Bezeichnungen in Haskell

Wie in den meisten Programmiersprachen gibt es in Haskell Bezeichnungen, die nicht als Funktionsnamen oder Funktionsargumente verwendet werden dürfen, da sie reserviert sind. Dazu zählen z. B. folgende Schlüsselwörter:

```
if      then      else      case
of      where     let
```

Haskell unterscheidet zwischen Groß- und Kleinschreibung. Bezeichnungen werden in Haskell immer klein geschrieben. Zusätzlich wird auf Leerzeichen, Sonderzeichen und Umlaute verzichtet.

Datentypen in Haskell

Haskell hat ein **statisches Typsystem**. Der Datentyp der Funktion wird statisch während der Übersetzungszeit des Programms abgeleitet.

Das hat durchaus Vorteile:

- Datentypfehler werden früher erkannt
- Durch die Reduzierung der Typ-Überprüfung reduziert sich auch die Ausführungszeit

Haskell benötigt in jedem Programm **Datentypen** für die lokale Funktionsdefinition.

`Int`	ganzzahlige Werte	$(-2^{31} \dots 2^{31} - 1)$
`Integer`	ganzzahlige Werte	`(unbeschränkt)`
`Bool`	Wahrheitswerte	`True, False`
`Char`	Zeichen	`'a'  '1'  '+'`
`Float`	Gleitkommazahlen	`(32 Bits)`
`Double`	Gleitkommazahlen	`(64 Bits)`
`Typ1 -> Typ2`	Funktionen	

Kommentare in Haskell

Kommentare werden wie folgt verwendet:

```
-- Dies ist ein Zeilenkommentar


{- Dies ist ein
     mehrzeiliger
      Blockkommentar -}
```

Gewöhnen Sie sich das Kommentieren Ihres Codes unbedingt an, da es in sämtlichen Prüfungen Punkte dafür geben wird.

Typ- und Funktionsdefinitionen

Ein Haskell-Programm besteht aus einem oder mehreren Modulen. Ein Modul besteht aus Funktionsdefinitionen und Typ-Deklarationen.

„Argument von Typ Int nach Typ Int“

```
haelfte :: Int -> Int              -- Typ-Deklaration
haelfte x = div x 2                -- Funktionsdefinition
```

„Argumente von den Typen Int und Int nach Typ Int“

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

```
doppel :: Int -> Int
doppel a = 2*a
```

```
pi :: double
pi = 3.141598                      -- Konstante Definition
```