

Benutzung und Erzeugung von Listen in Haskell

Zur Erinnerung

Listen werden in eckigen Klammern angegeben und die (List-)Elemente durch Kommas getrennt, z.B. [1,2,3]. Die Elemente können immer nur vom gleichen Typ sein. Listen können leer sein: []. Die verkürzte Schreibweise [1..5] wird von Haskell als [1,2,3,4,5] interpretiert.

Es gibt mehrere Möglichkeiten, eine Liste zu erzeugen.

Die einfachste ist sicherlich die direkte Angabe der Liste, z.B. [34, 56, 87, 98, 23445567, 87654, 1, 0]

Beachte: bei einer Liste müssen alle Elemente den gleichen Typ haben.

Dies steht im Gegensatz zu Tupeln, z.B. ("Hallo",34,True, 0.78765) oder [(„ZZ“,123),(„AB“,34),(„True“,7)]¹

Wenn man Listenelemente mit gleichen Abständen benötigt, so bietet sich die Definition mit .. an:

[1..11] enthält alle ganze Zahlen von 1 bis 11: [1,2,3,4,5,6,7,8,9,10,11]

Dabei lässt sich die Schrittweite angeben: [1,3..11] erzeugt [1,3,5,7,9,11].

Man beachte hierbei, dass die Vervollständigung mit zwei Punkten .. erfolgen muss und kein Komma nach dem zweiten Element erfolgt.

Das geht auch mit Floats: [1.5,3.5..11] ergibt [1.5,3.5,5.5,7.5,9.5,11.5]

Es lässt sich sogar die Liste aller Kleinbuchstaben erzeugen:

['a'..'z'] ergibt "abcdefghijklmnopqrstuvwxy".

Man beachte hierbei die Verwendung des ' um ein Char zu kennzeichnen.

Auf diese Listen lassen sich selbstverständlich die Basis-Funktionen von Haskell für Listen anwenden:

```
>[34, 56, 87, 98, 23445567, 87654, 1, 0]!!4
>23445567
```

```
umkehren xs = reverse xs
>umkehren [34, 56, 87, 98, 23445567, 87654, 1, 0]
>[0, 1, 87654, 23445567, 98, 87, 56, 34]
```

Erzeugung von Listen mit Hilfe des Listengenerators (Listcomprehension)

Interessant wird es, wenn wir Listen auf Grund von bestimmten Bedingungen erzeugen können. Dies geschieht mit Hilfe des Listengenerators (*Listcomprehension*).

Beispiele

allegeradenZahlenbis n = [geradezahl | geradezahl <- [1..n], mod geradezahl 2 == 0]

oder, wenn Start und Ende angegeben werden:

allegeradenZahlenVonBis start ende = [geradezahl | geradezahl <- [start..ende], mod geradezahl 2 == 0]

[x | x <- [1 .. 20], (mod x 3) == 0]

damit gibt man Haskell die Anweisung, eine Liste zu bilden aus allen Elementen, die genau durch 3 geteilt werden können. Lies: „Erstelle eine Liste von Werten für x, für die gilt, dass der aktuelle Wert von x die Bedingung mod 3 == 0 erfüllt“. Es werden somit nur die Werte in die Liste aufgenommen, die die (alle) Bedingung(en) erfüllen.

Mehrere Bedingungen müssen durch Komma getrennt angegeben werden.

Im Bildungsbereich der Listenelemente (ganz links vor dem | Zeichen), lassen sich beliebige Arten von Listenelementen bilden, hier das obige Beispiel mit einem (2er-)Tupel-Element, dass den x-Wert, und sein Quadrat enthält:

[(x , x*x) | x <- [1 .. 20], (mod x 3) == 0]

Es lassen sich Werte auch aus mehreren Listen speisen: [(x,y) | x <- [1 .. 26], y <- ['a'..'z']] --verschachtelte Listen

Aufgabe: Probieren Sie dieses Beispiel von ineinander verschachtelten Listen aus und interpretieren Sie das Ergebnis. Welche Speisung ließe sich als *innere*, welche als *äußere* Liste bezeichnen?

Die Lösung für die Bakterienfläche (vgl. Aufgabe aus dem Unterricht) ließe sich mit Hilfe eines Listgenerators ermitteln:

loesungB1 start ende = [(stunde, baktFlaeche 0.2 5 stunde) | stunde <- [start..ende]]

Aufgabe: wie ließe sich die Lösungsmenge durch einen (oder mehrere) Filter weiter einschränken bzw. reduzieren?²

¹ Beachte, dass im Gegensatz dazu [(„ZZ “,123),(„AB “,34),(True,False)] keine korrekte Liste ist! – Warum?

² b69lc3VuZ0lyIHNOYXJ0IGVuzGUgPjVoc3R1bmRlGHa3RGRGFFY2hllDAuMlA1IHNOdW5kZSkGfCBzdHVuZGUgPC1bc3RhcncQuLmVuZGVdLCAoYmFrdEZsYWVjaGUGMc4yDUJgc3R1bmRlRKSAA+IDAuNF0=