Christoph Denzler
Daniel Kröni

Übung 2 – Typen

In dieser Übung geht es darum einfache Programmieraufgaben mit verschiedenen Basistypen von Haskell zu lösen.

1. Basis Typen

a) Implementieren Sie eine Funktion:
`threeEquals :: Int -> Int -> Int -> Bool`

welche nur dann True zurück gibt, falls alle drei Argumente gleich sind.

b) Jetzt geht es um die Funktion:
`fourEquals :: Int -> Int -> Int -> Int -> Bool`

welche nur dann True zurück gibt, falls alle vier Argumente gleich sind.

Implementieren Sie die Funktion fourEquals auf zwei verschiedenen Arten:

- Geben Sie eine Implementation welche analog zur Implementation von threeEquals ist.
- Geben Sie eine Implementation welche threeEquals aufruft um das Resultat zu berechnen.

Vergleichen Sie die beiden Implementationen.

c) Implementieren Sie eine Funktion, welche den Durchschnitt dreier Ints berechnet und als Double zurück gibt.
Hinweis: Sie benötigen die Funktion `fromIntegral`
`averageThree :: Int -> Int -> Int -> Double`

d) Implementieren Sie die Funktion `xor :: Bool -> Bool -> Bool` die nur dann True zurück gibt, wenn die beiden Argumente unterschiedlich sind.

2. Aufzählungstypen

a) Gegeben ist der Typ Op der für arithmetische Operationen steht:
```
data Op = Add | Sub
```

Definieren Sie die Funktion `calc :: Op -> Int -> Int -> Int`
Der erste Parameter bestimmt mit welcher Operation der zweite und der dritte Parameter verknüpft werden:
Bsp: `calc Add 2 3 ~> 5`

b) In dieser Aufgabe implementieren Sie eine einfache McDonalds Kasse. In unserer Filiale werden nur zwei Menus angeboten: BigMac und CheeseRoyal. Die Menus gibt es jeweils in zwei unterschiedlichen Grössen: Small und Large. Überlegen Sie, wie der Typ einer Bestellung (Order) abgebildet werden könnte und implementieren Sie dann die Funktion `price :: Order -> Int`, die den Preis einer Bestellung berechnet. Ein BigMac Menu kostet 10 CHF und ein CheeseRoyal Menu kostet 11 CHF. Die gegebenen Preise gelten für die kleinen Menus, die grossen Menus kosten jeweils zwei CHF mehr.

3. Typen bestimmen

Bestimmen Sie jeweils den allgemeinst möglichen Typen der folgenden Funktionen:

a) `swap (x, y)  = (y, x)`

b) `pair x y = (x, y)`

c) `double x = x * 2`

d) `crazy (a, '&', (b, True)) = not (a && b)`
Hinweis: Was muss der Typ von a und b sein, damit && verwendet werden kann?

e) `twice f x = f (f x)`
Hinweis: überlegen Sie sich zuerst was f eigentlich ist. Dann bestimmen Sie den Typ von f!

4. Funktionen bestimmen

Geben Sie eine beliebige legale Implementierung für folgende Definitionen:
Hinweis: Verzweifeln Sie nicht, wenn Sie eine Funktion nicht implementieren können!

a) `f1 :: Int -> Int`

b) `f2 :: (Int, Bool) -> Int`

c) `f3 :: a -> (a,Int)`

d) `f4 :: a -> b`

e) `f5 :: a -> (a ->  b) -> b`