

Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №9 по курсу "Функциональное и логическое программирование"

Тема	Использование функционалов и рекурсии	
Студе	ент Пересторонин П.Г.	
Групі	паИУ7-63Б	
Оцені	ка	
Преп	одаватель Толпинская Н. Б.	

Оглавление

1	Зад	дания	2
	1.1	Написать функцию, которая по своему аргументу-списку lst	
		определяет, является ли он полиндромом (то есть равны ли	
		lst и (reverse lst))	2
	1.2	Написать предикат set-equal, который возвращает t, если	
		два его множества-аргумента содержат одни и те же элемен-	
		ты, порядок которых не имеет значения	2
	1.3	Напишите необходимые функции, которые обрабатывают таб-	
		лицу из точечных пар: (страна . столица), и возвращают	
		по стране столицу, а по столице — страну	2
	1.4	Напишите функцию swap-first-last, которая переставляет	
		в списке аргументе первый и последний элементы	3
		1.4.1 Разрушающая структуру	3
		1.4.2 Не разрушающая структуру	3
	1.5	Напишите функцию swap-two-ellement, которая перестав-	
		ляет в списке-аргументе два указанных своими порядковыми	
		номерами элемента в этом списке	4
	1.6	Разрушающая структуру	4
	1.7	Не разрушающая структуру	4
	1.8	Напишите две функции, swap-to-left и swap-to-right, ко-	
		торые производят круговую перестановку в списке-аргументе	
		влево и вправо, соответственно	5
2	Отн	веты на вопросы к лабораторной работе	6
	2.1	Способы определения функций	6
	2.2	Варианты и метолы молификации списков	6

1 Задания

1.1 Написать функцию, которая выбирает из заданного списка только ты числа, которые больше 1 и меньше 10 (Вариант: между 2 заданными границами)

Реализация для случая, когда между 2 заданными границами

1.1.1 С помощью функционалов

```
(defun list-between-only (from to lst)
(reverse (reduce #'(lambda (acc el)
(cond ((and (< el to) (> el from)) (cons el acc))
(T acc)))
(cons nil lst))))
```

1.1.2 Рекурсия

```
(defun list-btwn-only-internal (from to lst acc)
(cond ((null lst) acc)
(T (let ((head (car lst)))
(cond ((and (< head to) (> head from))
(list-btwn-only-internal from to (cdr lst) (cons head acc)))
(T (list-btwn-only-internal from to (cdr lst) acc))))))
(defun list-btwn-only-rec (from to lst)
(reverse (list-btwn-only-internal from to lst nil)))
```

1.2 Написать функцию, вычисляющую декартово произведение двух своих списковаргументов

1.2.1 С помощью функционалов

```
(defun list-mul (lst1 lst2)
(reduce #'(lambda (acc outer-el)
(append acc (mapcar #'(lambda (inner-el)
(cons outer-el inner-el))
1st2)))
(cons nil lst1)))
```

1.2.2 Рекурсия

```
(defun list-mul-internal (cur-lst1 src-lst1 lst2 acc)
(cond ((null lst2) acc)
((null cur-lst1) (list-mul-internal src-lst1 src-lst1 (cdr lst2) acc))
(T (list-mul-internal (cdr cur-lst1) src-lst1 lst2 (cons (cons (car cur-lst1) (car lst2)) acc)))))
(defun list-mul-rec (lst1 lst2)
(and lst1 lst2 (reverse (list-mul-internal lst1 lst1 lst2 nil))))
```

1.3 Почему так реализовано reduce, в чем причина?

```
(reduce '+ ()) -> 0
```

Поведение в данном примере обусловлено работой функции +. Эта функция — функционал, который при 0 количестве аргументов возвращает значение 0. Если подать на вход reduce функцию, которая не может обработать 0 аргументов (например, математическая функция cons), то вызов reduce с пустым списком в качестве второго аргумента вернет ошибку

(invalid number of arguments: 0). При этом, если подано более одного аргумента, то reduce выполняет следующие действия:

- 1. сохраняет первый элемент списка в область памяти (для определенности назовем ее acc);
- 2. для всех остальных элементов списка выполняет переданную в качестве первого аргумента функцию, подавая на вход 2 аргумента (асс и очередной элемент списка) и сохраняя результат в асс.

Пример упрощенной реализации **reduce** (в данной реализации опущены проверки аргументов):

```
(defun my-reduce-internal (func lst acc)
(cond ((null lst) acc)
(T (my-reduce-internal func (cdr lst) (funcall func acc (car lst)))))
(defun my-reduce (func lst)
(cond ((null lst) (funcall func))
(T (my-reduce-internal func (cdr lst) (car lst)))))
```

1.4 Пусть list-of-lists — список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов list-of-lists, то есть, например, для аргумента ((1 2) (3 4)) -> 4

1.4.1 С помощью функционалов

```
(defun sum-lens (list-of-lists)
(reduce #'(lambda (acc lst) (+ acc (length lst)))
(cons 0 list-of-lists)))
```

1.4.2 Рекурсия

```
(defun sum-lens-rec-internal (lol acc)
(cond ((null lol) acc)
(T (sum-lens-rec-internal (cdr lol) (+ acc (length (car lol))))))
(defun sum-lens-rec (list-of-lists)
(and (listp list-of-lists) (sum-lens-rec-internal list-of-lists 0)))
```

1.5 Используя рекурсию, написать функцию, которая по исходному списку стоит список квадратов чисел смешанного структурированного списка

```
(defun square (x) (* x x))
(defun square-list-internal (lst acc)
(cond ((null lst) acc)
(T (square-list-internal (cdr lst) (cons (square (car lst)) acc)))))
(defun square-list (lst)
(reverse (square-list-internal lst nil)))
```

- 2 Ответы на вопросы к лабораторной работе
 - 2.1 Классификация рекурсивных функций