**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ**

**Лабораторна робота № 5**

з дисципліни «Непроцедурне Програмування »

Тема роботи: «Рекурсія»

Виконав студент

групи AнД-31

Швед І.Г.

Перевірив:

Гайна Г.А.

**Київ – 2023**

1. Записати N, N+5  та N+22 (N- ваш номер за списком) елементи послідовності Фібоначчі F(n), де 

fibonacci :: Int -> Integer

fibonacci 0 = 0

fibonacci 1 = 1

fibonacci n = fibonacci (n - 1) + fibonacci (n - 2)

1. Записати функцію Min, яка повертає мінімальний елемент зі списку

min' :: Ord a => [a] -> a

min' [] = error "Empty list"

min' [x] = x

min' (x:xs) = min x (min' xs)

1. Записати функцію Elem, яка перевіряє, чи є даний елемент в списку.

elem' :: (Eq a) => a -> [a] -> Bool

elem' \_ [] = False

elem' x (y:ys) = x == y || elem' x ys

1. Функцію, яка сортує список від меншого елементу до більшого.

sortList :: (Ord a) => [a] -> [a]

sortList = sort

1. Факторіал числа

factorial :: Integer -> Integer

factorial 0 = 1

factorial n = n \* factorial (n - 1)

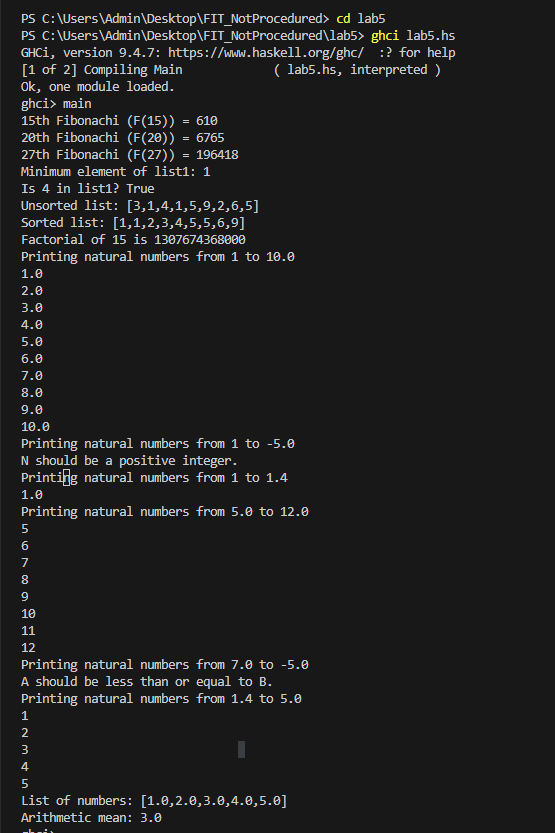
1. Є число N. Треба вивести всі натуральні числа від 1 до N.
2. printNaturalNumbers :: Double -> IO ()
3. printNaturalNumbers n
4. | n < 1     = putStrLn "N should be a positive integer."
5. | otherwise = mapM\_ (print) [1..n]
6. Є 2 натуральні числа А та Б. Треба вивести всі натуральні числа від А до Б.
7. printNaturalNumbersInRange :: Double -> Double -> IO ()
8. printNaturalNumbersInRange a b
9. | a > b     = putStrLn "A should be less than or equal to B."
10. | a < 1     = putStrLn "A should be a positive integer."
11. | b < 1     = putStrLn "B should be a positive integer."
12. | otherwise = mapM\_ (print) [truncate a..truncate b]
13. Функція, яка рахує середнє арифметичне елементів зі списку.

average :: Fractional a => [a] -> a

average [] = 0

average xs = sum xs / fromIntegral (length xs)

**Результат роботи програми:**

****

**Код програми:**

module Main where

import Data.List (sort)

fibonacci :: Int -> Integer

fibonacci 0 = 0

fibonacci 1 = 1

fibonacci n = fibonacci (n - 1) + fibonacci (n - 2)

min' :: Ord a => [a] -> a

min' [] = error "Empty list"

min' [x] = x

min' (x:xs) = min x (min' xs)

elem' :: (Eq a) => a -> [a] -> Bool

elem' \_ [] = False

elem' x (y:ys) = x == y || elem' x ys

sortList :: (Ord a) => [a] -> [a]

sortList = sort

factorial :: Integer -> Integer

factorial 0 = 1

factorial n = n \* factorial (n - 1)

printNaturalNumbers :: Double -> IO ()

printNaturalNumbers n

  | n < 1     = putStrLn "N should be a positive integer."

  | otherwise = mapM\_ (print) [1..n]

printNaturalNumbersInRange :: Double -> Double -> IO ()

printNaturalNumbersInRange a b

  | a > b     = putStrLn "A should be less than or equal to B."

  | a < 1     = putStrLn "A should be a positive integer."

  | b < 1     = putStrLn "B should be a positive integer."

  | otherwise = mapM\_ (print) [truncate a..truncate b]

average :: Fractional a => [a] -> a

average [] = 0

average xs = sum xs / fromIntegral (length xs)

main :: IO ()

main = do

  let n = 15

      nPlus5 = n + 5

      nPlus12 = n + 12

  let fibN = fibonacci n

      fibNPlus5 = fibonacci nPlus5

      fibNPlus12 = fibonacci nPlus12

  putStrLn $ "15th Fibonachi (F(" ++ show n ++ ")) = " ++ show fibN

  putStrLn $ "20th Fibonachi (F(" ++ show nPlus5 ++ ")) = " ++ show fibNPlus5

  putStrLn $ "27th Fibonachi (F(" ++ show nPlus12 ++ ")) = " ++ show fibNPlus12

  let list1 = [3, 1, 4, 1, 5, 9, 2, 6, 5]

  putStrLn $ "Minimum element of list1: " ++ show (min' list1)

  let list2 = [3, 1, 4, 1, 5, 9, 2, 6, 5]

  putStrLn $ "Is 4 in list1? " ++ show (elem' 4 list1)

  let unsortedList = [3, 1, 4, 1, 5, 9, 2, 6, 5]

  putStrLn $ "Unsorted list: " ++ show unsortedList

  let sortedList = sortList unsortedList

  putStrLn $ "Sorted list: " ++ show sortedList

  let n2 = 15

  putStrLn $ "Factorial of " ++ show n ++ " is " ++ show (factorial n2)

  let n3 = 10

      p = 1.4

      k = -5

  putStrLn $ "Printing natural numbers from 1 to " ++ show n3

  printNaturalNumbers n3

  putStrLn $ "Printing natural numbers from 1 to " ++ show k

  printNaturalNumbers k

  putStrLn $ "Printing natural numbers from 1 to " ++ show p

  printNaturalNumbers p

  let a = 5

      b = 12

      a1 = 7

      b1 = -5

      a2 = 1.4

      b2 = 5.0

  putStrLn $ "Printing natural numbers from " ++ show a ++ " to " ++ show b

  printNaturalNumbersInRange a b

  putStrLn $ "Printing natural numbers from " ++ show a1 ++ " to " ++ show b1

  printNaturalNumbersInRange a1 b1

  putStrLn $ "Printing natural numbers from " ++ show a2 ++ " to " ++ show b2

  printNaturalNumbersInRange a2 b2

  let numbers = [1.0, 2.0, 3.0, 4.0, 5.0]

  putStrLn $ "List of numbers: " ++ show numbers

  putStrLn $ "Arithmetic mean: " ++ show (average numbers)

**Висновок:** Протягом виконання лабораторної роботи я навчився працювати з рекурсіями функцій