9) Substrings and advanced sorting

Matt Webster
IMBIM, BMC
matthew.webster@imbim.uu.se

index

```
$where = index ($big_string, $small_string)
```

returns the position of $small_string within $big_string
beginning of string = 0
not found in string = -1

```
$dna = "ATTGCATCGACGACGTATTACGTATAAGGGTAG";

$where = index ($dna,"GCAT");
```

$where is 3

substr

```
$part = substr ($string, start, length)
```

returns substring of `$string` with given start and length

```
$dna = "ATTGCATCGACGACGTATTACGTATAAGGGTAG";

$codon = substr ($dna,3,3);
$codon is GCA


for ($i=0; $i<length($dna); $i+=3) {
   $codon = substr($dna,$i,3);
}
```

comparing sequences

```
$human_seq="CATGCAGCGCGTACTAGATCGCTAGTCACCAGCACTCGAC";
$chimp_seq="CATGATACGCGTACTAGAGCGCTAGTCAGCAGCACGCGAC";

$diff=0;

for ($i=0; $i<=length($human_seq); $i++) {
   $human_base = substr($human_seq,$i,1);
   $chimp_base = substr($chimp_seq,$i,1);
   $diff++ unless ($human_base eq $chimp_base);
}

print "there are $diff differences\n";
```

more substrings

```
$human_seq="CATGCAGCGCGTACTAGATCGCTAGTCACCAGCACTCGAC";
```

```
substr($human_seq,2)
```
missing the 3rd argument includes the rest of the string

```
substr($human_seq,-10,2)
```
if 2nd argument is negative it counts from the end

```
substr($human_seq,3,3) = "ATG";
```
makes a replacement

```
substr($human_seq,3,3,"ATG");
```
does the same as above

format with `sprintf`

`sprintf` does the same as `printf`
except that the result is stored in a variable

```
my $date_tag = sprintf
"%4d/%02d/%02d %2d:%02d:%02d",
$yr, $mo, $da, $h, $m, $s;
```

advanced sorting

we have used sort to sort lists alphanumerically

```
@students = qw/Doreen Oskar Elin Sangeet Malin/;

@sorted_students = sort @students;
```

advanced sorting

the rules for sorting a list can be broken down into the rule for comparing two values, `$a` and `$b`

you can add a subroutine after the sort command to define how to perform the sort

the subroutine should return 1, -1 or 0:

`-1 = $a before $b`

`1 = $b before $a`

`0 = the same`

advanced sorting

```
@students = qw/Doreen Oskar Elin Sangeet Malin/;

@sorted_students = sort sort_method @students;

sub sort_method {
   if ($a lt $b) {
      return -1;
   } elsif ($a gt $b) {
      return 1;
   } else {
      return 0;
   }
}
```

this will give the default behaviour

three-way string comparison: cmp

```
@students = qw/Doreen Oskar Elin Sangeet Malin/;

@sorted_students = sort sort_method @students;

sub sort_method {
   $a cmp $b;
}
```

this will give the default behaviour

numerical sorting

```
@numbers = (43, 1, -4, 9999, 1323, 1, 1, 1, 42);

@sorted_numbers = sort sort_method @numbers;

sub sort_method {
   if ($a < $b) {
      return -1;
   } elsif ($a > $b) {
      return 1;
   } else {
      return 0;
   }
}
```

using numerical comparisons > and < to sort numerically

three-way number comparison: <=>

the starship operator: <=>

```
@numbers = (43, 1, -4, 9999, 1323, 1, 1, 1, 42);

@sorted_numbers = sort sort_method @numbers;

sub sort_method {
   $a <=> $b;
}
```

or you can simply include it in a single line:

```
@sorted_numbers = sort {$a <=> $b} @numbers;
```

three-way number comparison: <=>

the starship operator: <=>

```
@numbers = (43, 1, -4, 9999, 1323, 1, 1, 1, 42);

@ascending = sort {$a <=> $b} @numbers;



@descending = reverse sort {$a <=> $b} @numbers;
```

or

```
@descending = sort {$b <=> $a} @numbers;
```

sorting hashes

`%word_count`
contains the count of every word in the human genome paper

```
foreach $word (sort keys %word_count) {
   print "$word occurs $word_count{$word} times\n";
}
```
#prints all words in alphabetical order

```
foreach $word (sort by_value keys %word_count) {
   print "$word occurs $word_count{$word} times\n";
}

sub by_value {
   $word_count{$b} <=> $word_count{$a};
}
```
prints all words in descending order of their occurrence

sorting hashes

sorting subroutines can have many levels:

```
@patron_IDs = sort {
    &fines($b) <=> &fines($a) or
    $items{$b} <=> $items{$a} or
    $family_name{$a} cmp $family_name{$b} or
    $personal_name{$a} cmp $family_name{$b} or
    $a <=> $b
} @patron_IDs;
```

summary: strings and sorting

- strings

```
index ($big_string, $small_string)
```

find the first start position of `$small_string` in `$big_string`

```
substr ($string, start, length)
```

take substring of `$string` with start and length

```
sprintf
```

number formatting

- advanced sorting

sort subroutine compares `$a` with `$b`

```
$a <=> $b
```

sort numerically

```
$a cmp $b
```

sort alphanumerically

```
$hash{$a} cmp $hash{$b}
```

sort by hash value