Perl

- Skriptsprache
- Wenig Code für einfache Aufgaben
- Sehr viele String-Funktionen
- Keine Typsicherheit, keine Deklarationen notwendig
- Fast immer mehr als ein Weg, eine Aufgabe zu erledigen

Perl-Skript

```
#!/usr/bin/perl

# die folgende Zeile gibt "Hallo Welt" aus
print "Hallo Welt\n";
```

Skript starten

```
perl versuch.pl

perl -w versuch.pl

./versuch.pl
```

Datentypen

Skalar: $scalar
Array: @array
Hash: %hash
Für Arrays und Hashes keine vorherige Initialisierung mit fester (oder variabler) Größe notwendig
my $var: Geltungsbereich auf aktuellen Block beschränken

Skalare

```
$hallo = "Welt";

$wert = 1;

$doublewert = 6.023e23;

$wert2 = $wert1;
```

Arrays

```
@array = (1, 2, 3, 4);

@array = ("Hallo", " ", "Welt");

@array = (1, "Hallo", 1.5);


$array[0] = "Hallo";

$array[10] = "Welt";


$wert = $array[7]
```

Hashes

```
%hash = ("Hallo" => 1, "Welt" => 2);

%hash = ("Hallo" => "Welt", "Welt" => "weit");


$hash{"Hallo"} = "Welt";

$hash{"Welt"} = 5;


$hash{"Hallo"} = $hash{"Welt"} * 0.43;
```

Besonderheiten

Besondere Variablen:

$0 # *Name des Programms*

@ARGV # *Argumente des Programms*

$_ # *oft aktueller Wert einer (Schleifen–) Anweisung*

Länge eines Arrays:

$length = @array;

Index des letzten Elements eines Arrays:

$last = $#array;

Operatoren

Wie üblich:

+, *, /, -, %

Potenz: **
Konkatenation: .
String-Multiplikation: x
Mit Zuweisung:
+=, *=, /=, -=, .=, x=
Inkrement/Dekrement:
++$a, $a++, --$a, $a--

Operatoren (2)

Logische Operatoren:
&&, ||, !
and, or, not, xor
Vergleich:
==, !=, <, >, <=, >=
$a <=> $b

$a < $b: −1
$a > $b: 1
$a == $b: 0

Bereichsoperator (..)

@array = @array[1 .. 5];

@array = @array[−5 .. −1] # *letzte 5 Elemente*

@array = @array[10 .. $#array] # *10. bis letztes Element*

„magisches“ Autoinkrement:

@alph = (”A” .. ”Z”);

@twosym = (”AA” .. ”ZZ”);

Kontrollstrukturen

```
if (...){
      ...
} elsif (...){
 ...
}else{
 ...
}

unless (...){
 ...
}
```

Kontrollstrukturen (2)

```
while (...){
 ...
}

until (...){
 ...
}

for($i=0;$i<100;$i++){
 ...
}

foreach $value (@array){

}
```

Dateihandles

STDIN, STDOUT, STDERR

open(SESAM, "<file") # *lesen*

open(SESAM, ">file") # *schreiben*

open(SESAM, ">>file") # *anhaengen*

Testen ob Datei existiert:
if (−e /pfad/zur/datei){ ... }

open(SESAM, ">file") or **die** "Could␣not␣open␣file!\n";

Lesen

```
while(<SESAM>){
    print $_;
}

while(<SESAM>){
    print;
}

while(my $line = <SESAM>){
print $line;
}


if( defined( $line = <SESAM> ) ){
    print $line;
}
```

Schreiben

```
open(SESAM,">file"); #oder
open(SESAM,">>file");

print  SESAM "Hallo␣Welt!\n";

print  STDOUT "Hallo␣Welt"\n";
```