

Subroutinen

# Subroutinen

...oder Funktionen oder Prozeduren oder ...

# Subroutinen

Definition:

```
sub routine{  
  ... # auszufuehrender Code  
}
```

Aufruf:

```
routine ();
```

# Argumente

- ▶ alle Argumente in Array `@_` gespeichert
- ▶ keine Typsicherheit (Perl!)
- ▶ keine feste Anzahl
- ▶ Ausweg 1: (teilweise) Prototypen (aber nicht in dieser Vorlesung)
- ▶ Ausweg 2: Gute und vollständige Dokumentation der erwarteten Parameter!

Zugriff auf Elemente von `@_` wie bei jedem anderen Array (`$_[0]`).

## Tricks mit Argumenten

```
sub setHash{  
    my ($key, $value) = @_  
    $hash{$key} = $value;  
}
```

```
sub peek{  
    my $next = shift # oder shift(@_)  
    return $next;  
}
```

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

```
sub createHash{  
    my %hash = @_  
    return %hash;  
}
```

```
%here = createHash("a" => 1, "b" => 2);
```

Referenzen

Referenzen

# Referenzen

Symbolische Referenzen:

```
$name = "count";  
$$name = 1; # Zugriff auf $count  
$name = "count2";  
@$name = ( 2, 5, 7 ); # Array @count2  
$count3 = $$name[2]; # Zugriff auf $count2[2]
```

Harte Referenzen:

```
$name = \ $count;  
$$name = 1; # Zugriff auf $count  
$name = \ @count2;  
@$name = ( 2, 5, 7 ); # Array @count2  
$count3 = $$name[2]; # Zugriff auf $count2[2]
```

Uns interessieren in erster Linie harte Referenzen.

# Harte Referenzen

Erzeugen von Referenzen:

`$scalarref = \ $scalar;`

`$arrayref = \@array;`

`$hashref = \%hash;`

Zugriff:

`$$scalarref = 5;`

`@$arrayref = ( 2, 5, 7 );`

`$$hashref{"bla"} = "blub";`

`$$arrayref [2] = 3; # entspricht`

`#{ $arrayref } [2] = 3; # und nicht`

`#{ $arrayref [2] } = 3;`

# Pfeil-Operator

Automatische Dereferenzierung vor Zugriff:

```
$$arrayref [2] = 3 # entspricht
```

```
$arrayref ->[2] = 3;
```

```
$$hashref{"bla"} = "blub"; #entspricht
```

```
$hashref->{"bla"} = "blub";
```

*Benötigen wir später für Zugriff auf Methoden von Objektreferenzen*

# Anonymität

Anonyme Arrays:

```
$arrayref = [ 2, 5, 7];  
$arrayref->[2] = 3;
```

Anonyme Hashes:

```
$hashref = { "bla" => "blub", "Hallo" => "Welt"};  
$hashref->{"bla"} = 3;
```

Zugriff nur noch über Referenz.

*Objekte in Perl sind oft Referenzen auf anonyme Hashes.*

Pakete

Pakete

# Pakete

- ▶ Pakete definieren Namesräume für Variablen
- ▶ Mehrere Pakete pro Datei
- ▶ Mehrere Dateien für ein Paket
- ▶ Ein Paket pro Datei und eine Datei für jedes Paket (für uns am relevantesten)  
⇒ Module
- ▶ Durch Pakete Wiederverwenden von Code möglich
  - ▶ Klare Trennung zwischen lokal definierten Variablen und denen des Paketes
  - ▶ Variablen des Paketes können nicht aus Versehen verändert werden

# Deklaration

Deklaration des Beginns eines Paketes mit

**package** paketname;

Beispiele:

**package** Color;

**package** Color::Red;

- ▶ Deklaration an beliebiger Stelle im Code
- ▶ Geltungsbereich bis
  - ▶ zur nächsten Paket-Deklaration
  - ▶ Ende des einschließenden Blocks
  - ▶ Ende der Datei

# main-Paket

Im Paket main liegen alle Variablen, außerhalb anderer Pakete und

- ▶ \$\_
- ▶ STDIN
- ▶ STDOUT
- ▶ STDERR
- ▶ ARGV
- ▶ ...

Das main-Paket enthält alle anderen Pakete und sich selbst:

```
main::Color :: Red # entspricht  
Color :: Red
```

```
main::main::main # entspricht  
main::main # entspricht  
main
```

# Zugriff

Eine Variable `$var`, die im Paket `Color :: Red` deklariert wurde, erreicht man (in anderen Paketen) mit

```
$Color :: Red::var
```

Für Variablen im `main`-Paket gilt

```
$main::var # entspricht
```

```
$::var # entspricht
```

```
$var
```

Module

Module

# Module

Module sind Pakete, die

- ▶ aus genau einer Datei bestehen
- ▶ deren Datei auf `.pm` (für Perl-Modul) endet
- ▶ deren Datei genauso heißt, wie das Paket
- ▶ `::` entspricht Pfad-Trennzeichen

Beispiel:

Datei `./Color/Red.pm`

```
package Color::Red;
```

```
our $VERSION = 1.0; # Versionsnummer
```

...

## Module (2)

Benutzung von Modulen:

**use** Color :: Red; *# ohne .pm*

- ▶ Das Modul Color :: Red muss in einer Datei Color/Red.pm definiert sein,
- ▶ das Verzeichnis Color muss im aktuellen Pfad (@INC) von Perl liegen
- ▶ *Der Pfad kann über die Umgebungsvariable PERL5LIB geändert werden.*
- ▶ das aktuelle Verzeichnis (.) ist (normalerweise) in @INC enthalten