Was kontra Wie

Softwareentwicklung ist ein schrittweiser Prozess von Was nach Wie:

- Was: Spezifikation
- Wie: Implementierung

Phasen:

- Anforderungsanalyse
- Entwurf (Design)
- Implementierung

Deklarative Programmierung

Ziel der deklarativen Programmierung: ausführbare Spezifikation

Logische Programmierung:

- Spezifikation durch Predikatenkalkül erster Ordnung
- Lösung wird abgeleitet (*infer*)
- Ausführung: Inferenzmaschine (*inference machine*)

Einfaches Beispiel

is_in(x, L) ist wahr wenn x ein Element der Liste L ist

```
for all elements x and lists L: is_in(x, L) iff
    L = [x]
    or
    L = L1 • L2 and
        (is_in(x, L1) or is_in(x, L2))
```


Beispiel

for all integer lists X, Y: sort(X, Y) iff
    permutation(X, Y) and sorted(Y)

sorted(Y) iff
    forall j such that $1 \leq j <$ length(Y)
        $Y_j \leq Y_{j+1}$

permutation(X, Y) iff
    is_empty(X) and is_empty(Y)
    or else
    has_head(Y, H) and has_tail(Y, Ys) and
        delete(H, X, Xs) and permutation(Xs, Ys)

Prolog

Terme:

- Konstante: alpha
- Konstante: 125
- Variable: X
- zusammengesetzter (*ground*) Term: abs(-10, 10)
- zusammengesetzter Term: abs(suc(X), 5))
- leere Liste: []
- Liste: [toot, duck, donald]
- Listenzelle: [head | tail]

Fakten und Regeln

Fakten:

```
it_rains
less_than(5, 99)
```

Regeln:

```
conclusion :- condition
```

```
length([], 0).
length([X|Y], N) :-
    length(Y, M),
    N is M + 1.
```


Abfragen (*queries*)

```
?- sort([3, 2, 7, 1], [1, 2, 3, 7]).
?- sorted([2, 3, 7]).
?- sort([3, 2, 7], X).
```


Arithmetik

```
fact(0, 1).
fact(N, F) :-
   N > 0,
   N1 is N - 1,
   fact(N1, F1),
   F is N * F1.
```

Relationen

Datenabfragesprachen (SQL)

?- composer(mozart, 1756, 2001).

?- composer(mozart, (X, Y).

?- composer(X, Y, 1901).

?- composer(X, Y, Z).