

Perl-Workshop, Teil I

Ingo Blechschmidt,
Michael Hartmann

3. Januar 2007



Inhalt

1 Einführung

- Was ist Perl?
- Merkmale
- Hilfe bekommen
- Tugenden eines Perl-Programmierers

2 Erstes Programm

- Kommentare
- Pragmas
- Ausführung

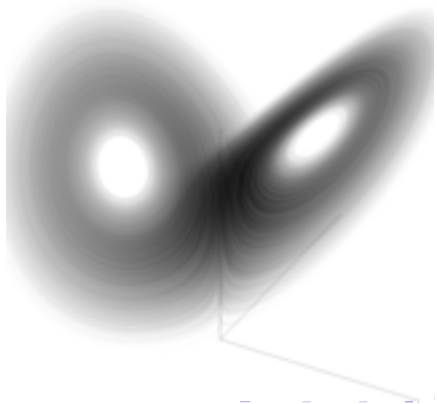
3 Skalare Variablen

4 Operationen und Zuweisungen

- Zuweisungen
- Zahloperationen
- Stringoperationen

Was ist Perl?

- Perl: Programmiersprache
- perl: Compiler/Interpreter/Laufzeitumgebung
- dynamisch
- frei
- flexibel
- modular
- magisch
- „schnell genug“
- cool :)



Merkmale

- “There’s more than one way to do it” (TIMTOWTDI)
- “makes easy jobs easy and hard jobs possible”
- Kontextsensitivität
- Comprehensive Perl Archive Network (CPAN)



Hilfe bekommen

- Befehle: `$ perldoc -f Befehl`
- Module: `$ perldoc Modulnamen`
- IRC: `irc.perl.org, #perl`
- Lugalist :)

Tugenden eines Perl-Programmierers

- Faulheit
- Ungeduld
- Anmaßung

Hallo-Welt-Programm

```
1 #!/usr/bin/perl
2
3 use warnings;      # Warnungen
4 use strict;        # sauberes Programmieren
5 use diagnostics;   # ausführliche Warnungen
6
7 # Jetzt wollen wir auch etwas ausgeben
8 print "Hallo Welt\n";
```


Erstes Program – Kommentare

- Kommentareeröffnung durch Raute (#)
- Ignorieren von allem zwischen zwischen (einschließlich) # und dem Zeilenende
- Shebang: Kennzeichen für den Kernel, die Datei an perl zu übergeben

Erstes Program – Pragmas

- Veränderung bestimmter Eigenschaften der Sprache
- *warnings*: detaillierte Warnungen
- *strict*: Erschwerung bestimmter Arten unsauberer Programmierung
- *diagnostics*: detaillierte Hilfetexte zu Warnungen

Erstes Program – Ausführung

- `$ perl programm.pl`
- `$ chmod +x programm.pl`
`$./programm.pl`

Skalare Variablen

- Kennzeichnung skalarer Variablen durch `$`-Zeichen
- Name muss auf `/[a-z_]\w*/` passen ;)
- Analog zu Dateien:
 - Variablenname $\hat{=}$ Dateiname
 - Variablenwert $\hat{=}$ Dateiinhalt
 - Möglichkeit der Ersetzung des Inhalt durch anderen Inhalt

Skalare Variablen (Forts.)

- Inhalt: Zeichenketten (Strings) oder Zahlen
- Automatische Interpretation des Variableninhalts als Zahl oder String wenn nötig (umgangssprachlich: „Variable wird interpretiert“)
- Zwang zur Deklaration durch `my` bei der ersten Verwendung

Beispiel

```
1 #!/usr/bin/perl
2
3 use warnings;
4 use strict;
5 use diagnostics;
6
7 my $name      = 'Han Solo';
8 my $medium    = "All";
9 my $alter     = 34;
10 my $c         = 2.99792458e8;
11
12 print "$name ($alter) fliegt mit...\n";
13 print "...$c m/s durch das $medium.\n";
```

Zuweisungen

Kontextabhängige Bedeutung des Variablennamens

```
$foo = $foo + 17;
```

- `$foo`: Interpretation als Variable
- `$foo`: Interpretation als Wert

Vergleiche mit Shellprogrammierung:

```
PATH="$PATH:/usr/local/sbin"
```

Zahloperationen

Addition:	+	<code>my \$a = 5+2</code>
Subtraktion:	-	<code>my \$a = 5-2</code>
Multiplikation:	*	<code>my \$a = 5*2</code>
Division:	/	<code>my \$a = 5/2</code>
Potenzieren:	**	<code>my \$a = 5**2</code>

Modulo („Rest“):	%	<code>my \$a = 5%2</code>
------------------	---	---------------------------

Inkrementieren:	++	<code>my \$a; ++\$a; \$a++</code>
Dekrementieren:	--	<code>my \$a; --\$a; \$a--</code>

Beispiel

```
1 #!/usr/bin/perl
2
3 use warnings;
4 use strict;
5 use diagnostics;
6
7 my $zinssatz = 0.05;
8 my $darlehen = 100;
9 my $jahre    = 5;
10
11 my $schulden =
12     $darlehen * ($zinssatz+1)**$jahre;
13
14 print "Ihre Schulden nach $jahre Jahren:\n";
15 print "$schulden\n";
```

Stringoperationen

Zusammenhängen: .

Stringmultiplikation: x

```
my $a = "Peter" . "Huth"
```

```
my $a = "wah" x 10
```

Fragen

- Was ist die Shebang?
- Wie kommentiert man Programmcode?
- Was sind Pragmas und wie nutzt man sie?
- Was sind skalare Variablen?

Bildnachweis

- <http://www.cyberarmy.net/files/pub/dfe70346e198cf46e84dd790e62e3825/perl.jpg>
- <http://gnosislivre.org/twiki/pub/PerlMongersSSA/WebHome/camel.gif>
- http://upload.wikimedia.org/wikipedia/commons/0/0e/Lorenz_attractor.png
- <http://cpan.org/misc/jpg/cpan.jpg>