Perl-Workshop, Teil III

Ingo Blechschmidt,
Michael Hartmann

4. April 2007

Inhalt

Wiederholung: Zahlen

- Operationen:
  `+`, `-`, `*`, `/`, `**`, `%`, `++`, `--`
- Funktionen:
  `abs`, `int`, `sqrt`

Wiederholung: Strings (Zeichenketten)

- Operationen:
  - `.` (Zusammenhängen):
    ```
    ¨hallo ¨ . ¨welt¨ # ¨hallo welt¨
    ```
  - `x` (Stringmultiplikation):
    ```
    ¨-¨ x 5 # ¨-----¨
    ```
- Interpolation nur in doppelten Anführungszeichen
- Funktionen: `lc`, `lcfirst`, `uc`, `ucfirst`, `length`, `reverse`, `chomp`

Wiederholung: skalare Variablen

- Inhalt: Zahlen und Strings (Zeichenketten)
- Kennzeichnung: `$`-Zeichen
- Deklaration: `my $variable`
- Zuweisung: `my $variable = 42`

Beispiel:

```
1 $variable = 7;
2 $variable++;
3 $variable = sqrt($variable**2 - 15);
4 $variable = ¨00$variable¨;
```

Wiederholung: Eingabe/Ausgabe (IO)

- Ausgabe auf die Kommandozeile: `print`
- Zeile von der Tastatur einlesen: `<STDIN>`
- **inkl.** `chomp`: `chomp(my $line = <STDIN>)`

Welche Werte sind wahr?

Alle Werte sind wahr, außer:

- Zahlenwert `0`
- Leerer String `""`
- String mit Inhalt `0`: `"0"`
- Undefinierter Wert `undef`

Viele Funktionen signalisieren einen Fehler durch Rückgabe eines falschen Werts.

Vergleiche

	Strings	Zahlen
größer	`gt`	`>`
größergleich	`ge`	`>=`
kleiner	`lt`	`<`
kleinergleich	`le`	`<=`
gleich	`eq`	`==`
ungleich	`ne`	`!=`

Verknüpfungen

Präzedenz

Rangfolge von Operatoren (z.B.: $5 + 7 \cdot 2 = 5 + (7 \cdot 2)$)

	niedrigere Präzedenz	höhere Präzedenz
und	`and`	`&&`
oder	`or`	`||`
nicht	`not`	`!`

Beispiel: Was ist wahr?

```
1 my $i = 15;
2 my $s = ¨otto¨;
3
4 ($i >= 20) and ($i <= 30)
5 ($i <  20) and ($i <  30)
6
7 (lc $s eq ¨otto¨) or (lc $s eq ¨hans¨)
8
9 ($i < 20 and $i != 100) or (lc $s ne ¨udo¨)
```

Beispiel

```
 1 #!/usr/bin/perl
 2
 3 use warnings;
 4 use strict;
 5
 6 chomp(my $alter = <STDIN>);
 7
 8 if($alter < 20) {
 9   print "Du bist juenger als zwanzig.\n";
10 } elsif($alter >= 20 and $alter < 30) {
11   print "Du bist in den zwanzigern.\n";
12 } elsif($alter >= 30 and $alter < 40) {
13   print "Du bist in den dreissigern.\n";
14 } else {
15   print "Du bist mindestens 40 Jahre alt.\n";
16 }
```

Struktur

```
1 if(BEDINGUNG1) {
2    # entweder fuehre nur diesen Block aus...
3 } elsif(BEDINGUNG2) {
4    # ...oder nur diesen...
5 } elsif(BEDINGUNG3) {
6    # ...oder nur diesen...
7 } else {
8    # ...ansonsten fuehre diesen Block aus
9 }
```

Struktur

- der Code in einem `if`- oder `elsif`-Block wird ausgeführt, wenn die Bedingung wahr ist
- es kann **höchstens** ein Block ausgeführt werden
- einem `if` können beliebig viele `elsif` folgen
- abschließend kann ein `else` angegeben werden

Übung

Der Benutzer soll zwei Zahlen auf der Kommandozeile eingeben und das Programm soll entscheiden, ob die erste oder zweite Zahl größer ist oder beide gleich groß sind.

Übung

Der Benutzer soll zwei Zahlen auf der Kommandozeile eingeben und das Programm soll entscheiden, ob die erste oder zweite Zahl größer ist oder beide gleich groß sind.

```
 1 chomp(my $a = <STDIN>);
 2 chomp(my $b = <STDIN>);
 3
 4 if($a > $b) {
 5   print "Die erste Zahl ist groesser.\n";
 6 } elsif($a < $b) {
 7   print "Die zweite Zahl ist groesser.\n";
 8 } else {
 9   print "Beide Zahlen sind gleich gross.\n";
10 }
```

Übung

Der Benutzer soll eine Zahl auf der Kommandozeile eingeben und das Programm soll prüfen, ob die Zahl zwischen 0 und 100 liegt.

