



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

Название: Использование управляющих структур, работа со списками.

Дисциплина: Функциональное и логическое программирование

Студент ИУ7-64Б
(Группа)

(Подпись, дата)

Л.Е.Тартыков
(И.О. Фамилия)

Преподаватель

(Подпись, дата)

Н.Б.Толпинская
(И.О. Фамилия)

Преподаватель

(Подпись, дата)

Ю.В.Строганов
(И.О. Фамилия)

Москва, 2022

1 Теоретические вопросы

1.1 Структуроразрушающие и не разрушающие структуру списка функции.

Не разрушающие структуру списка функции – функции, которые не изменяют исходный список. Примеры таких функций:

1. **append** – имеет переменное число параметров; выполняет объединение списков в один; возвращает список в качестве значения функции; выполняет копирование всех элементов списка, кроме последнего.
2. **remove** – имеет два аргумента; удаляет все вхождения элемента *el* из списка *lst* (формула (1.1)).

$$(remove\ el\ lst) \quad (1.1)$$

3. **reverse** – имеет один аргумент и меняет порядок элементов аргумента на противоположный.
4. **substitute** – имеет три аргумента; заменяет все элементы списка, которые равны второму аргументу на значение первого.
5. **member** – имеет два аргумента; возвращает хвост списка, начиная со списковой ячейки, удовлетворяющей первому аргументу; иначе возвращает *nil*
6. **nthcdr** – имеет два аргумента (номер элемента (*n*) и список) ; возвращает хвост списка, начиная с *n*-ого элемента.
7. **nth** – имеет два аргумента (номер элемента (*n*), список); возвращает *car*-указатель на *n*-ый элемент списка.
8. **length** – имеет один аргумент (последовательность – строки или список); возвращает число элементов второго аргумента.

Структуроразрушающие функции – функции, которые изменяют указатели списка. Таким образом, не происходит копирования элементов списка. Такие функции обычно начинаются с символа *n* (*nreverse*, *nconc*, ...). Примеры таких функций:

1. **nconc** – аналогично функции *append*, но копирования элементов не происходит.
2. **delete** – аналогично функции *remove*; не создает копии исходного списка.
3. **nreverse** – аналогично функции *reverse*; не создает копии исходного списка.
4. **nsubstitute** – аналогично функции *substitute*; не создает копии исходного списка.

1.2 Отличие в работе функций *cons*, *list*, *append*, *nconc* и в их результате.

cons – имеет два аргумента и возвращает бинарный узел. Если вторым аргументом является атом, то возвращается точечная пара; если список – список. Создает в памяти списочную ячейку для аргумента.

list – имеет произвольное число аргументов и возвращает список. Создаются столько списочных ячеек, сколько переданных аргументов.

append – имеет произвольное число аргументов; важным свойством является то, что создается копия всех аргументов (новые списочные ячейки), кроме последнего, и дальнейшая работа ведется с ней; при этом сохраняется возможность работать с исходным списком. В результате функции возвращается список.

nconc – функция, аналогичная *append* за исключением того, что исходный список не копируется (структуроразрушающая функция). Создается столько списочных ячеек, сколько переданных аргументов.

2 Практические задания

2.1 Написать функцию, которая по своему списку-аргументу `lst` определяет, является ли он палиндромом.

Листинг 2.1 – Задание 1

```
1 (defun is_palindrom_1 (lst)
2   (equal lst (reverse lst)))
3
4 (defun is_palindrom_2 (lst)
5   (is_equal_2 lst (my_reverse lst))
6 )
7
8 (defun is_equal_1 (lst reverse_lst)
9   (cond ((null lst) T)
10         ((and (numberp (car lst)) (numberp (car reverse_lst)))
11            (is_equal_numbers lst reverse_lst))
12         ((and (listp (car lst)) (listp (car reverse_lst)))
13            (is_equal_1 (car lst) (car reverse_lst)))
14   )
15 )
16
17 (defun is_equal_2 (lst reverse_lst)
18   (cond ((null lst) T)
19         ((= (car lst) (car reverse_lst)) (is_equal_2 (cdr lst) (cdr
20               reverse_lst)))
21   )
22 )
23
24 (defun move_to (lst result)
25   (cond ((null lst) result)
26         (T (move_to (cdr lst) (cons (car lst) result))))
27 )
28
29 (defun my_reverse (lst)
30   (move_to lst ()))
31
32 (defun is_equal_numbers (lst reverse_lst)
33   (if (= (car lst) (car reverse_lst))
34       (is_equal (cdr lst) (cdr reverse_lst)) nil)
35 )
```

