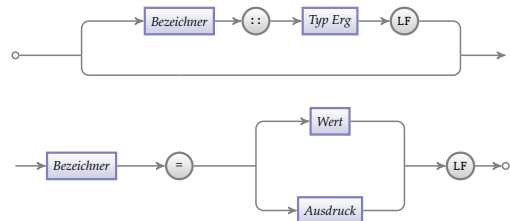


## 1 Kommentare

- Zeilenkommentar: `-- Text`
- Blockkommentar: `{- Text -}`

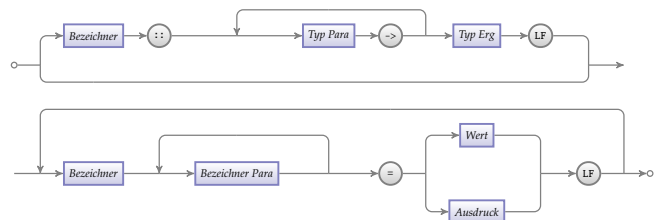
## 2 Konstanten

```
pi :: Double
pi = 3.14
```



## 3 Funktionen

```
plus :: Int -> Int -> Int
plus a b = a + b
```



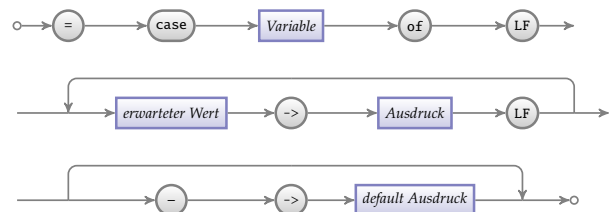
## 4 If-Then-Else

```
ggT a b = if b == 0
  then a
  else ggT b (mod a b)
```



## 5 Case Of

```
ggT a b = case b of
  0 -> a
  _ -> ggT b (mod a b)
```



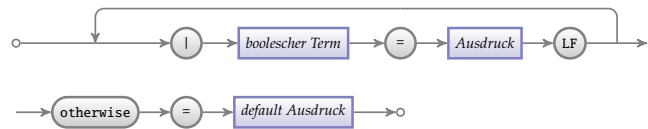
## 6 Pattern Matching

```
ggT a 0 = a
ggT a b = ggT b (mod a b)
```



## 7 Guards

```
ggT a b | b == 0    = a
        | otherwise =
          ggT b (mod a b)
```



## 8 Module

```
module Wurf(weite, square) where
weite :: Double -> Double -> Double
weite v0 phi = ((square v0) / 9.81) * sin (2 * phi)
square :: Double -> Double
square x = x * x

module Foo where
import Wurf hiding (weite)
bar ... = ... (square a) ...
```

## 9 Eigene Datentypen

```
data Point = Point{x :: Double, y :: Double}

data Shape = Circle{center :: Point,
  radius :: Double}
  | Rectangle{point :: Point, width :: Double,
  height :: Double}
  | Triangle{point1 :: Point, point2 :: Point,
  point3 :: Point}
```

## 10 Typparameter

Keine Typparameter in Datentyp-Definitionen!

```
data (Eq a, Ord a) => Pair a b =
  PairConst {first :: a, second :: b} deriving(Show)
```

Besser:

```
data Pair a b = PairConst a b deriving(Show)

instance Eq a => Eq (Pair a b) where
  (PairConst i1 _) == (PairConst i2 _) = i1 == i2

instance Ord a => Ord (Pair a b) where
  (PairConst i1 _) <= (PairConst i2 _) = i1 <= i2

bubbleSort :: (Ord a) => [(Pair a b)] -> [(Pair a b)]
bubbleSort [] = []
bubbleSort (x:xs) = step $ foldl go (x,[]) xs where
  go (y,acc) x = (min x y, max x y : acc)
  step (x,acc) = x : bubbleSort acc
```

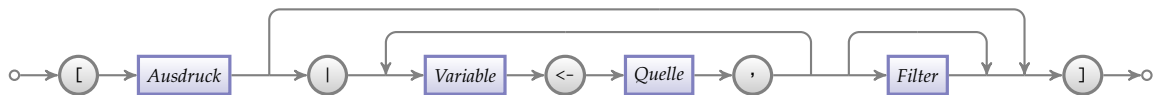
\$ ist nichts anderes als:

```
(7 + 6) / (5 + 3) = (7 + 6) / $ 5 + 3
```

Die „Klammerung“ geht bis zum Zeilenende.

## 11 Listen

```
primes = sieve [2..]
  where
    sieves (p:xs) = p:sieves [x|x<- xs, mod x p > 0]
```



## 12 Operationen

Bezeichner	Typ	Bedeutung
(+)	$a \rightarrow a \rightarrow a$	Addition
(-)	$a \rightarrow a \rightarrow a$	Subtraktion
(*)	$a \rightarrow a \rightarrow a$	Multiplikation
<b>negate</b>	$a \rightarrow a$	Negation
<b>abs</b>	$a \rightarrow a$	Absolutbetrag
<b>signum</b>	$a \rightarrow a$	Vorzeichenbildung

Tabelle 1: Für die Typen **Int**, **Integer**, **Float** und **Double**

Bezeichner	Typ	Bedeutung
<b>succ</b>	$a \rightarrow a$	Nachfolgerbildung
<b>pred</b>	$a \rightarrow a$	Vorgängerbildung
<b>div</b>	$a \rightarrow a \rightarrow a$	ganzzahlige Division, Ergebnis wird abgerundet
<b>mod</b>	$a \rightarrow a \rightarrow a$	zur ganzzahligen Division <b>div</b> gehörender Rest
<b>quot</b>	$a \rightarrow a \rightarrow a$	ganzzahlige Division, Ergebnis wird Richtung 0 gerundet
<b>rem</b>	$a \rightarrow a \rightarrow a$	zur ganzzahligen Division <b>quot</b> gehörender Rest

Tabelle 2: Für die Typen **Int** und **Integer**

Bezeichner	Typ	Bedeutung
(/)	$a \rightarrow a \rightarrow a$	Division
<b>recip</b>	$a \rightarrow a$	Kehrwertbildung
(**)	$a \rightarrow a \rightarrow a$	Potenzieren
<b>sqrt</b>	$a \rightarrow a$	Ziehen der Quadratwurzel

Tabelle 3: Für die Typen **Float** und **Double**

Bezeichner	Typ	Bedeutung
(  )	$a \rightarrow a \rightarrow a$	Oder
(&&)	$a \rightarrow a \rightarrow a$	Und
<b>not</b>	$a \rightarrow a$	Verneinung

Tabelle 4: Für den Typ **Bool**