**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 №5 по курсу "Функциональное и логическое программирование"

**Тема** Использование функционалов

**Студент** Гурова Н.А.

**Группа** ИУ7-64Б

**Оценка (баллы)**

**Преподаватель** Толпинская Н.Б., Строганов Ю.В.

Москва — 2023 г.

# 1 Теоретические вопросы

**Функционал (функция высшего порядка)** — функция, аргументом или результатом которой является другая функция.

## 1.1 APPLY

Функционал apply является обычной функцией с двумя вычисляемыми аргументами, обращение к ней имеет вид

```
1    (apply f l)          ;   f — функциональный аргумент
2                         ;   l — список фактических параметров
```

Значение функционала — результат применения f к этим фактическим параметрам. Примеры:

```
1    (apply (lambda (x y) (* x y)) '(9 8))        ; 72
2
3    (defun f1 (x y) (* x y))
4    (apply 'f1 '(9 8))                           ; 72
```

## 1.2 FUNCALL

Функционал funcall — особая функция с вычисляемыми аргументами, обращение к ней

```
1    (funcall f e1 .. en)          ; f — функциональный аргумент
2                                  ; n >= 0
```

Действие аналогично apply, отличие в том, что аргументы передаются не в виде списка, а по отдельности. Примеры:

```
1    (funcall (lambda (x y) (* x y)) 9 8)         ; 72
2
3    (defun f1 (x y) (* x y))
4    (funcall 'f1 9 8)                            ; 72
```

## 1.3 MAPCAR

Значение функции mapcar вычисляется путем применения функции fn к последовательным элементам xi списка, являющегося вторым аргументом mapcar.

```
1    (mapcar f (x1 x2 ... xn))     ; f — функциональный аргумент
2                                  ; n >= 0
3
4    (mapcar f (x1 y1 z1...) (x2 y2 z2...) (xn yn zn...)))
```

В качестве значения функционала возвращается список, построенный из результатов вызовов функционального аргумента mapcar.

```
1    (mapcar (lambda (x) (list x)) '(a b c))     ; ((A) (B) (C))
2    (mapcar 'f2 '(1 2 3))                       ; (3 6 9)
3    (mapcar 'f1 '(1 2) '(3 4) )                 ; (3 8)          !!!
```

## 1.4 MAPLIST

Функционал maplist принимает два аргумента. Значение первого аргумента должно быть функцией списком. Второй аргумент - список. Функция, задаваемая первым аргументом, должна принимать на вход список. Выполнение функционала заключается в том, что функция, заданная первым аргументом, последовательно применяется к значению второго аргумента, к этому списку без первого элемента, без первых двух элементов и т.д. до исчерпания списка. Результаты вызова функции объединяются в список, который функционал вернет в качестве значения.

Примеры:

```
1    (maplist (lambda (x) x) '(a b c))       ; ((A B C) (B C) (C))
2    (maplist 'reverse '(a b c))             ; ((C B A) (C B) (C))
```

## 2 Практические задания

### 2.1 Напишите функцию, которая уменьшает на 10 все числа из списка-аргумента этой функции, проходя по верхнему уровню списковых ячеек

```
(
defun reduce_it_by_10 (lst)
    (mapcar (lambda (x) (if (numberp x) (- x 10) x)) lst)
)

(print (reduce_it_by_10 '()))                    ; NIL
(print (reduce_it_by_10 '(1 2 3 4 5)))           ; (-9 -8 -7 -6 -5)
(print (reduce_it_by_10 '(1 (2) 3 "4" 5)))       ; (-9 (2) -7 "4"
   -5)
```

### 2.2 Написать функцию которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке

```
(
defun squares_of_numbers (lst)
    (mapcar (lambda (x) (* x x)) lst)
)

(print (squares_of_numbers '()))                 ; NIL
(print (squares_of_numbers '(1 2 3 4 5)))        ; (1 4 9 16 25)
(print (squares_of_numbers '(-1 -2 -3)))         ; (1 4 9)
```

### 2.3 Напишите функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда все элементы списка — числа, элементы списка – любые объекты

```
(
defun multiply_all_numbers (lst y)
    (mapcar (lambda (x) (if (numberp x) (* x y) x)) lst)
)

(multiply_all_numbers '() 1)                    ; NIL
(multiply_all_numbers '(1 2 3 4 5) 5)           ; (5 10 15 20 25)
(multiply_all_numbers '(-1 -2 -3) 5)            ; (-5 -10 -15)
(multiply_all_numbers '("1" -2 (-3)) -2)        ; ("1" 4 (-3))
```

