Ilutrükk

- Ilutrükkimise (ingl. pretty printing) eesmärgiks on:
  - programmi/andmestruktuuri teisendamine stringiks;
  - tulemus peaks olema loetava paigutusega.
- Ilutrükkimise soovitavad omadused:
  - vahendid erinevate paigutuste esitamiseks;
  - sobivate paigutuste valik vastavalt teksti soovitud laiusele;
  - efektiivsus.

Ilutrükk kasutades stringe

- Näide: aritmeetilised avaldised

```
data Expr = Num Int | Add Expr Expr


showExpr (Num x)      = show x
showExpr (Add e1 e2) = showExpr e1 ++ "+" ++ showExpr e2
```

- Väga ebaefektiivne (ruutkeerukusega!)
- Loetavate paigutuste tekitamine keeruline

Ilutrükk ver. 1 — abstraktne liides

- Dokumentide konstrueerimine

```
nil   :: Doc
text  :: String -> Doc
(<>)  :: Doc -> Doc -> Doc
```

- Dokumentide trükkimine

```
pretty :: Doc -> String
```

- Näide:

```
showExpr = pretty . showE
   where showE (Num x)     = text (show x)
         showE (Add e1 e2) = showE e1 <> text "+" <> showE e2
```

Ilutrükk ver. 1 — realisatsioon

- Dokumendi esitamine

```
data Doc = Nil
         | Text String
         | Doc :<> Doc
```

- Dokumentide konstrueerimine

```
nil  = Nil
text = Text
(<>) = (:<>)
```

- Dokumentide trükkimine

```
pretty d = convert [d]

convert []                = ""
convert (Nil : ds)        = convert ds
convert (Text s : ds)     = s ++ convert ds
convert (d1 :<> d2 : ds) = convert (d1:d2:ds)
```

Ilutrükk ver. 2 — paigutus ja taanded

- Abstraktne liides

```
line :: Doc
nest :: Int -> Doc -> Doc
```

- Näide:

```
data Tree = Node String [Tree]

tree = Node "aa" [ Node "b" [ Node "c" [] ],
                   Node "dd" [],
                   Node "e" [ Node "f" [] ] ]
```

Ilutrükk ver. 2 — paigutus ja taanded

- Näide (järg):

```
ppTree (Node s ts) = text s <> nest (length s) (ppBracket ts)
  where ppBracket [] = nil
        ppBracket ts = text "[" <> nest 1 (ppTrees ts) <> text "]"
        ppTrees [t]    = ppTree t
        ppTrees (t:ts) = ppTree t <> text "," <> line <> ppTrees ts
```

```
Main> putStr . pretty . ppTree $ tree
aa[b[c],
   dd,
   e[f]]
```

Ilutrükk ver. 2 — paigutus ja taanded

- Näide (järg):

```
ppTree (Node s ts) = text s <> ppBracket ts
  where ppBracket [] = nil
        ppBracket ts = text "[" <> nest 2 (line <> ppTrees ts)
                                <> line <> text "]"

Main> putStr . pretty . ppTree $ tree
aa[
  b[
    c
  ],
  dd,
  e[
    f
  ]
]
```

Ilutrükk ver. 2 — realisatsioon

- Dokumendi esitamine ja konstrueerimine

```
data Doc = ...
         | Line
         | Nest Int Doc
line = Line
nest = Nest
```

- Dokumentide trükkimine

```
pretty d = convert [(d,0)]

convert []                      = ""
convert ((Nil,i) : ds)          = convert ds
convert ((Text s,i) : ds)       = s ++ convert ds
convert ((d1 :<> d2,i) : ds) = convert ((d1,i):(d2,i):ds)
convert ((Line,i) : ds)         = "\n" ++ copy i ' ' ++ convert ds
convert ((Nest n d,i) : ds)  = convert ((d,n+i):ds)
```

Ilutrükk ver. 3 — paigutuste valik

- Abstraktne liides

```
group :: Doc -> Doc
pretty :: Int -> Doc -> String
```

- Näide:

```
ppTree (Node s ts) = group (text s <> nest (length s) (ppBracket ts))
```

```
Main> putStr . pretty 10 . ppTree $ tree
aa[b[c],
   dd,
   e[f]]

Main> putStr . pretty 20 . ppTree $ tree
aa[b[c], dd, e[f]]
```

Ilutrükk ver. 3 — realisatsioon

- Dokumendi esitamine ja konstrueerimine

```
data Doc = ...
         | Doc :<|> Doc

group x = flatten x :<|> x

flatten Nil           = Nil
flatten (Text s)      = Text s
flatten (d1 :<> d2)   = flatten d1 :<> flatten d2
flatten Line          = Text " "
flatten (Nest i d)    = Nest i (flatten d)
flatten (d1 :<|> d2) = flatten d1
```

Ilutrükk ver. 3 — realisatsioon

- Dokumentide trükkimine

```
pretty w d = convert w 0 [(d,0)]

convert w c []                      = ""
convert w c ((Nil,i) : ds)          = convert w c ds
convert w c ((Text s,i) : ds)       = s ++ convert w (c + length s) ds
convert w c ((d1 :<> d2,i) : ds) = convert w c ((d1,i):(d2,i):ds)
convert w c ((Line,i) : ds)  = "\n" ++ copy i ' ' ++ convert w i ds
convert w c ((Nest n d,i) : ds) = convert w c ((d,n+i):ds)
convert w c ((d1 :<|> d2,i) : ds)
                              = better w c (convert w c ((d1,i):ds))
                                           (convert w c ((d2,i):ds))
```

- Parima dokumendi valimine

```
better w c s1 s2 = if fits (w-c) s1 then s1 else s2
fits w xs = w >= 0 && (null xs || head xs == '\n'
                               || fits (w-1) (tail xs))
```