Operatsioone listidega

- Listi iga elemendi korrutamine kahega

```
doubleList :: [Int] -> [Int]
doubleList []     = []
doubleList (x:xs) = (2*x) : doubleList xs
```

- Stringi kõikide tähtede muutmine suurteks

```
upperString :: String -> String
upperString []     = []
upperString (c:cs) = toUpper c : upperString cs
```

- Mõlemal juhul on rekursiooniskeem täpselt sama!

Operatsioone listidega

- Funktsioon `map`

$$\texttt{map}\ f\ [x_1, x_2, \ldots, x_n] \quad \Longrightarrow \quad [f\ x_1, f\ x_2, \ldots, f\ x_n]$$

- Definitsioon:

```
map          :: (a -> b) -> [a] -> [b]
map f []      = []
map f (x:xs) = f x : map f xs
```

- Näited:

```
doubleList  = map (\x -> 2*x)
upperString = map toUpper
```

- Kehtib võrdus

```
map f xs = [f x | x <- xs]
```

Operatsioone listidega

- Listi elementide summa

```
sum :: [Int] -> Int
sum []     = 0
sum (x:xs) = x + sum xs
```

- Listide listi konkateneerimine

```
concat :: [[a]] -> [a]
concat []       = []
concat (xs:xss) = xs ++ concat xss
```

- Jällegi on rekursiooniskeemid samad!

Operatsioone listidega

- Funktsioon `foldr`

$$\texttt{foldr}\ (\oplus)\ e\ (x_1 : (x_2 : (x_3 : [])))$$
$$\Longrightarrow (x_1 \oplus (x_2 \oplus (x_3 \oplus e)))$$

- Definitsioon:

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f b []     = b
foldr f b (x:xs) = f x (foldr f b xs)
```

- `foldr` näiteid:

```
sum    = foldr (+) 0
concat = foldr (++) []
and    = foldr (&&) True
or     = foldr (||) False
length = foldr (\_ n -> 1+n) 0
map f  = foldr (\x ys -> f x : ys) []
```

Operatsioone listidega

- Täisarvude listi maksimaalne element

```
maximum :: [Integer] -> Integer
maximum = foldr max ???
```

- Sobivat baaselementi ei leidu!
- Funktsioon `foldr1`

```
foldr1 :: (a->a->a) -> [a] -> a
foldr1 f [x]    = x
foldr1 f (x:xs) = f x (foldr1 f xs)
```

- `foldr1` näiteid:

```
maximum = foldr1 max
minimum = foldr1 min
```

Operatsioone listidega

- Funktsioon `foldl`

$$\texttt{foldl}\ (\oplus)\ e\ (x_1 : (x_2 : (x_3 : [\,]))) \Longrightarrow (((e \oplus x_1) \oplus x_2) \oplus x_3)$$

- Definitsioon:

```
foldl :: (a -> b -> a) -> a -> [b] -> a
foldl f a []     = a
foldl f a (x:xs) = foldl f (f a x) xs
```

- `foldl` näiteid:

```
sum     = foldl (+) 0
product = foldl (*) 1
length  = foldl (\ a _ -> 1+a) 0
reverse = foldl (\ a x -> x:a) []
decimal = foldl (\ a x -> 10*a+x) 0
```

Operatsioone listidega

- Funktsioon scanl

```
scanl :: (a -> b -> a) -> a -> [b] -> [a]
scanl f a []     = [a]
scanl f a (x:xs) = a : scanl f (f a x) xs
```

- Funktsioon filter

```
filter :: (a -> Bool) -> [a] -> [a]
filter p [] = []
filter p (x:xs) | p x       = x : filter p xs
                | otherwise = filter p xs
```

- Funktsioon zipWith

```
zipWith :: (a->b->c) -> [a] -> [b] -> [c]
zipWith f (x:xs) (y:ys) = f x y : zipWith f xs ys
zipWith _     _      _  = []
```

Kalender

- Väljastada etteantud aasta kalender

```
Calendar> calendar 2003
January 2003              February 2003             March 2003

Sun     5 12 19 26        Sun     2  9 16 23        Sun     2  9 16 23 30
Mon     6 13 20 27        Mon     3 10 17 24        Mon     3 10 17 24 31
Tue     7 14 21 28        Tue     4 11 18 25        Tue     4 11 18 25
Wed  1  8 15 22 29        Wed     5 12 19 26        Wed     5 12 19 26
Thu  2  9 16 23 30        Thu     6 13 20 27        Thu     6 13 20 27
Fri  3 10 17 24 31        Fri     7 14 21 28        Fri     7 14 21 28
Sat  4 11 18 25           Sat  1  8 15 22           Sat  1  8 15 22 29

