Subroutinen:

- bestehen aus Subroutinenkopf und -rumpf
- Subroutinenkopf: Subroutinenname, übergebene Parameter in Klammern
  z.B. sub addiere($$)
- Subroutinenrumpf: Parametersektion (Variablendeklaration und Parameterübergabe), Anweisungen
  z.B. {
  my $zahl1 = @_[0];
  my $zahl2 = @_[1];
  my $erg = $zahl1 + $zahl2;
  return ($erg);
  }
- Aufruf einer Subroutine: &name(argumente)
  z.B. $erg = &addiere($zahl1, $zahl2);

Möglichkeiten des Datenaustauschs:

- über Argumente: &drucke($wert)
- über Rückgabewert in Subroutine: return $ergebnis
- über globale Variablen: our $zahl

Zugriff auf Argumentvariablen:

- Argumentvariablen werden in @_ (Stack Listenvariable) gespeichert
- Zugriff bei Skalaren: über Index → 1. Element steht an @_[0], 2. an @_[1], …
- Zugriff bei Listen/ Hashes: gesamtes @_
- Gemischter Zugriff:
  - geht mit nur wenn Skalar als erstes Argument übergeben wird, die Länge der Liste/ des Hashes unbekannt ist
  - oder call by reference

Lokale vs. globale Variablen:

- lokale Variablen: werden mit **my** deklariert und gelten nur innerhalb des Blocks in dem sie deklariert werden
- globale Variablen: werden mir **our** deklariert und gelten im gesamten Programm

Call by Value:

- der Wert der Variable wird kopiert
- Subroutine kann mit diesem arbeiten, aber nicht die ursprüngliche Variable ändern

Call by Reference:

- Referenz auf Variable wird übergeben
- Subroutine arbeitet mit der eigentlichen Variable!!!!
- wird beim Übergeben an die Subroutine mit \ gekennzeichnet z.B. &aendere (\@namen)
- Referenz wird gespeichert: my ($liste_referenz) = @_[0];
- Zugriff auf ganze Liste: @{$liste_referenz}
- Zugriff auf erstes Element: ${$liste_referenz}[0]

Beispiel:

```
{
  my $zahl = 1;
  my $erg = &addiereEins($zahl);
  print "Ergebnis: $erg\n";
  print "Zahl: $zahl\n";
}

sub addiereEins($) {
  my $zahl = @_[0];
  $zahl++;
  return $zahl;
}
```

→ Call by Value: $zahl ist nach wie vor 1

```
use strict;
{
  my $zahl = 1;
  my $erg = &addiereEins(\$zahl);
  print "Ergebnis: $erg\n";
  print "Zahl: $zahl\n";
}

sub addiereEins($) {
  my $zahl = @_[0];
  ${$zahl}++;
  return ${$zahl};
}
```

→ Call by Reference: $zahl ist jetzt auch 2