Übung

Der Benutzer soll eine Zahl auf der Kommandozeile eingeben und das Programm soll prüfen, ob die Zahl zwischen 0 und 100 liegt.

```
1 chomp(my $i = <STDIN>);
2
3 if($i > 0 and $i < 100) {
4   print "$i liegt zwischen 0 und 100.\n"
5 } else {
6   print "$i liegt nicht zwischen 0 und 100.\n";
7 }
```

while-Schleife

```
1 while(BEDINGUNG) {
2   # fuehre Block aus
3 }
```

- der Code im Block wird ausgeführt, solange die Bedingung wahr ist
- `next`: aktuellen Durchgang der Schleife verlassen
- `last`: Schleife verlassen

Beispiel

```
 1 #!/usr/bin/perl
 2
 3 use warnings;
 4 use strict;
 5
 6 my $i = 15;
 7 while($i > 0) {
 8   print "$i\n";
 9   $i--;
10 }
```

Übung

Ein Programm soll die nächsten fünf Zahlen nach der eingegebenen Zahl ausgeben.

Mögliche Lösung

```
 1 #!/usr/bin/perl
 2
 3 use warnings;
 4 use strict;
 5
 6 chomp(my $zahl = <STDIN>);
 7
 8 my $ende = $zahl + 5;
 9 $zahl    = $zahl + 1;
10
11 while($zahl <= $ende) {
12   print $zahl++ . "\n";
13 }
```

bis EOF von der Kommandozeile lesen

```
1 #!/usr/bin/perl
2
3 use warnings;
4 use strict;
5
6 while(defined(my $zeile = <STDIN>)) {
7   ... # $zeile verarbeiten
8 }
```

- `defined`: gibt einen wahren Wert zurück, außer Inhalt ist nicht definiert (`undef`)
- Schleife liest solange von der Kommandozeile, bis es ein EOF (Ende der Datei) liest

bis EOF von der Kommandozeile lesen

```
1 #!/usr/bin/perl
2
3 use warnings;
4 use strict;
5
6 while(my $zeile = <STDIN>) {
7   ... # Zeile enthaelt immer ¨\n¨ => immer wahr
8 }
```

- `defined`: gibt einen wahren Wert zurück, außer Inhalt ist nicht definiert (`undef`)
- Schleife liest solange von der Kommandozeile, bis es ein EOF (Ende der Datei) liest

bis EOF von der Kommandozeile lesen

```
1 #!/usr/bin/perl
2
3 use warnings;
4 use strict;
5
6 while(<STDIN>) {
7   ... # Zeile ist nun in der Variable $_
8 }
```

- `defined`: gibt einen wahren Wert zurück, außer Inhalt ist nicht definiert (`undef`)
- Schleife liest solange von der Kommandozeile, bis es ein EOF (Ende der Datei) liest

bis EOF von der Kommandozeile lesen

```
1 #!/usr/bin/perl
2
3 use warnings;
4 use strict;
5
6 while(<>) {
7   ... # Zeile immer noch in der Variable $_
8 }
```

- `defined`: gibt einen wahren Wert zurück, außer Inhalt ist nicht definiert (`undef`)
- Schleife liest solange von der Kommandozeile, bis es ein EOF (Ende der Datei) liest

bis EOF von der Kommandozeile lesen

```
1 #!/usr/bin/perl
2
3 use warnings;
4 use strict;
5
6 while(chomp(my $zeile = <STDIN>)) {
7   ... # Zeile in $zeile, aber
8       # ohne Zeilenumbruch
9 }
```

- `defined`: gibt einen wahren Wert zurück, außer Inhalt ist nicht definiert (`undef`)
- Schleife liest solange von der Kommandozeile, bis es ein EOF (Ende der Datei) liest

Übung

Das Programm soll den Text, den es einliest, mit Zeilennummern wieder ausgeben.

Übung

Das Programm soll den Text, den es einliest, mit Zeilennummern wieder ausgeben.

```
 1 #!/usr/bin/perl
 2
 3 use warnings;
 4 use strict;
 5
 6 my $nr = 1;
 7 while(defined(my $zeile = <STDIN>)) {
 8   # Zeilenumbruch nicht durch chomp entfernen
 9   print "$nr $zeile";
10
11   $nr++;
12 }
```

Übung

Ein Programm soll 10 Zahlen einlesen und die größte Zahl wieder ausgeben

Übung

```
 1 #!/usr/bin/perl
 2
 3 use warnings;
 4 use strict;
 5
 6 my $i   = 1;
 7 my $max = 0;
 8
 9 while($i++ <= 10) {
10   chomp(my $zahl = <STDIN>);
11
12   $max = $zahl if($zahl > $max);
13 }
14
15 print "groesste Zahl ist: $max\n";
```

Hausaufgaben

- Palindrom-Filter im UNIX-Stil:
  Das Skript soll Wörter von der Kommandozeile einlesen, Palindrome erkennen und wieder ausgeben.
- Wertetabelle für $f(x) = \frac{1}{x}$:
  Das Skript soll die Funktionswerte für die Umkehrfunktion ($\frac{1}{x}$) für -5,-4,. . . ,5 berechnen und ausgeben (Achtung: Division durch Null vermeiden!):
  -5: -0,2
  -4: -0,25
  . . .

Hausaufgaben

- Gerade Zahlen zwischen 0 und 100:
  Das Skript soll alle geraden zahlen zwischen 0 und 100 (2, 4, 6, . . .) ausgeben.

Bildnachweis

- http://gnosislivre.org/twiki/pub/PerlMongersSSA/WebHome/camel.gif
- http://imagecache2.allposters.com/images/pic/153/996106~Jeans-String-Posters.jpg
- http://www.laborsys.com/Images/BlackKeyboard.JPG