

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ
ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Звіт до лабораторної роботи №2
на тему
«Списки. Додаткові задачі»

Студента 3 курсу ФКНК
групи ТТП-31
Корнієнка Олександра

Київ-2023

Зміст

<i>Вступ</i>	<i>3</i>
<i>Код програми</i>	<i>3</i>
<i>Тести</i>	<i>4</i>
Тест 1.....	5
Тест 2.....	5
Тест 3.....	5
Тест 4.....	5
Тест 5.....	5
<i>Результати тестів</i>	<i>7</i>

Вступ

Модуль 2

Розділ 2. Списки. Додаткові задачі

Варіант 25

Розбити заданий список на кілька підсписків, записуючи, за можливості, у перший і останній по р 1 елементів, потім, за можливості, у другий і передостанній – по р 2 елементів і т.д. Тут через р і позначено і-е просте число у списку всіх простих чисел.

Код програми

```
printResult :: (Show a) => [a] -> IO ()
printResult array = putStrLn $ "Array: " ++ show array ++ ", Length: " ++ show (length
array)

isNotDivisible :: Int -> Int -> Bool
isNotDivisible x n = n `mod` x /= 0

isPrime :: Int -> Bool
isPrime n
  | n <= 1 = False
  | otherwise = all (`isNotDivisible` n) [2 .. limit]
  where
    limit = floor (sqrt (fromIntegral n))

findNextPrime :: Int -> Int
findNextPrime num
  | isPrime (num + 1) = num + 1
  | otherwise = findNextPrime (num + 1)

subString :: [a] -> Int -> ([a], [a], [a])
subString array divider =
  let firstDivider = divider
```

```

    lastDivider = length array - divider
    (beforeFirst, rest) = splitAt firstDivider array
    (between, afterLast) = splitAt (lastDivider - firstDivider) rest
    in (beforeFirst, between, afterLast)

recursiveSubString :: (Show a) => [a] -> Int -> IO ()
recursiveSubString array divider
  | null array || divider <= 1 = return ()
  | divider * 2 > length array = printResult array
  | otherwise = do
    printResult first
    recursiveSubString between (findNextPrime divider)
    printResult last
  where
    (first, between, last) = subString array divider

main :: IO ()
main = do
  contents <- readFile "input.txt"
  let numbers = map read (words contents) :: [Int]
  putStrLn "input:"
  print numbers
  putStrLn "outcome:"
  if null numbers then putStrLn "[]" else recursiveSubString numbers (findNextPrime 0)

```

Тести

Тести почергово вносяться у файл input.txt, де перший рядок відповідає за масив цілих чисел

Тест 1

Вхід:

Вихід []

Тест 2

Вхід: 1

Вихід: [1]

Тест 3

Вхід: 1 2 3

Вихід: [1,2,3]

Тест 4

Вхід: 1 2 3 4

Вихід: [1,2] [3,4]

Тест 5

Вхід: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
100

Вихід:

[0,1] [2,3,4] [5,6,7,8,9] [10,11,12,13,14,15,16] [17,18,19,20,21,22,23,24,25,26,27]

[28,29,30,31,32,33,34,35,36,37,38,39,40]

[41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59]

[60,61,62,63,64,65,66,67,68,69,70,71,72] [73,74,75,76,77,78,79,80,81,82,83] [84,85,86,87,88,89,90]

[91,92,93,94,95] [96,97,98] [99,100]



Prime numbers

2	3	5	7
11	13	17	19
23	29	31	37
41	43	47	53
59	61	67	71
73	79	83	89
97	101	103	107
109	113		

Результати тестів

```
task2_25.hs x | Haskell
1 printResult :: [Int] -> [Int] -> IO ()
2 printResult array = putStrLn $ "Array: " ++ show array ++ ", Length: " ++ show (length array)
3
4 isNotDivisible :: Int -> Int -> Bool
5 isNotDivisible x = n `mod` x /= 0
6
7 isPrime :: Int -> Bool
8 isPrime n
9   | n <= 1 = False
10  | otherwise = all ('isNotDivisible' x) [2..limit]
11  where
12    limit = floor (sqrt (fromIntegral n))
13
14 findNextPrime :: Int -> Int
15 findNextPrime num
16   | isPrime (num + 1) = num + 1
17   | otherwise = findNextPrime (num + 1)
18
19 substring :: Int -> Int -> [Int] -> [Int]
20 substring array divider =
21   let firstDivider = divider
22       lastDivider = length array - divider
23       (beforeFirst, rest) = splitAt firstDivider array
24       (between, afterLast) = splitAt (lastDivider - firstDivider) rest
25       in (beforeFirst, between, afterLast)
26
27 recursiveSubstring :: [Int] -> [Int] -> Int -> IO ()
28 recursiveSubstring array divider
29   | null array || divider == 1 = return ()
30   | divider > 2 = length array = printResult array
31   | otherwise = do
32     printResult first
33     recursiveSubstring between (findNextPrime divider)
34     printResult last
35   where
36     (first, between, last) = substring array divider
37
38 main :: IO ()
39 main = do
40   contents <- readFile "input.txt"
41   let numbers = map read (words contents) :: [Int]
42       putStrLn "input:"
43       print numbers
44       putStrLn "output:"
45   if null numbers then putStrLn "[]" else recursiveSubstring numbers (findNextPrime 0)
46
```

```
Running: runghc "D:\Users\igora\Desktop\haskell\task2_25.hs"
Input:
[]
Output:
[]
[Done] exited with code=0 in 0.021 seconds

Running: runghc "D:\Users\igora\Desktop\haskell\task2_25.hs"
Input:
[1]
Output:
Array: [1], Length: 1
[Done] exited with code=0 in 0.025 seconds

Running: runghc "D:\Users\igora\Desktop\haskell\task2_25.hs"
Input:
[1,2,3]
Output:
Array: [1,2,3], Length: 3
[Done] exited with code=0 in 0.393 seconds

Running: runghc "D:\Users\igora\Desktop\haskell\task2_25.hs"
Input:
[1,2,3,4]
Output:
Array: [1,2], Length: 2
Array: [3,4], Length: 2
[Done] exited with code=0 in 0.43 seconds

Running: runghc "D:\Users\igora\Desktop\haskell\task2_25.hs"
Input:
[8,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100]
Output:
Array: [1,1], Length: 2
Array: [2,3,4], Length: 3
Array: [5,6,7,8,9], Length: 5
Array: [10,11,12,13,14,15,16], Length: 7
Array: [17,18,19,20,21,22,23,24,25,26,27], Length: 11
Array: [28,29,30,31,32,33,34,35,36,37,38,39,40], Length: 13
Array: [41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59], Length: 19
Array: [60,61,62,63,64,65,66,67,68,69,70,71,72], Length: 13
Array: [73,74,75,76,77,78,79,80,81,82,83], Length: 13
Array: [84,85,86,87,88,89,90], Length: 7
Array: [91,92,93,94,95], Length: 5
Array: [96,97,98], Length: 3
Array: [99,100], Length: 2
[Done] exited with code=0 in 0.954 seconds
```