



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»

## Отчёт по лабораторной работе №5 по курсу

### «Функциональное и логическое программирование»

Тема Использование управляющих структур, работа со списками.

Студент Сироткина П.Ю.

Группа ИУ7-66Б

Преподаватели Толпинская Н.Б., Строганов Ю.В.

Москва — 2022 г.

# Практические задания

**1. Написать функцию, которая по своему списку-аргументу `lst` определяет, является ли он палиндромом (то есть равны ли `lst` и `reverse(lst)`).**

```
1 (defun palindrome(lst) (equal lst (reverse lst)))
```

**2. Написать предикат `set-equal`, который возвращает `T`, если два его множества-аргумента содержат одни и те же элементы, порядок которых не имеет значения.**

```
1 (defun set_equal(set1 set2)
2   (and (= (length set1) (length set2))
3         (every #'(lambda (elem) (member elem set1 :test #'equal)) set2)
4         (every #'(lambda (elem) (member elem set2 :test #'equal)) set1)))
```

**3. Напишите свои необходимые функции, которые обрабатывают таблицу из 4-х точечных пар: (страна.столица), и возвращает по стране столицу, а по столице - страну.**

```
1 (defun get_country(table capital)
2   (cond ((null table) Nil)
3         ((equal (cadr table) capital) (cadr table))
4         (T (get_country (cddr table) capital))))
5
6 (defun get_capital(table country)
7   (cond ((null table) Nil)
8         ((equal (cadr table) country) (cadr table))
9         (T (get_capital (cddr table) country))))
10
11 (defun get_country(table capital)
12   (car (rassoc capital table)))
13
14 (defun get_capital(table country)
15   (cdr (assoc country table)))
```

4. Напишите функцию `swap-first-last`, которая переставляет в списке-аргументе первый и последний элементы.

```
1 (defun swap_first_last(lst)
2   (let ((first_elem (car lst)))
3     (setf (car lst) (car (last lst)))
4     (setf (car (last lst)) first_elem)
5     lst))
```

5. Напишите функцию `swap-two-element`, которая переставляет в списке-аргументе два указанных своими порядковыми номерами элемента в этом списке.

```
1 (defun swap_two_elements(lst i j)
2   (if (and (< -1 i (length lst)) (< -1 j (length lst)))
3       (let ((i_elem (nth i lst)) (j_elem (nth j lst)))
4         (setf (nth i lst) j_elem) (setf (nth j lst) i_elem) lst)
5       lst))
```

6. Напишите две функции, `swap-to-left` и `swap-to-right`, которые производят одну круговую перестановку в списке-аргументе влево и вправо соответственно.

```
1 (defun swap_to_left(lst)
2   (cond ((null lst) Nil)
3         (T (append (cdr lst) (cons (car lst) Nil)))))
4
5 (defun swap_to_right(lst)
6   (cond ((null lst) Nil)
7         (T (cons (car (reverse lst)) (reverse (cdr (reverse lst)))))))
```

7. Напишите функцию, которая добавляет к множеству двухэлементных списков новый двухэлементный список, если его там нет.

```
1 (defun add_pair_to_set(src_set pair)
2   (if (not (member pair src_set :test #'equal))
3       (append src_set (list pair))
4       src_set))
```

**8. Напишите функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка-аргумента, когда**

- Все элементы списка – числа;
- Элементы списка – любые объекта.

```
1 (defun mul_num(lst multiplier)
2   (mapcar #'(lambda(elem) (* elem multiplier)) lst))
3
4 (defun mul(lst multiplier)
5   (mapcar #'(lambda (elem)
6             (cond ((listp elem) (mul (cdr elem) multiplier))
7                   ((numberp elem) (* elem multiplier))
8                   (T elem))) lst))
```

**9. Напишите функцию select-between, которая из списка-аргумента из 5 чисел выбирает только те, которые расположены между двумя указанными границами-аргументами и возвращает их в виде списка (упорядоченного по возрастанию списка чисел (+2 балла)).**

```
1 (defun select_between(lst a b)
2   (cond ((null lst) Nil)
3         ((< a (car lst) b)(cons (car lst)(select_between (cdr lst) a b)))
4         (T (select_between (cdr lst) a b))))
```