7. Übung zur Vorlesung "Fortgeschrittene funktionale Programmierung"

Funktoren und IO-Monade

Boris Dudelsack

17. November 2016

Aufgabe 1: Funktoren

```haskell
instance Functor Maybe where
  fmap Nothing = Nothing
  fmap f (Just x) = Just (f x)

instance Functor Tree where
  fmap Empty = Empty
  fmap f x v y = Node (fmap x) f v (fmap y)

instance Functor (Either e) where
  fmap f (Right x) = Right (f x)
  fmap _ (Left x) = Left x
```

Aufgabe 2: Either

```haskell
readInt :: String → Either String Int
readInt str = case reads str of
  [(n,"")] → Right n
  _        → Left err
  where
    err = "Die Zeichenkette " ++ str ++ " ist keine Zahl"

rangeCheck :: Int → Int → Int → Either String Int
rangeCheck x y a
  | a ≥ x && a ≤ y = Right a
  | otherwise      = Left err
  where
    err = "Die Zahl " ++ show a ++ " liegt ausserhalb des Bereiches"

readIntRange :: Int → Int → String → Either String Int
readIntRange x y str = readInt str >≥ rangeCheck x y
```

Aufgabe 3: Interaktives Menü

```haskell
isRight :: Either a b → Bool
isRight (Right x) = True
isRight _ = False

unpackEither :: Either a b → b
unpackEither (Right x) = x

menu :: [String] → IO Int
menu xs = do
  mapM_ (\(i,s) → putStrLn (show i ++ ". " ++ s ))(zip [1 .. ] xs)
  putStr "Please make a choise: "
  s ← getLine
  let i = readIntRange 1 (length xs) s
  if isRight i
      then return (unpackEither i)
      else do
        putStrLn "Your selection is out of range. Please try again."
        menu xs

menuSelect :: [(String, IO a)] → IO a
menuSelect xs = do
  let items = map fst xs
  let ios = map snd xs
  i ← menu items
  ios!!(i-1)
```

Aufgabe 4: Word-Count

```haskell
m :: [(String, String → Int)]
m = [
  ("Count chars in file", length),
  ("Count words in file", length . words),
  ("Count lines in file", length . lines),
  ("Quit", const 0)]

wordCount :: FilePath → IO ()
wordCount fp = do
  c ← readFile fp
  menuSelect (map (\(l,f) → (l, print (f c))) m)
```