Лекция 6

Характерни приложения на списъците

Текстообработка

Една от най-важните задачи при обработката на „чисти” текстове (plain texts) е подреждането на текста в редове с определена дължина.

Първоначалният текст може да е разпокъсан или недобре подреден. Задачата е той да бъде разделен на редове със зададена дължина и след това евентуално да бъде подравнен чрез добавяне на интервали на подходящи места между думите.

Разделяне на входния текст на думи

Ще предполагаме, че текстът, който трябва да се форматира, е зададен под формата на символен низ.

Първата задача е входният низ да бъде разделен на думи.

Дума е всяка поредица от знакове, която не включва разделители (the whitespace characters space, tab and newline):

```
-- The “whitespace” characters.

whitespace :: String
whitespace = ['\n','\t',' ']
```

Най-напред ще дефинираме функцията getWord, която връща като резултат първата дума от даден низ, ако този низ започва с дума (а не с разделител), или празен низ – в противния случай.

Примери
getWord “ boo” ⟶ “”
getWord “cat dog” ⟶ “cat”

В дефиницията на функцията getWord ще използваме стандартната (вградената) функция ***elem***, която проверява дали даден обект е елемент на даден списък:
elem ‘a’ whitespace ⟶ False
elem ‘ ‘ whitespace ⟶ True

Дефиниция:

```
-- Get a word from the front of a string.

getWord :: String -> String
getWord []     = []
getWord (x:xs)
  | elem x whitespace   = []
  | otherwise           = x : getWord xs
```

Проследяване на изпълнението на примерно обръщение към функцията getWord:

getword "cat dog"
⟶ 'c' : getword "at dog"
⟶ 'c' : 'a' : getword "t dog"
⟶ 'c' : 'a' : 't' : getword " dog"
⟶ 'c' : 'a' : 't' : []
⟶ "cat"

По сходен начин ще дефинираме функция, която отделя („изтрива”) първата дума от даден символен низ (в случай, че този низ започва с дума, а не с разделител):

```
dropWord :: String -> String
dropWord []     = []
dropWord (x:xs)
  | elem x whitespace   = (x:xs)
  | otherwise           = dropWord xs
```

Примери

dropWord “cat dog” ⟶ “ dog”
dropWord “ dog” ⟶ “ dog”

Отделянето на водещите разделители от даден текст (низ) ще може да се извършва с помощта на функцията dropSpace:

```
-- To remove the whitespace character(s) from the
-- front of a string.

dropSpace :: String -> String
dropSpace []     = []
dropSpace (x:xs)
  | elem x whitespace   = dropSpace xs
  | otherwise           = (x:xs)
```

Нека предположим, че е даден символен низ ***st***, който не съдържа разделители в началото си.

Тогава разделянето на ***st*** на (списък от съдържащите се в него) думи може да се извърши по следния начин:

- първата дума може да се отдели чрез прилагане на getWord към st;
- останалите думи могат да бъдат получени чрез отделяне на поредицата от разделители в началото на останалата (след отделянето на първата дума) част на st, последвано от прилагане на описваната операция.

Дефиниция:

```
-- A word is a string.

type Word = String

-- Splitting a string into words.

splitWords :: String -> [Word]
splitWords st = split (dropSpace st)

split :: String -> [Word]
split [] = []
split st
  = (getWord st) : split (dropSpace (dropWord st))
```

Пример

splitWords " dog cat"

⟶ split "dog cat"

⟶ (getWord "dog cat")
: split (dropSpace (dropWord "dog cat"))

⟶ "dog" : split (dropSpace " cat")

⟶ "dog" : split "cat"

⟶ "dog" : (getWord "cat")
: split (dropSpace (dropWord "cat"))

⟶ "dog" : "cat" : split (dropSpace [])

⟶ "dog" : "cat" : split []

⟶ "dog" : "cat" : []

⟶ ["dog" , "cat"]

Разделяне на текста на редове

Сега ще покажем как даден текст (по-точно, даден списък от думи) може да бъде разделен на редове с дължина, не по-голяма от lineLen:

```
lineLen :: Int
lineLen = 80

-- A line is a list of words.

type Line = [Word]
```

Функцията getLine извлича първия ред от даден списък от думи:

- ако списъкът от наличните думи е празен, то се формира и връща като резултат празен ред;
- ако първата налична дума е w, тя се включва в реда, ако има достатъчно място за нея (нейната дължина трябва да бъде не по-голяма от дължината на реда). В такъв случай остатъкът от реда се запълва от (част от) оставащите думи, като се предвиди място за поне един интервал след първата дума;
- ако първата дума не може да се побере в реда, то редът остава празен.

Дефиниция:

```
-- Getting a line from a list of words.

getLine :: Int -> [Word] -> Line
getLine len []      = []
getLine len (w:ws)
  | length w <= len     = w : restOfLine
  | otherwise           = []
    where
    newlen      = len - (length w + 1)
    restOfLine  = getLine newlen ws
```

Пример

getLine 20 [“Mary”, “Poppins”, “looks”, “like”, …]

⟶ “Mary” : getLine 15 [“Poppins”, “looks”, “like”, …]

⟶ “Mary” : “Poppins” : getLine 7 [“looks”, “like”, …]

⟶ “Mary” : “Poppins” : “looks” : getLine 1 [“like”, …]

⟶ “Mary” : “Poppins” : “looks” : []

⟶ [“Mary”, “Poppins”, “looks”]

По аналогия с функцията dropWord може да се дефинира и функция dropLine, която отделя (“изтрива”) първия ред от даден списък от думи.

```
-- Dropping the first line from a list of words.

dropLine :: Int -> [Word] -> [Word]

-- dropLine = .....
```

Тогава функцията splitLines, която разделя даден списък от думи на редове с дължина най-много lineLen, може да бъде дефинирана както следва:

```
-- Splitting into lines.

splitLines :: [Word] -> [Line]
splitLines [] = []
splitLines ws
  = getLine lineLen ws
          : splitLines (dropLine lineLen ws)
```

Заключение

Разделянето на даден текст (символен низ) на редове може да се извърши с помощта на функцията fill:

```
-- To fill a text string into lines, we write

fill :: String -> [Line]
fill = splitLines . splitWords
```

Конвертирането на резултата в подходящ символен низ може да се извърши с помощта на подходяща функция, например

```
joinLines :: [Line] -> String
```

Освен тази функция следва да се дефинира също така и функция, която извършва подравняването на редовете (добавянето на допълнителни интервали между думите с цел подравняване на текста от дясно).