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

Листинг 2.2 – Задание 2

```
1 (defun is_elem_in_set (elem set)
2   (cond ((null set) nil)
3         ((= elem (car set)) T)
4         (T (is_elem_in_set elem (cdr set))))
5   )
6 )
7
8 (defun is_equal_set (set_1 set_2)
9   (cond ((null set_1)
10         ((is_elem_in_set (car set_1) set_2)
11          (is_equal_set (cdr set_1) set_2)
12          )
13         (T nil)
14   )
15 )
16
17 (defun is_set (set)
18   (cond ((null set) T)
19         ((eql (is_double_in_set set) 0) (is_set (cdr set)))
20   )
21 )
22
23 (defun is_double_in_set (set)
24   (if (member (car set) (cdr set)) 1 0)
25 )
26
27 (defun set_equal (set_1 set_2)
28   (cond ((not (is_set set_1)) nil)
29         ((not (is_set set_2)) nil)
30         ((not (= (length set_1) (length set_2))) nil)
31         ((is_equal_set set_1 set_2) T)
32   )
33 )
```

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

Листинг 2.3 – Задание 3

```
1 (defvar table)
2 (setq table '((Russia . Moscow) (Ireland . Dublin) (Japan . Tokyo)
3               (China . Pekin)))
4
5 (defun get_country (capital)
6   (let ((country 0))
7     (cond ((mapcar #'(lambda (x) (cond ((eq (cdr x) capital)
8                                         (setq country (car x))))
9                                         table)
10          country)
11   )
12 )
13 )
14
15 (defun get_capital (country)
16   (let ((capital 0))
17     (cond ((mapcar #'(lambda (x) (cond ((eq (car x) country)
18                                         (setq capital (cdr x))))
19                                         table)
20          capital)
21   )
22 )
23 )
```

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

Листинг 2.4 – Задание 4

```
1 (defun swap_first_last (lst)
2   (cons (car (last lst))
3         (nconc (cdr (nreverse (cdr (nreverse lst)))) (cons (car lst) nil)))
4 )
```

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

Листинг 2.5 – Задание 5

```
1 (defvar first_elem)
2 (defvar second_elem)
3
4 (defun append_elem (lst index index_1 index_2)
5   (cond ((null lst) nil)
6         ((= index index_1)
7          (cons second_elem (append_elem (cdr lst) (+ index 1) index_1 index_2)))
8         ((= index index_2)
9          (cons first_elem (append_elem (cdr lst) (+ index 1) index_1 index_2)))
10        (T (cons (car lst)
11                  (append_elem (cdr lst) (+ index 1) index_1 index_2))))
12 )
13 )
14
15 (defun swap_two_elements (lst index_1 index_2)
16   (setq first_elem (nth index_1 lst))
17   (setq second_elem (nth index_2 lst))
18   (let ((len_lst (length lst))
19         (index 0))
20     (cond ((or (< index_1 0) (> index_1 len_lst)) "Некорректный размер")
21           ((or (< index_2 0) (> index_2 len_lst)) "Некорректный размер")
22           ((>= index_1 index_2) "Некорректный размер")
23           (T (append_elem lst 0 index_1 index_2)))
24   )
25 )
26 )
```

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

Листинг 2.6 – Задание 6

```
1 (defun without_last (lst)
2   (nreverse (cdr (nreverse lst))))
3 )
4
5 (defun swap_to_right (lst)
6   (cons (car (last lst)) (without_last lst)))
7 )
8
9 (defun swap_to_left (lst)
10  (nconc (cdr lst) (cons (car lst) nil)))
11 )
```

