Real World Haskell
Blatt 9

Julian Fleischer, Alexander Steen

Mittwoch, den 07. 08. 2013

Gegeben (**Matrix**)

```
matAt :: (Nat m, Nat n) => Matrix m n a -> (Int, Int) -> a
matAt (Matrix cells) (m, n) = cells !! pred m !! pred n

matMult :: forall m n o a. (Nat m, Nat n, Nat o, Num a)
        => Matrix m n a -> Matrix n o a -> Matrix m o a
matMult m1 m2 = mkMatrix __ __ (\(i,j) -> sum [at1 i z * at2 z j | z <- [1..toInt (__ :: n)]])
  where
    at1 i j = m1 `matAt` (i,j)
    at2 i j = m2 `matAt` (i,j)
```

Vollständig unter `http://page.mi.fu-berlin.de/scravy/realworldhaskell/tag11/Matrix.hs`

Aufgabe 1 (**Strict State Threads**)

Machen Sie sich mit der `ST` Monade[1] vertraut (`Control.Monad.ST`). Schreiben Sie `matMult` unter Verwendung von `ST` und einem veränderbaren (*mutable*) Array[2] um!

Aufgabe 2 (**Quickcheck**)

Schreiben Sie einen Quickcheck Unit Test der ihre Funktion mit der vorhandenen matMult vergleicht!

Das Package: `http://hackage.haskell.org/package/QuickCheck`

Haskellwiki: `http://www.haskell.org/haskellwiki/QuickCheck`

[1] `http://hackage.haskell.org/packages/archive/base/4.5.0.0/doc/html/Control-Monad-ST.html`
[2] `http://hackage.haskell.org/packages/archive/array/0.4.0.1/doc/html/Data-Array-ST.html`