Робота № 1

Визначення рекурсивних функцiй

Варіант № 12

**Мета роботи**

Набути досвiду визначення рекурсивних функцiй, використання механiзму

**Завдання**

Визначте вказанi функцiї в кожному з завдань:

а) без застосування,

б) із застосуванням вбудованих функцiй.

1.12 Вставити у список через кожнi n елементiв вказане значення, напр.

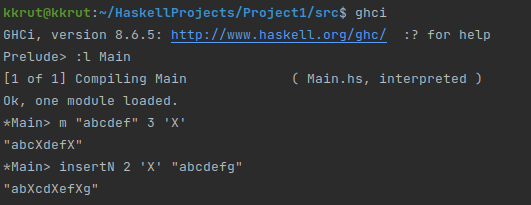
через n=2 значення ’z’: "1234590"⇒ "12z34z59z0".

**Код програми**

module Main (main) where  
  
import Data.List  
import Data.Numbers.Primes  
  
  
main :: IO ()  
main = do  
 putStrLn "hello world"  
  
  
{-  
 Визначте вказанi функцiї в кожному з завдань: а) без застосування, б) з  
 застосуванням вбудованих функцiй.  
  
 1.12 Вставити у список через кожнi n елементiв вказане значення, напр.  
 через n=2 значення ’z’: "1234590"⇒ "12z34z59z0".  
  
 2.12 Знайти перше просте число в указаному дiапазонi.  
-}  
  
  
-- 1.12 ----------------------------------------------------------------------------------------------------------------  
  
*-- | Фунція для вставлення значення у список через кожні n елементiв*  
*-- >> insertN 3 0 [3..12]*  
*-- > [3,4,5,0,6,7,8,0,9,10,11,0,12]*  
*--*  
*-- >> insertN 2 'z' "abcdefg"*  
*-- > "abzcdzefzg"*  
insertN :: Int -> a -> [a] -> [a]  
insertN n el list  
 | n == 0 || length list < n = list  
 | null list = []  
 | otherwise = take n list ++ [el] ++ insertN n el (drop n list)  
  
  
*-- | Функція для індексування списку елементів*  
*-- >> indexed "abcdefg"*  
*-- > [('a',1),('b',2),('c',3),('d',4),('e',5),('f',6),('g',7)]*  
*--*  
*-- >> indexed [1..9]*  
*-- > [(1,1),(2,2),(3,3),(4,4),(5,5),(6,6),(7,7),(8,8),(9,9)]*  
indexed :: [a] -> [(a, Int)]  
indexed ls = zip ls [1..length ls]

*-- | Функція для перевірки якого елементу додавати*  
*-- >> checkN ('a', 4) 2 '0'*  
*-- > "0"*  
checkN :: (a, Int) -> Int -> a -> [a]  
checkN x n el  
 | snd x `mod` n == 0 = [el]  
 | otherwise = []  
  
  
*-- | Функція для конкатенації елементу списку з необхідним елементом*  
*-- >> generateEl ('a', 8) 2 'X'*  
*-- > "aX"*  
*--*  
*-- >> (4, 8) 2 0*  
*-- > [4,0]*  
generateEl :: (a, Int) -> Int -> a -> [a]  
generateEl x n el = [fst x] ++ (checkN x) n el  
  
  
*-- | Функція для конкатенації списку з потрібним елементом у вказаній позиції*  
*-- >> mapping [('a',1),('b',2),('c',3),('d',4),('e',5),('f',6),('g',7)] 3 'X'*  
*-- > "abcXdefXg*  
*--*  
*-- >> mapping [(1,1),(2,2),(3,3),(4,4),(5,5),(6,6),(7,7),(8,8),(9,9)] 3 0*  
*-- > [1,2,3,0,4,5,6,0,7,8,9,0]*  
mapping :: [(a, Int)] -> Int -> a -> [a]  
mapping ls n el = concat (map (\x -> generateEl x n el) ls)  
  
  
*-- | Фунція для вставлення значення у список через кожні n елементiв*  
*-- >> m "abcdef" 3 'X'*  
*-- > "abcXdefX"*  
*--*  
*-- >> m [1..9] 3 0*  
*-- > [1,2,3,0,4,5,6,0,7,8,9,0]*  
m :: [a] -> Int -> a -> [a]  
m ls n el = mapping (indexed ls) n el  
  
  
-- 2.12 ----------------------------------------------------------------------------------------------------------------  
  
*-- | Функція для перевірки на просте число*  
*-- >> isPrime2 11*  
*-- > True*  
isPrime2 :: Integer -> Bool  
isPrime2 k = length [ x | x <- [2..k], k `mod` x == 0] == 1  
  
*-- | Функція для знаходження першого простого числа в масиві чисел*  
*-- | з використанням вбудованої функції isPrime з модуля Primes модуля Numbers*  
*-- >> getPrime [2..11]*  
*-- > Just 2*  
getPrime :: [Integer] -> Maybe Integer  
getPrime list = find isPrime list  
  
  
*-- | Функція для знаходження першого простого числа в масиві чисел*  
*-- | з використанням функції isPrime2*  
*-- >> getPrime2 [4, 6]*  
*-- > Nothing*  
getPrime2 :: [Integer] -> Maybe Integer  
getPrime2 [] = Nothing  
getPrime2 (x:xs) | isPrime2 x = Just x  
 | otherwise = getPrime2 xs

**Приклад роботи**



**Висновок**: під час виконання цієї лабораторної работи було освоєно основи розробки лексичного аналізатору, набуто навички з його розробки та використано їх на практиці. В результаті виконання роботи було розроблено лексичний аналізатор для мови програмування KK.