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

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

|  |
| --- |
| ОТЧЕТ |
| **к лабораторной работе №9(5)**  **Дисциплина**: «Программирование на платформе .NET» |

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

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

|  |
| --- |
| Полоцк, 2022 г. |

**Задание**

Для произвольно выбранного типа данных (например, Maybe) реализуйте функции функтора, аппликативного функтора, монады.

Проверьте для Вашей реализации справедливость соответствующих законов для функтора и аппликативного функтора (тех законов, которые можно проверить с использованием F#). Некоторые законы могут не выполняться. Это означает что данный тип не является в полной мере функтором, аппликативным функтором, монадой.

**Ход работы**

Изучив теоритический материал, преступил к выполнению заданий.

**Листинг 1** – Функтор и проверки

let rec fmapList f list =

match list with

| [] -> []

| x::xs -> f x :: fmapList f xs

//проверка 1

//Пусть id – функция, которая возвращает неизменным значение аргумента.

//Тогда подъем этой функции в контекст не влияет на вычисление:

let list1 = [1..10]

let lst1 = fmapList (fun x->x+1) list1

let id x = x

let lst3 = fmapList id list1

//проверка 2

//Для двух функций f и g композиция их подъемов эквивалентна

//подъему композиции.

let func\_f x = x + 1

let func\_g x = x \* 2

let lst2\_1 = fmapList func\_f list1

let lst2\_2 = fmapList func\_g lst2\_1

let lst2\_3 = fmapList (func\_f >> func\_g) list1

**Листинг 2** – Апликаторная функция и проверка

let rec applyList (lf: ('a->'b) list) (list: 'a list) =

[ for f in lf do

for x in list do

yield f x ]

let rec applyList2 (list: 'a list) (lf: ('a->'b) list) =

[ for x in list do

for f in lf do

yield f x]

let al1 = applyList [(fun x-> x+1);(fun x->x\*10)] list1

//Закон 1

//Функция id переданная в списке [id] применяется также как и функцияid вне списка

let at11 = id 1

let at12 = applyList [id] list1

//Закон 2

//Если y=f(x), то подъем функции f и значения х и применение к ним

//функции apply приведет к такому же результату, что и подъем y

let app\_f = fun x -> x+1

let app\_x = 1

let app\_y = [app\_f app\_x]

let app\_y2 = applyList [app\_f] [app\_x]

//Закон 3

//Аргументы apply можно менять местами.

let at31 = applyList [(fun x->x\*10);(fun x-> x\*20)] list1

let at32 = applyList2 list1 [(fun x->x\*10);(fun x-> x\*20)]

**Вывод:** изучили основы F# функторов и апликаторныз функций. Выполнили задания лабараторной работы и усвоили теоретический материал.