Textverarbeitung mit Perl
Dozentin: Miriam Butt
Universität Konstanz, Sommer 2009
Dokument erstellt von Veronika Walther

Perl – Syntax II

Numeric and string comparison operators

Comparison	*Numeric*	*String*
Equal	==	eq
Not equal	!=	ne
Less than	<	lt
Greater than	>	gt
Less than or equal to	<=	le
Greater than or equal to	>=	ge
Order	$a <=> $b (spaceship)	$a cmp $b

The if Control Structure

- if (*boolean value/Bedingungsausdruck*) {
  if this is true, do what is said in this block
  }

  elsif (*boolean value/Bedingungsausdruck*) {
  but if this is true, do what is said here
  }

  elsif (*boolean value/Bedingungsausdruck*) {
  and if this is true, do what is said in this block
  }

  .
  .
  .

  else {
  if nothing of the above is true, do what is said here
  }

- by adding '!' (*not*) you can tell perl to do something, if a condition is not true:
  if (! *boolean value/Bedingungsausdruck*) {
  if the condition is not (!) true, do what is said in this block
  }

The while Control Structure

- a looping structure/Schleifenstruktur
- while (*truth value/Bedingung*) {
  while there is something in the loop, do what is said here
  }

Arrays

- @array = qw/ fred barney wilma /;
- an array is a list of values
- the first place in an array is 0
- to refer to one value of an array: $array[*place in array*] (so here $array[0] would be fred)
- to refer to the last value in an array: $array[$#array]
- pop
  - takes last element off of an array
  - pop @array; (so the array contains fred and barney only)
- push
  - adds an element/list of elements to the end of an array
  - push @array, $dino; (now the array contains fred, barney and dino)
- shift
  - same as pop, but at the start of an array
  - shift @array; (the array contains barney and dino now)
- unshift
  - same as push, but at the start of an array
  - unshift @array, $fred; (now the array contains fred, barney and dino again)
- the foreach control strucutre
  - foreach $array (@array) {
    do whatever is said here for each element $array of @array
    };
- reverse
  - @reversed = reverse @array; (@reversed contains elements of @array in reversed order)

- sort
  - @sorted = sort @array; (@sorted contains elements of @array in sorted order)

Subroutines

- user-defined functions
- can be used many times in one program
- are global
- can be anywhere in the program
- if there are two subroutines with the same name, the later one overwrites the earlier one

-
  ```
  sub sum_of_fred_and_barney {
  print “Hey, you called the sum_of_fred_and_barney subroutine!\n”;
  $fred + $barney; # that’s the return value
  }
  ```
- you can use the subroutine as follows:
  ```
  $fred = 3;
  $barney = 4;
  $wilma = &sum_of_fred_and_barney; # $wilma gets 7
  print “\$wilma is $wilma.\n”;
  $betty = 3 * &sum_of_fred_and_barney; # $betty gets 21
  print “\$betty is $betty.\n”;
  ```
- the output is:
  ```
  Hey, you called the sum_of_fred_and_barney subroutine!
  $wilma is 7.
  Hey, you called the sum_of_fred_and_barney subroutine!
  $betty is 21.
  ```
- as you can see, the subroutine is reused in this program

The foreach Control Structure

- to process an entire array/list

-
  ```
  foreach $array (@array) {
  $array = “\t$array”; # put a tab in front of each element of @array
  $array .= “\n”; # put a newline on the end of each
  }
  print “The names are: \n”, @array; # each one is indented, on its own line
  ```

(aus Learning Perl/Einführung in Perl, O‘Reilly)