10. Globalübung (zu Übungsblatt 12)

Inhalt

- Haskell
  - Daten- und Typkonstruktoren
  - Definition eigener Datentypen
  - Funktionen höherer Ordnung
  - unendliche Listen

Datenkonstruktoren

- dienen zum Aufbau von Objekten einer Datenstruktur
- werden bei der Datentypdefinition eingeführt
- Bezeichner beginnen mit Großbuchstaben
- Beispiele:
  - `True` und `False` (Datenstruktur: `Bool`)
  - `[]` und `:` (vordef. Datenstruktur: Listen)
  - `Nil` und `Cons` (selbstdef. Datenstruktur: Listen)

Typkonstruktoren

- Typen werden i.a. mithilfe von Typkonstruktoren aus anderen Typen erzeugt

- Beispiele:
  - `Bool, Int, Float, Char`
  - `->` (Funktionsraumkonstruktor)

Polymorphie

- parametrisch:
  - ein und dieselbe Funktion wird für Argumente verschiedener Typen verwendet.
    Beispiel: Länge einer Liste

```
len :: [a] -> Int
len []     =  0
len (x:xs) =  1 + len xs
```

- ad-hoc:
  - gleiches Funktionssymbol wird verwendet
  - aber unterschiedliche Funktionen aufgerufen

Funktionen höherer Ordnung

- haben Funktionen als Argumente oder als Resultat
- ermöglichen übersichtlichere Strukturierung von Problemen (s. Beispiel)

Beispiele für Funktionen höherer Ordnung

- Funktionskomposition
  ```
  comp :: (b->c) -> (a->b) -> (a->c)
  ```
- `curry` und `uncurry`
- `map` (sowohl Resultat als auch Argument sind Funktionen)
- `fold` (s. Beispiel)

Funktionen höherer Ordnung

- oft: Abstraktion um ähnliches Verhalten (ähnliche Rekursionsmuster) von Funktionen in einer allgemeinen Funktion zusammenzufassen
- (Bsp.: die Funktion `fold`)
- zur Abstraktion:
  - Abstraktion vom Datentyp der Listenelemente
  - Abstraktion von der Funktion, die auf jedes einzelne Listenelement angewendet werden soll

Rekursionsmuster

- Verwendung führt zu grösserer Modularisierbarkeit, Wiederverwendbarkeit und Lesbarkeit von Programmen
- Idee bei Listen:
  Durchlaufe eine Liste und wende eine Funktion auf jedes ihrer Elemente an
- Idee analog bei Bäumen

Programmieren mit unendlichen Datenobjekten

- generelles Vorgehen:
  - zuerst: potentiell unendliche Liste von Approximationen an die Lösung
  - dann: Herausfiltern der gewünschten Lösung(en)