April 2003                May 2003                  June 2003

Sun     6 13 20 27        Sun     4 11 18 25        Sun  1  8 15 22 29
Mon     7 14 21 28        Mon     5 12 19 26        Mon  2  9 16 23 30
Tue  1  8 15 22 29        Tue     6 13 20 27        Tue  3 10 17 24
Wed  2  9 16 23 30        Wed     7 14 21 28        Wed  4 11 18 25
Thu  3 10 17 24           Thu  1  8 15 22 29        Thu  5 12 19 26
Fri  4 11 18 25           Fri  2  9 16 23 30        Fri  6 13 20 27
Sat  5 12 19 26           Sat  3 10 17 24 31        Sat  7 14 21 28
```

Pildid

- Piltide esitamine

```
type Picture   =  [[Char]]
```

- Piltide atribuudid

```
height, width :: Picture -> Int
height p       = length p
width  p       = length (head p)
```

- Tühja pildi konstrueerimine

```
empty      :: (Int,Int) -> Picture
empty (h,w) = replicate h (replicate w ' ')
```

- Pildi väljastamine

```
printPicture :: Picture -> IO ()
printPicture  = putStr . unlines
```

```
[ "    ##    ",
  "   #  #   ",
  "  #    #  ",
  " ######## ",
  "    ##    "]
```

Piltide kombinaatorid

- Piltide vertikaalne kompositsioon

- Definitsioon

```
($$) :: Picture -> Picture -> Picture
($$)  = (++)
```

- Piltide listi vertikaalne kompositsioon

```
stack :: [Picture] -> Picture
stack  = foldr1 ($$)
```

Piltide kombinaatorid

- Piltide horisontaalne kompositsioon

```
[ "   ##   ",     [ "   ##   ",      [ "   ##       ##   ",
  "  #  #  ",       "  #  #  ",        "  #  #     #  #  ",
  " #    # ",  <>   " #    # ",  ==>   " #    #   #    # ",
  " ######## ",     " ######## ",      " ########  ######## ",
  "   ##   "]       "   ##   "]        "   ##       ##   "]
```

- Definitsioon

```
(<>) :: Picture -> Picture -> Picture
(<>)  = zipWith (++)
```

- Piltide listi horisontaalne kompositsioon

```
spread :: [Picture] -> Picture
spread  = foldr1 (<>)
```

Piltide paigutamine

- Piltide tabelisse paigutamine

```
block, blockT :: Int -> [Picture] -> Picture
block n        = stack . map spread . groups n
blockT n       = spread . map stack . groups n

groups       :: Int -> [a] -> [[a]]
groups n []   = []
groups n xs   = take n xs : groups n (drop n xs)
```

- Pildi paigutamine tühja pildi vasakpoolsesse ülemisse nurka

```
lframe          :: (Int,Int) -> Picture -> Picture
lframe (m,n) p =    (p <> empty (h,n-w))
                 $$ empty (m-h,n)
                 where h = height p
                       w = width p
```

Kalender

- Kalendri põhifunktsioon

```
calendar :: Int -> IO ()
calendar =   printPicture . block 3
           . map picture  . months
```

- Ühe kuu pildi konstrueerimine

```
picture (mn,yr,fd,ml)  = title mn yr $$ table fd ml
```

- Ühe kuu pildi päis

```
title mn yr = lframe (2,25) [mn ++ " " ++ show yr]
```

- Ühe kuu ülejäänud pilt

```
table fd ml = lframe (8,25) (dnames <> entries fd ml)
```

- Nädalapäevade nimed

```
dnames = ["Sun","Mon","Tue","Wed","Thu","Fri","Sat"]
```

Kalender

- Kuupäevade tabeli konstrueerimine

```
entries fd ml = blockT 7 (dates fd ml)
```

- Kuupäevade listi genereerimine

```
dates fd ml = map (date ml) [1-fd..42-fd]
```

- Liigsete kuupäevade eemaldamine

```
date ml d | d<1 || ml<d  = ["   "]
          | otherwise    = [rjustify 3 (show d)]
```

- Paremale reastamine

```
rjustify n s = replicate (n - length s) ' ' ++ s
```

Kalender

- Kalendri konstrueerimine

```
months year = zip4 mnames (replicate 12 year)
                   (fstdays year) (mlengths year)
```

- Kuude nimed

```
mnames = ["January","February","March","April",
          "May","June","July","August","September",
          "October","November","December"]
```

- Kuude pikkused

```
mlengths year = [ 31, feb, 31, 30, 31, 30,
                  31,  31, 30, 31, 30, 31]
                where feb | leap year = 29
                          | otherwise = 28
```

Kalender

- Kas on liigaasta?

```
leap year | year`mod`100 == 0  = year`mod`400 == 0
          | otherwise           = year`mod`4   == 0
```

- Aasta esimese päeva leidmine

```
jan1st year = (year + last`div`4
                    - last`div`100
                    + last`div`400) `mod` 7
              where last = year - 1
```

- Kuude esimeste päevade leidmine

```
fstdays year = take 12
                    (map (`mod`7)
                         (scanl (+) (jan1st year)
                                    (mlengths year)))
```