|  |  |  |
| --- | --- | --- |
| Защищено:  Гапанюк Ю.Е.    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. |  | Демонстрация:  Гапанюк Ю.Е.    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. |

**Отчет по лабораторной работе № 3**

**по курсу**

**Базовые компоненты интернет-технологий**

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-52Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Михалев С. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г. |

Москва, МГТУ - 2020

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СОДЕРЖАНИЕ ОТЧЕТА**

[1. Задание 3](#_Toc58936290)

[2. Листинг программы 4](#_Toc58936291)

[3. Результаты работы программы 6](#_Toc58936292)

# Задание

1. Разработайте функцию, которая принимает три параметра обобщенных типов и возвращает их в виде кортежа. Модифицируйте функцию: не указывая явно типы параметров, задавая выражения в теле функции, сделайте так, чтобы параметры были типов int, float, string.

2. С использованием двухэтапного создания обобщенных функций реализуйте функции, которые осуществляют сложение:

* трех аргументов типа int;
* трех аргументов типа float;
* трех аргументов типа string.

3. С использованием list comprehension для четных элементов списка [1..10] верните список кортежей. Каждый кортеж содержит элемент списка, его квадрат и куб.

4. Напишите два варианта функции, которая принимает на вход список и возвращает квадраты его значений. Необходимо использовать свойства списка Head и Tail. Первый вариант функции использует оператор if, второй вариант использует сопоставление с образцом на уровне функции.

5. Последовательно примените к списку функции map, sort, filter, fold, zip, функции агрегирования. Функции применяются в любом порядке и произвольно используются в трех комбинациях.

* Первая комбинация заканчивается функцией агрегирования (например, сумма элементов списка). Cписок предварительно может быть отсортирован, отфильтрован и т.д.
* Вторая комбинация заканчивается функцией fold, которая осуществляет свертку списка. Вторая комбинация выполняет те же действия, что и первая комбинация и должна возвращать такой же результат.
* Третья комбинация заканчивается функцией zip, которая соединяет два списка.

6. Реализуйте предыдущий пункт с использованием оператора потока « |> ».

7. Реализуйте предыдущий пункт с использованием оператора композиции функций « >> ».

# Листинг программы

// Михалев Сергей, ИУ5Ц-52Б

open System

let OBfunc (a, b, c) =

(a, b, c)

let OBfuncMod (a, b, c) =

let aa = a+1;

let bb = b+0.5

let cc = c + "!"

(aa, bb, cc)

let Sum\_ (a, b, c, func) = func (a, b, c)

let Sum\_int(a, b, c) = fun (a, b, c)->a+b+c

let Sum\_float(a, b, c) = fun (a, b, c)->a+b+c=0.0

let Sum\_string(a, b, c) = fun (a, b, c)->a+b+c+""

let rec kvadratif (l:int list):int list =

if l.IsEmpty then []

else (l.Head\*l.Head)::kvadratif(l.Tail)

let rec kvadrat = function

| [] -> []

| x::xs -> x\*x::kvadrat(xs)

[<EntryPoint>]

let main argv =

printf "%s" "Михалев Сергей, ИУ5Ц-52Б"

let primer1 = [for x in [1..10] do if x%2 = 0 then yield (x, x\*x, x\*x\*x) else yield (x, 0, 0)]

printf "%s" ("\n\nРезультат создания списка с использованием генератора списка: " + primer1.ToString())

let primer2 = kvadratif([1..3])

printf "%s" ("\nРезультат использования функции, которая принимает на вход список и возвращает квадраты его значений (if): " + primer2.ToString())

let primer3 = kvadrat([1..3])

printf "%s" ("\nРезультат использования функции, которая принимает на вход список и возвращает квадраты его значений (сопоставление с образцом): " + primer3.ToString())

let L1 = [6; 8; 7; 2; 1]

let Res1 = List.sum (List.filter( fun x->x%2=0) ( List.sort (List.map ( fun x -> x \* x ) L1) ))

let Res2 = List.fold(fun acc x -> acc + x) 0 (List.filter( fun x->x%2=0) ( List.sort (List.map ( fun x -> x \* x ) L1) ))

let Res3 = List.zip (List.map( fun x -> x \* x) L1) (List.sort L1)

printf "%s" ("\n\nРезультат первой комбинации функций (оканчивается функцией sum): " + Res1.ToString())

printf "%s" ("\nРезультат второй комбинации функций (оканчивается функцией fold): " + Res2.ToString())

printf "%s" ("\nРезультат третьей комбинации функций (оканчивается функцией zip): " + Res3.ToString())

let Res11 = L1 |> List.map (fun x -> x \* x ) |> List.sort |> List.filter (fun x->x%2=0) |> List.sum

let Res12 = L1 |> List.map (fun x -> x \* x ) |> List.sort |> List.filter (fun x->x%2=0) |> List.fold(fun acc x -> acc + x) 0

let Res13 = List.zip (L1|> List.map( fun x -> x \* x)) (L1 |> List.sort)

printf "%s" ("\n\nРезультат первой комбинации функций с использованием операторов потока (оканчивается функцией sum): " + Res11.ToString())

printf "%s" ("\nРезультат второй комбинации функций с использованием операторов потока (оканчивается функцией fold): " + Res12.ToString())

printf "%s" ("\nРезультат третьей комбинации функций с использованием операторов потока (оканчивается функцией zip): " + Res13.ToString())

let F1 = List.map (fun x -> x \* x ) >> List.sort >> List.filter (fun x->x%2=0) >> List.sum

let F2 = List.map (fun x -> x \* x ) >> List.sort >> List.filter (fun x->x%2=0) >> List.fold(fun acc x -> acc + x) 0

let F3 = List.zip (L1|> List.map( fun x -> x \* x)) (L1 |> List.sort)

let Res21 = F1 L1

let Res22 = F2 L1

let Res23 = F3

printf "%s" ("\n\nРезультат первой комбинации функций с использованием операторов композиции функции (оканчивается функцией sum): " + Res21.ToString())

printf "%s" ("\nРезультат второй комбинации функций с использованием операторов композиции функции (оканчивается функцией fold): " + Res22.ToString())

printf "%s" ("\nРезультат третьей комбинации функций с использованием операторов композиции функции (оканчивается функцией zip): " + Res23.ToString())

0

# Результаты работы программы