## 2.4 Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом (то есть равны ли lst и (reverse lst)), для одноуровнего смешанного списка

```
; Идея функции — получить список из 1 и 0, проверяя
; равны ли элементы списка и перевернутого списка
; Потом перемножить эти значения — получили 1 значит палиндром,
; иначе нет

(
    defun is_palindrome_content (lst)
        (apply
            '*
            (mapcar
                (lambda (x y) (if (equal x y) 1 0))
                lst
                (reverse lst)
            )
        )
)

(
    defun is_palindrome (lst)
        (if
            (= (is_palindrome_content lst) 1)
```

```
22                "Да, это палиндром"
23                "Нет, не палиндром"
24            )
25    )
26
27    (print (is_palindrome '()))                    ; Да, это палиндром
28    (print (is_palindrome '(1 2 3 4 5)))           ; Нет, не палиндром
29    (print (is_palindrome '(-1 -2 -3)))            ; Нет, не палиндром
30    (print (is_palindrome '(1 2 1 2 1 1)))         ; Нет, не палиндром
31    (print (is_palindrome '(1 2 2 3 2 2 1)))       ; Да, это палиндром
32    (print (is_palindrome '(1 (2 1) (2 1) 1)))     ; Да, это палиндром
```

## 2.5 Используя функционалы, написать предикат set-equal, который возвращает t, если два его множества-аргумента (одноуровневые списки) содержат одни и те же элементы, порядок которых не имеет значения

```
1     (
2         defun is_first_equal_second (lst1 lst2)
3             (apply
4                 '*
5                 (mapcar
6                     (
7                         lambda (x)
8                         (apply
9                             '+
10                            (mapcar
11                                (lambda (y) (if (equal x y) 1 0))
12                                lst2
13                            )
14                        )
15                    )
16                    lst1
17                )
18            )
19    )
```

```
20
21      (
22          defun set_equal (lst1 lst2)
23              (if
24                  (= 1
25                     (is_first_equal_second lst1 lst2)
26                     (is_first_equal_second lst2 lst1)
27                  )
28                  "Эквивалентны"
29                  "Не эквивалентны"
30              )
31      )
32
33
34      (print (set_equal '(1 2 3) '(3 2 1)))              ; Эквивалентны
35      (print (set_equal '(1 2 0 4 5) '(1 2 3 4 5 0)))    ; Не эквивалентны
36      (print (set_equal '(1 3 2 0 4 5) '(1 2 3 4 5 0))) ; Эквивалентны
```

## 2.6 Напишите функцию, select-between, которая из списка-аргумента, содержащего только числа, выбирает только те, которые расположены между двумя указанными числами — границами-аргументами и возвращает их в виде списка

```
1       ; Обрезать слева, перевернуть, снова обрезать слева и перевернуть
2       (
3       defun select_between (lst start end)
4           (reverse
5               (apply
6                   'append
7                   (maplist
8                       (lambda (part_lst)
9                           (if
10                              (= (car part_lst) end)
11                              part_lst
12                          )
13                      )
```

```
                    (reverse
                        (apply
                            'append
                            (maplist
                                (lambda (part_lst)
                                    (if
                                        (= (car part_lst) start)
                                        part_lst
                                    )
                                )
                                lst
                            )
                        )
                    )
                )
            )
        )
    )


    (print (select_between '(1 2 3 4) 1 3))             ; (1 2 3)
    (print (select_between '(1 2 3 4) 2 3))             ; (2 3)
    (print (select_between '(1 2 3 4) 2 4))             ; (2 3 4)
    (sort  (select_between '(5 2 1 10 3 4 6) 2 4) #'<)  ; (1 2 3 4 10)
```

## 2.7 Написать функцию, вычисляющую декартово произведение двух своих списков-аргументов (A x B это множество всевозможных пар (a b), где a принадлежит A, принадлежит B)

```
1   (
2   defun cartesian_product (lst1 lst2)
3       (apply
4           'append
5           (mapcar
6               (lambda (a) (mapcar (lambda (b) (list a b)) lst2))
7               lst1
```

```
            )
        )
    )

    (cartesian_product '() '())          ; ()
    (cartesian_product '(1) '(3))        ; ((1 3))
    (cartesian_product '(1 2) '(3 4))    ; ((1 3) (1 4) (2 3) (2 4))
    (cartesian_product '(1 2) '(3))      ; ((1 3) (2 3))
    (cartesian_product '(2) '(3 4))      ; ((2 3) (2 4))
    (cartesian_product '() '(3 4))       ; ()
```