

Inhalt:

- Allgemeines
- Grundlagen
- Hello World
- Variablen
- Variablen einlesen
- Rechenzeichen und Vergleichsoperatoren
- If - Anweisung
- Das erste kleine Perl Programm
- Arrays

Allgemeines

Dieses Tutorial soll ein kleiner Einstieg für Perl Anfänger sein. Falls euch PERL auf den ersten Blick gefällt und ihr beschliessen solltet, damit weiterzumachen, lege ich euch das Buch

"Einführung in Perl" von O'Reilly ans Herz. Perl steht für "Practical Extraction and Report Language" und wurde von Larry Wall entwickelt. Im weiteren Verlauf dieses Tuts werde ich annehmen das ihr Windows benutzt.

Grundlagen

Erstmal kommen wir nun dazu wie ihr ein Perl Programm schreibt und wie ihr es ausführt.

Ich werde dabei nun auf 2 verschiedene Möglichkeiten eingehen. Zunächst benötigt ihr natürlich eine Version von ActivePerl.

1. Ihr habt einen Perl - Editor, zum Beispiel DZSoft Perleditor (www.dzsoft.com). Nun braucht ihn nur noch den Editor zu öffnen, euren Perl Code zu schreiben und starten. Dies ist sicherlich eine einfache Möglichkeit.

2. Ihr benutzt keinen Perl Editor.

Ihr legt nun eine Text Datei an, in der ihr euren Perl Code schreibt. Diese Datei speichert ihr mit der Endung : *.pl. Nun geht ihr in die MSDOS-Eingabe Aufforderung. Dort startet ihr euer Programm folgendermassen:

```
C:/euerperlordner/perl.exe dateiname.pl
```

Nun sollte das Programm starten, oder eine Fehlermeldung erscheinen, die euch auf einen Fehler im Programmcode hinweist.

Hello World

Was sollte nun auch anderes kommen als das berühmte "Hello World" :)
Dieser Code lässt euch ein "Hello World" erscheinen:

```
#usr/bin/perl  
  
print "Hello World\n";
```

Erklärung:

#usr/bin/perl ---- > Diese Zeile muss vor jedem Perl Programm stehen.

print "Hello World\n"; --- > Print ist die Ausgabeanweisung in Perl. Was nach print in Anführungszeichen steht wird ausgegeben. Das "\n" steht für "Newline". Also eine Zeile weiter.

Ein Newline wird immer mit in die Print Anweisung genommen. Das Semikolon am Ende der Zeile beendet die Anweisung.

Variablen

Ein Dollarzeichen (\$) vor einem Wort oder Buchstaben steht für eine Variable.

\$variable

Dieser Variablen kann mit einem "=" ein Wert übermittelt werden:

\$variable="Ich bin eine Variable\n";

Soll eine Variable einen Text als Wert erhalten, so muss dieser wie bei der print-Anweisung in Anführungszeichen stehen. Wieder wird die Anweisung durch ein Semikolon beendet.

Variablen einlesen

Das Einlesen von Variablen geschieht in Perl mit der Anweisung : <STDIN>.
Das dürfte dann in euerem Programm ungefähr so aussehen:

\$variable=<STDIN>;chomp(\$variable);

Das "chomp(\$variable)" 'schneidet' das Newline am Ende des eingelesenen Textes ab.

Rechenzeichen und Vergleichsoperatoren

```
$zahl1="1";  
$zahl2="5";
```

\$ergebnis=\$zahl1 + \$zahl2 ---> Dieser Befehl addiert die beiden Zahlen
\$ergebnis=\$zahl1 - \$zahl2 ----> Dieser Befehl subtrahiert die Zahlen
\$ergebnis=\$zahl1 * \$zahl2 -----> Dieser Befehl multipliziert die beiden Zahlen
\$ergebnis=\$zahl1 / \$zahl2 -----> Dieser Befehl dividiert die beiden Zahlen

Bei Zahlen verwendet man folgende Vergleichsoperatoren:

< kleiner
<= kleiner oder gleich
> grösser
=> grösser oder gleich
== entspricht,gleich
!= nicht gleich

Bei Zeichenketten verwendet man folgende Vergleichsoperatoren:

anstatt > muss "gt" verwendet werden (greater than)
anstatt < muss "lt" verwendet werden (less than)
anstatt <= muss "le" verwendet werden (less or equal)
anstatt => muss "ge" verwendet werden (greater or equal)
anstatt == muss "eq" verwendet werden (equal)
anstatt != muss "ne" verwendet werden (not equal)

Die If Anweisung

Die If-Anweisung ist folgendermassen aufgebaut:

```
if (Bedingung) { anweisung }  
else {anweisung}
```

Beispiel

```
#usr/bin/perl  
$zahl="5";  
if ($zahl == "5"){  
print "Die Zahl ist 5\n"; }  
else { print "Die Zahl ist nicht 5\n";}
```

Das erste kleine Perlprogramm

Nun wollen wir ein kleines Perl Programm schreiben.

```
#usr/bin/perl
```

```

print "Bitte geben sie ihren Namen ein\n";
$name=<STDIN>;chomp($name);

if ($name == "Max Mustermann"){
print "Herzlich Willkommen Max\n";}
else {
print "Sie sind nicht Max Mustermann sondern $name\n";}

```

Wenn ihr das Tutorial bis hier her gelesen habt , müsset ihr eigentlich den Programmaufbau verstehen.Deshalb werde ich diese Programm jetzt nicht weiter erklären.

Arrays

Wie ihr bereits wisst, wird eine Variable durch ein Dollarzeichen gekennzeichnet.Arrays werden mit einem @ gekennzeichnet.

```
@array=.....
```

Das Array wird mit runden Klammern eingeschlossen und natürlich durch ein Semikolon beendet.Um die einzelnen Werte eines Array zu trennen , werden Kommas benutzt, die einzelnen Werte werden in Anführungszeichen gesetzt.

```
@wochentage=("Montag", "Dienstag" , "Mittwoch" , "Donnerstag " , "Freitag" ,
"Samstag" , "Sonntag");
```

Wenn man sich diese Anführungszeichen sparen will setzt man vor die Klammern noch ein "qw".

```
@wochentage=qw(Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag,
Sonntag);
```

Was nun wenn wir das Wort "Mittwoch" aus dem Array nehmen wollen?Das Array beginnt mit dem Wert Null.

```
Montag ist Element 0 des Arrays
Dienstag ist Element 1 des Arrays
Mittwoch ist Element 2 des Arrays
```

```
.
.
.
```

Um nun "Mittwoch" auszuwählen, müssen wir also das zweite Element des Arrays auswählen.Wenn man ein Element aus einem Array auswählen möchte, geschieht das folgendermassen:

Anstatt ein @, schreibt man ein \$ vor den Arraynamen.
Dann eine eckige Klammer auf, dann eine Zahl oder ggf eine weitere Variable, dann die Klammer wieder zu und ein Semikolon hintendran. Damzufolge hat:

```
$wochentage[2];
```

den Wert "Mittwoch". Das ganze kann natürlich auch mit Variablen gemacht werden.

```
$i=2;  
$wort=$wochentage[$i];
```

hat ebenfalls den Wert Mittwoch.