

Listen II

Bereichslisten, unendliche Listen

Listen aus Zahlen, Zeichen und anderen Aufzähltypen können bequem durch die Angabe eines Wertebereichs erzeugt werden. Das erspart, besonders bei großen Listen, die explizite Angabe aller Elemente:

Muster	Ergebnis	Bemerkung
[2..7]	[2,3,4,5,6,7]	Natürliche Zahlen zwischen 2 und 7
['a'..'d']	['a','b','c','d']	Zeichen im Bereich 'a' bis 'd'
[0.0,0.3..1.0]	[0.1,0.3,0.6,0.9]	Dezimalzahlen zwischen 0.0 und 1.0, Schrittweite 0.3
[1,0..10]	[]	Bildungsvorschrift erzeugt keine Werte

Die Muster: [1..], [2,4..] usw. erzeugen unendliche Listen. Das mag zunächst absurd erscheinen, erweist sich jedoch für bestimmte Aufgabenstellungen als sinnvoll.

Listenbeschreibung (list comprehension)

Listenbeschreibungen generieren Listen, indem sie die Elemente anderer Listen *filtern* und *transformieren*. Die Syntax der Listenbeschreibung ist von der mathematischen Mengenschreibweise abgeleitet. Das Beispiel:

```
[x^2 | x <- [1..100], x `mod` 3 == 0]
```

entspricht der mathematischen Schreibweise:

$$\{x^2 | x \in [1..100], x \bmod 3 = 0\}$$

und liefert die Quadrate aller durch 3 teilbaren Zahlen im Intervall [1..100]. Die einzelnen Teile der Listenbeschreibung bedeuten:

Ergebnisterm	x^2	Erzeuge eine Liste mit den Elementen x^2
Generator	$x <- [1..100]$	mit $x \in [1..100]$
Filter	$x \bmod 3 == 0$	und $x \bmod 3 = 0$

Eine Listenbeschreibung besitzt mindestens einen Generator. Filter und weitere Generatoren sind optional.

Höhere Listenfunktionen

Viele Listenfunktionen besitzen die gleiche Struktur, wie z.B.:

<pre> quadrade :: Num a => [a]->[a] quadrade [] = [] quadrade (x:xs) = x^2:quadrade xs </pre>	<pre> ascii :: [Int] -> [Char] ascii [] = [] ascii (x:xs) = chr x:ascii xs </pre>
„Quadriere <u>alle Elemente einer Liste</u> “.	„Wandle <u>alle Elemente einer Liste</u> in ihr ASCII-Zeichen um“.
Allgemein: Transformiere alle Elemente einer Liste mit einer Funktion f.	

Für diese Klasse von Funktionen existiert die höhere Listenfunktion map:

Signatur: $\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$

\uparrow \uparrow \uparrow
Eingabefunktion Eingabeliste Ergebnisliste

Allgemeine Form: $ys = \text{map } f \ xs$

Sie erhält als Argumente eine Funktion vom Typ $f :: a \rightarrow b$ und eine Liste $[a]$. Das Ergebnis ist eine Liste $[b]$. Die beiden Beispielfunktionen lassen sich mit map folgendermaßen definieren:

```

quadrade xs = map (^2) xs
ascii xs = map chr xs

```

Weitere Beispiele für höhere Listenfunktionen sind die Faltungsfunktionen foldr und foldl.

Höhere Listenfunktionen mit Prädikaten

Prädikate sind Testfunktionen vom Typ $a \rightarrow \text{Bool}$, wie z.B.:

```

istGerade x
| x `mod` 2 == 0 = True
| otherwise      = False

```

oder kürzer:

```

istGerade x = x `mod` 2 == 0

```

Eine Reihe höherer Listenfunktionen verwenden diese Prädikate zur Extraktion von Teillisten mit bestimmten Eigenschaften. Ein wichtiger Vertreter ist die Funktion filter:

Signatur: $\text{filter} :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$

\uparrow \uparrow \uparrow
Prädikat Eingabeliste Ergebnisliste

Allgemeine Form: $ys = \text{filter } p \ xs$

Sie bildet eine Liste aus allen Elementen der Eingabeliste, die das Prädikat p erfüllen. Beispiele:

<code>geradeZahlen xs = filter even xs</code>	Erzeugt eine Liste mit allen geraden Zahlen aus <code>xs</code> .
<code>kleinerAls n xs = filter (<n) xs</code>	Erzeugt eine Liste mit allen Zahlen aus <code>xs</code> , die kleiner als <code>n</code> sind.
<code>istGleich n xs = filter (=n) xs</code>	Erzeugt eine Liste mit allen Elementen von <code>xs</code> , die gleich <code>n</code> sind.

Weitere Beispiele für höhere Listenfunktionen mit Prädikaten sind `takeWhile` und `dropWhile`:

```
ys = takeWhile p xs  
ys = dropWhile p xs
```