Круть Катерина
ТІ-01
Лабораторна робота №1

Робота № 1
Визначення рекурсивних функцій
Варіант № 12

Мета роботи

Набути досвіду визначення рекурсивних функцій, використання механізму

Завдання

Визначте вказані функції в кожному з завдань:

а) без застосування,

б) із застосуванням вбудованих функцій.

1.12. Вставити у список через кожні n елементів вказане значення, напр. через n=2 значення ’z’: "1234590" ⇒ "12z34z59z0".

2.12. Знайти перше просте число в указаному діапазоні

Круть Катерина
ТІ-01
Лабораторна робота №1

Код програми

```haskell
module Main (main) where

import Data.List
import Data.Numbers.Primes


main :: IO ()
main = do
  putStrLn "hello world"


{-
  Визначте вказані функції в кожному з завдань: а) без застосування, б) з
  застосуванням вбудованих функцій.

  1.12 Вставити у список через кожні n елементів вказане значення, напр.
  через n=2 значення ’z’: "1234590"⇒ "12z34z59z0".

  2.12 Знайти перше просте число в указаному діапазоні.
-}


-- 1.12 ------------------------------------------------------------------------------------------------------------

-- | Фунція для вставлення значення у список через кожні n елементів
-- >> 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]
```

```haskell
-- | 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 елементів
-- >> 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
```

```haskell
-- | 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

```
kkrut@kkrut:~/HaskellProjects/Project1/src$ ghci
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
Prelude> :l Main
[1 of 1] Compiling Main             ( Main.hs, interpreted )
Ok, one module loaded.
*Main> m "abcdef" 3 'X'
"abcXdefX"
*Main> insertN 2 'X' "abcdefg"
"abXcdXefXg"
```

```
kkrut@kkrut:~/HaskellProjects/Project1/src$ ghci Main
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( Main.hs, interpreted )
Ok, one module loaded.
*Main> isPrime 11
True
*Main> isPrime2 11
True
*Main> isPrime 4
False
*Main> getPrime [4..11]
Just 5
*Main> getPrime2 [4..11]
Just 5
*Main> getPrime2 [4, 6]
Nothing
```

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 КК.