Perl++

Hagen Paul Pfeifer

https://www.jauu.net ● hagen@jauu.net

30. Mai 2005

Überblick

- Nützliches und Nutzbares
- Referenzen
- Objekte
- tie()
- eval()
- POD
- Benchmarking && Debugging
- Reguläre Ausdrücke vom Nahen

Nützliches und Nutzbares

- use strict;

```
use strict;
my $var = "foo";
$varr = "bar";   # fehlende Deklaration!
                 # (Global symbol $varr requires explicit ...
```

- perldoc -m Data::Dumper

Referenzen

- benannte Referenzen mit den Backslashoperator erzeugen: `$a = \$scalar`
- Derefernzierung: `${$a}` oder `$$a` (nicht immer moeglich)
- Bei Arrays: `$b = \@array;  scalar @{$b};` (letzter Index: `$#$b`
- Funktionen:
  - Referenzieren: `$f = \&funktion;`
  - Dereferenzierung: `&{$f}($a, $b);`
- . . . (Hashes, Foilehandles)

Anonyme Strukturen

- Anonyme Liste
  - `$a = ["Unix", "Friends", "And", "..."]`
  - Dereferenzierung via Pfeiloperator: `$a->[0]`
- Anonyme Hashes
  - `$b = {a => 0, b => 1, c => 2 }"`
  - Dereferenzierung: `$a->{'a'}`
- Anonyme Subroutinen
  - `$c = sub {print "xyz";}"`
  - Dereferenzierung: `$c->("arg0")`
  - oder `&{$c}("arg0")`

Komplexe Datenstrukturen

- Arrays von Arrays:
  - `$a = [’a’, ’b’, ["ca", "cb", ["cca"], "cd"], "d"]; print $a->[2][2][0]`
- Listen von Hashes:
  - `$b = [ {’a’ => 0, ’b’ => 1}, {’x’ => 0, ’y’ => 1} ]; print $b->[0]{’a’`
- Hashes von Listen:
  - `$c = { ’a’ => [0, 1, 2], ’b’ => [3, 4, 5] }; print $a->{’a’}[0];`
- Hashes von Hashes:
  - `$d = { ’a’ => {"aa" => "00", "ab" => "01"}, ’b’ => {"ba" => "10", "bb"`
- Hilfreich DATA::Dumper

Debugging

- integrieter Debugger: `perl -d`
- Kommandos (das nötigste):

Taste	Kommando	Bedeutung
l	list	zeigt aktuelle Zeile an
L	list breakpoints	zeigt alle Zeilen mit Breakpoints
S	list subs	zeigt alle Subroutinen an
b	breakpoint n	setzt Breakpoints bei Zeile n
s	single step	einfacher Schritt (betritt Funktionen)
n	next step	einfacher Schritt (betritt Funktionen NICHT)
c	continue	fortsetzen bis zum nächsten Breakpoint
p	print n	zeigt Wert von n
x	print n	zeigt Wert von n (pretty print)

- . . . der Rest praktisch am Codebeispielen!