
print "Length: ", $#num1 + 1, "\n";   # Length: 7
```

array.2

```
@nos = (3,2,5,9);
$sum = 0;
print "Numbers: @nos\n";

foreach $k ( @nos ) {
        $sum += $k;
        print "$k becomes ";
        $k -= 2;
        print "$k\n";
}
print "Sum: $sum\n";
```

```
Numbers: 3 2 5 9
3 becomes 1
2 becomes 0
5 becomes 3
9 becomes 7
Sum: 19
```

array.3

```
@nos = (3,2,5,9,7,13,16);

$first_elem = $nos[0];    # 3
$third_elem = $nos[2];    # 5

@a1 = @nos[2,3,4,5];      # 5 9 7 13
@a2 = @nos[2..5];         # 5 9 7 13
@b  = @nos[0,3..5];       # 3 9 7 13

$nos[5] = 24;
@nos[2..4] = (6,10,8);
print "@nos\n";           # 3 2 6 10 8 24 16

@c = @a1;                  # 5 9 7 13
@d = (0, @c, 4);           # 0 5 9 7 13 4
@d = (1,@d[1,2]);          # 1 5 9
@d = (6,@d,2);             # 6 1 5 9 2
```

array4.pl

```
@countries        = ("Israel","Norway","France","Argentina");
@sorted_countries = sort(@countries);

@numbers          = (1,2,4,8,16,18,32,64);
@sorted_numbers   = sort(@numbers);

print "ORIG:   @countries\n",
      "SORTED: @sorted_countries\n\n",
      "ORIG:   @numbers\n",
      "SORTED: @sorted_numbers\n";
```

```
ORIG:   Israel Norway France Argentina
SORTED: Argentina France Israel Norway

ORIG:   1 2 4 8 16 18 32 64
SORTED: 1 16 18 2 32 4 64 8
```

array5.pl

```
@stack = (1,3,5,7);
push(@stack,9,11,13);

print "@stack\n";

@stack = (1,3,5,7);
$n = shift(@stack);
print "$n\n@stack\n";
```

```
1 3 5 7 9 11 13
1
3 5 7
```

mygrep.pl

```
#!/usr/bin/perl

$pattern = shift(@ARGV);
while ( $_ = <ARGV> ) {
        if ( $_ =~ /$pattern/ ) {
                print $_;
        }
}
```

```
#!/usr/bin/perl

$pattern = shift(@ARGV);
while ( <> ) {
        if ( /$pattern/ ) {
                print;
        }
}
```

text.pl

```
$a = "AAAACCCCGGGGTTACGT";
$b = substr($a, 14, 4);
@c = split(/TT/, $a);
$d = join("TT", @c);
$e = join("TT", "AAAACCCCGGGG", $b);

$f = reverse($b);
$g = join("TT", reverse(@c));


print "$a\n";    # AAAACCCCGGGGTTACGT
print "$b\n";    # ACGT
print "@c\n";    # AAAACCCCGGGG ACGT
print "$d\n";    # AAAACCCCGGGGTTACGT
print "$e\n";    # AAAACCCCGGGGTTACGT
print "$f\n";    # TGCA
print "$g\n";    # ACGTTTAAAACCCCGGGG
```

split.pl

```
$str = "123 45  678   9";

@arr1 = split(/ /,  $str);
@arr2 = split(/  /, $str);
@arr3 = split(/\s*/,$str);
@arr4 = split(/\s+/,$str);
@arr9 = split(//,   $str);

$_ = "123 45  678   9";
@arrD = split;

$a1 = join(",", @arr1);   # 123,45,,678,,,9
$a2 = join(",", @arr2);   # 123 45,678, 9
$a3 = join(",", @arr3);   # 1,2,3,4,5,6,7,8,9
$a4 = join(",", @arr4);   # 123,45,678,9
$a9 = join(",", @arr9);   # 1,2,3, ,4,5, , ,6,7,8, , , ,9
$aD = join(",", @arrD);   # 123,45,678,9
```

hash1.pl

```
%empty = ();
@weights  = (hydrogen,1,carbon,12,oxygen,16);
%weightsa = @weights;
%weights1 = (hydrogen,1,carbon,12,oxygen,16);
%weights2 = (hydrogen=>1, carbon=>12, oxygen=>16);

print "%empty\n";
print %empty, "\n";
print "@weights\n";
print %weightsa, "\n";
print %weights1, "\n";
print %weights2, "\n";
```

```
%empty

hydrogen 1 carbon 12 oxygen 16
carbon12hydrogen1oxygen16
carbon12hydrogen1oxygen16
carbon12hydrogen1oxygen16
```

hash2.pl

```
%weights = (hydrogen=>1, carbon=>12, oxygen=>16);

$weights{sulphur} = 32;
$weights{hydrogen} += 1;
$weights{carbon} = $weights{carbon} + 2;

@weights = %weights;

print "@weights\n";
print "%weights\n";
print %weights, "\n";
print $weights{sulphur}, "\n";
print @weights{oxygen, carbon}, "\n";
```

```
carbon 14 hydrogen 2 sulphur 32 oxygen 16
%weights
carbon14hydrogen2sulphur32oxygen16
32
1614
```

hash3.pl

```
%weights = (hydrogen=>1, carbon=>12, oxygen=>16);

delete $weights{hydrogen};
if ( exists $weights{hydrogen} ) {
    print "Hydrogen’s weight is $weights{hydrogen}\n";
} else {
    print "Hydrogen is not in the list\n";
}

@a = each(%weights); print "@a\n";    # carbon 12
@b = each(%weights); print "@b\n";    # oxygen 16
@c = each(%weights); print "@c\n";    #

%weights = (hydrogen=>1, carbon=>12, oxygen=>16);
while ( ($e,$w) = each(%weights)) {
    print "[$e : $w] ";
}

# [carbon : 12] [hydrogen : 1] [oxygen : 16]
```

count_nucleotides1.pl

```
$sequence="ATGCATACCGACCGT";

while ( $sequence ) {
    $nucleotide = chop($sequence);
    if ( $nucleotide eq "A" ) { $counts{A} += 1; }
    if ( $nucleotide eq "C" ) { $counts{C} += 1; }
    if ( $nucleotide eq "G" ) { $counts{G} += 1; }
    if ( $nucleotide eq "T" ) { $counts{T} += 1; }
}
@counts = %counts;
print "@counts\n";
print %counts, "\n";
```

```
A 4 T 3 C 5 G 3
A4T3C5G3
```

count_nucleotides2.pl

```
$sequence="ATGCATACCGACCGT";
while ( $sequence ) {
    $nucleotide = chop($sequence);
    $counts{$nucleotide} += 1;
}

print "Keys:   ", keys(%counts),   "\n";
print "Values: ", values(%counts), "\n";

foreach $key ( keys(%counts) ) {
    print $key, " has value ", $counts{$key}, "\n";
}
```

```
Keys:   ATCG
Values: 4353
A has value 4
T has value 3
C has value 5
G has value 3
```