|  |
| --- |
| Министерство образования Республики Беларусь  УО «Полоцкий государственный университет им. Ефросинии Полоцкой» |

|  |  |
| --- | --- |
|  | Факультет информационных технологий  Кафедра технологий программирования |

|  |
| --- |
| ОТЧЕТ |
| **к лабораторной работе №7(3)**  **Дисциплина**: «Программирование на платформе .NET» |

|  |  |
| --- | --- |
| Выполнил: | Студент группы 19-ИТ-2  Гилимович А.Н. |

|  |  |
| --- | --- |
| Проверила: | Преподователь  Виноградова А.Д. |

|  |
| --- |
| Полоцк, 2022 г. |

**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. На основе пункта 3.8.1 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.

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

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

**Ход работы**

Изучив теоритический материал, преступил к выполнению заданий.

**Листинг 1** – задание 1

let ex1\_1(a, b, c) = (a, b, c)

let ex1\_2(a, b, c) = (a+a, b+0.0, c+"")

**Листинг 2** – задание 2

let sum\_generic(a, b, c, f) = f(a, b, c)

let ex2\_1(a, b, c) = sum\_generic(a, b, c, fun(a, b, c)->a+b+c)

let ex2\_2(a, b, c) = sum\_generic(a, b, c, fun(a, b, c)->0.0+a+b+c)

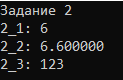
let ex2\_3(a, b, c) = sum\_generic(a, b, c, fun(a, b, c)->""+a+b+c)

printf "Задание 2"

printf "\n2\_1: %i" (ex2\_1(1,2,3));

printf "\n2\_2: %f" (ex2\_2(1.1,2.2,3.3));

printf "\n2\_3: %s" (ex2\_3("1","2","3"));



**Рисунок 1** – Задание 2

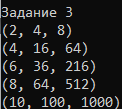
**Листинг 3** – задание 3

let list3 = [for x in [1..10] do if x % 2 = 0 then yield (x, x\*x, x\*x\*x)]

printf "\n\nЗадание 3\n"

for x in list3 do

printfn "%A" x



**Рисунок 2** – Задание 3

**Листинг 4** – задание 4

let list4 = [1..10]

let rec ex4\_1(l:int list):int list =

if l.IsEmpty then []

else (l.Head\*l.Head)::ex4\_1(l.Tail)

let rec ex4\_2 = function

| [] -> []

| x::xs -> x\*x::ex4\_2(xs)

let res4\_1 = ex4\_1(list4)

let res4\_2 = ex4\_2(list4)

printf "\nЗадание 4\n"

printf "ex4\_1: "

for x in res4\_1 do

printf "%A, " x

printf "\nex4\_2: "

for x in res4\_2 do

printf "%A, " x



**Рисунок 3** – Задание 4

**Листинг 5** – задание 5

printf "\n\nЗадание 5\n"

let testList = [1..10]

printfn "testList = %A" testList

let mapList = List.map(fun x->x\*x\*x) testList // Map

printfn "..map.. %A" mapList

let sortList = List.rev (List.sort mapList) // Sort

printfn "..sort.. %A" sortList

let filterList = List.filter(fun x->x % 2 = 0) sortList // Filter

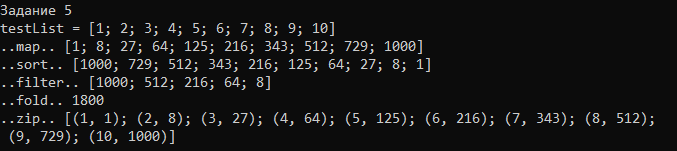
printfn "..filter.. %A" filterList

let foldValue = List.fold(fun acc x -> acc + x) 0 filterList // Fold

printfn "..fold.. %A" foldValue

let zipList = List.zip testList mapList // Zip

printfn "..zip.. %A" zipList



**Рисунок 4** – Задание 5

**Листинг 6** – задание 6

printf "\nЗадание 6\n"

let newList = testList |> List.map(fun x->x\*x\*x) |> List.rev |> List.filter(fun x->x % 2 = 0)

printfn "..map..sort..filter.. %A" newList



**Рисунок 5** – Задание 6

**Листинг 7** – задание 7

let ListFunc = List.map(fun x->x\*x\*x) >> List.rev >> List.filter(fun x->x % 2 = 0)

let newList1 = ListFunc testList

printfn "..map..sort..filter.. %A" newList1



**Рисунок 6** – Задание 7

**Вывод:** изучили основы F# колекций, функции агрегирования и другие функции высшего порядка, выполнили задания лабараторной работы и усвоили материал.