Aufbau einer Funktion

Funktionen werden – wie in der Mathematik – durch eine Gleichung dargestellt. Der linke Teil (Name der Funktion, gefolgt von Parametern, die durch Leerzeichen getrennt werden) wird durch den Teil rechts vom Gleichheitszeichen definiert.
Um den Typ der Parameter und des Rückgabewertes explizit festzulegen, wird der Funktionsdefinition eine Typdefinition vorangestellt.

Beispiel:

```
-- Quadrat einer ganzen Zahl
quadrat :: Int -> Int          -- Typdefinition
quadrat x = x*x                -- Funktionsdefinition
```

Arbeit mit Haskell-Dateien

Haskell-Programme werden in Textdateien mit der Endung `*.hs` gespeichert. Diese lassen sich üblicherweise im Interpreter mit dem Load-Befehl `:load <Dateiname>` oder kurz `:l <Dateiname>` laden. Beim Doppelklick der Dateien auf den Schulcomputern wird dieser Teil automatisch ausgeführt. Bei der Programmierung mit Haskell arbeitet man also in einem Texteditor und parallel in einem Interpreter, um die Programme auszuführen.
Bei Änderungen am Code werden diese im Texteditor in der Haskell-Datei gespeichert und müssen danach im Interpreter neu geladen werden. Das funktioniert über `:reload` oder kurz `:r`.

Übungen

1. Erstellen Sie eine Datei `Uebung03.hs` und tippen Sie die obige Beispielfunktion ab. Laden Sie anschließend die Datei in den Interpreter und testen Sie die Funktion z. B. durch die Auswertung des Ausdrucks `quadrat 4`.

2. Definieren Sie die folgenden Funktionen in der eben erstellten Datei und testen Sie diese. Alle Funktionen erwarten als Eingabe eine ganze Zahl. Notieren Sie für alle Funktionsdefinitionen auch die Typendefinitionen.
   a) `add5` addiert zu einer Zahl den Wert 5
   b) `hoch3` berechnet die 3. Potenz einer ganzen Zahl
   c) `inc` erhöht eine Zahl um eins
   d) `istZweistellig` liefert True, wenn eine Zahl zweistellig ist

3. Binden Sie ganz zu Beginn der Datei die Bibliothek Char in Ihr Skript durch den Befehl `import Data.Char` ein und implementieren Sie die folgenden Funktionen:
   a) `toNum` wandelt einen Großbuchstaben in eine Zahl um, dabei wird ‚A' auf 0 … ‚Z' auf 25 abgebildet.
   b) `toChar` liefert das zur Zahl passende Zeichen (Umkehrung von toNum)

   Hinweis: ord liefert den ASCII-Code eines Zeichens

   Beispiele:
   toNum ‚E' --> 4 toChar 10 --> ‚K'

4. Funktionen können auch mehrere Parameter enthalten. Diese werden durch Leerzeichen voneinander getrennt angegeben.

Beispiel:

```
volumen a b c = a*b*c
```

Definieren Sie folgende Funktionen:

a) `verschieben x delta n` addiert delta zu x und gibt den Rest der Ganzzahldivision des Ergebnisses durch n zurück

b) `zylindervolumen r h` bestimmt das Volumen eines Zylinders (Datentyp: Float)

Beispiele:

```
verschieben 21 7 26 --> 2
zylindervolumen 2.5 3 --> 58.90486
```