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

Листинг 2.7 – Задание 7

```
1 (defun check_lists_in_set (set_lists)
2   (cond ((null set_lists) T)
3         ((not (listp (car set_lists))) nil)
4         ((if (= 2 (length (car set_lists)))
5              (check_lists_in_set (cdr set_lists)) nil))
6   )
7 )
8
9 (defun is_in_set (new_list set_lists)
10  (cond ((null set_lists) T)
11        ((not (equal new_list (car set_lists)))
12         (is_in_set new_list (cdr set_lists)))
13  )
14 )
```


Листинг 2.8 – Задание 7 (продолжение)

```
1 (defun append_new_elem (new_list set_lists)
2   (cond ((not (listp new_list)) "Ошибка")
3         ((not (listp set_lists)) "Ошибка")
4         ((not (check_lists_in_set set_lists)) "Ошибка")
5         ((is_in_set new_list set_lists) (nconc set_lists '(,new_list)))
6         (T set_lists))
7   )
8 )
```

2.8 Напишите функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка- аргумента, когда а) все элементы списка — числа, б) элементы списка — любые объекты.

Листинг 2.9 – Задание 8

```
1 (defun multiply_var_1 (lst number)
2   (cond ((null lst) nil)
3         (T (cons (* number (car lst)) (cdr lst))))
4 )
5
6 (defun multiply_var_2 (lst number)
7   (cond ((null lst) nil)
8         (T (if (numberp (car lst))
9               (cons (* number (car lst)) (cdr lst))
10              (cons (car lst) (multiply_var_2 (cdr lst) number))))
11   )
12 )
13 )
14
15 (defun check_lst (lst number)
16   (cond ((not (numberp number)) nil)
17         ((not (listp lst)) nil)
18         ((not (= 3 (length lst))) nil)
19         (T T))
20 )
21 )
22
23 (defun multiply_list_by_number_1 (lst number)
24   (cond ((check_lst lst number) (multiply_var_1 lst number)))
25 )
```

Листинг 2.10 – Задание 8 (продолжение)

```
1 (defun multiply_list_by_number_2 (lst number)
2   (if (check_lst lst number) (multiply_var_2 lst number) "Ошибка")
3 )
```

2.9 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).

Листинг 2.11 – Задание 9

```
1 (defun select_between_1 (lst board_left board_right)
2   (let ((result_lst nil))
3     (setq result_lst (if (check_input_data lst board_left board_right)
4                          (find_left_board lst board_left board_right 0)
5                          nil)
6     )
7   )
8   (if (not (null result_lst)) (bubble_sort_asc result_lst) "Ошибка")
9   )
10 )
11
12 (defun check_input_data (lst board_left board_right)
13   (cond ((not (= (length lst) 5)) nil)
14         ((< board_left 0) nil)
15         ((>= board_right 5) nil)
16         ((<= board_right board_left) nil)
17         (T T)
18   )
19 )
20
21 (defun find_left_board (lst board_left board_right index_board)
22   (if (= index_board board_left)
23     (find_right_board (cdr lst) board_right (+ index_board 1))
24     (find_left_board (cdr lst) board_left board_right (+ index_board 1))
25   )
26 )
```

Листинг 2.12 – Задание 9 (продолжение)

```
1 (defun find_right_board (lst board_right index_board)
2   (if (= index_board board_right)
3       (cons (car lst) nil)
4       (cons (car lst) (find_right_board (cdr lst) board_right (+ index_board
5                                         1))))
6 )
7
8 (defun bubble (lst)
9   (cond ((atom (cdr lst)) lst)
10         ((> (car lst) (cadr lst))
11          (cons (cadr lst) (bubble (cons (car lst) (cddr lst)) ))
12          )
13         (T lst)
14   )
15 )
16
17 (defun bubble_sort_asc (lst)
18   (cond ((atom (cdr lst)) lst)
19         (T (bubble (cons (car lst) (bubble_sort_asc (cdr lst)))))
20   )
21 )
22
23 (defun select_between_2 (lst board_left board_right)
24   (bubble_sort_asc (mapcan #'(lambda (x)
25                                (and (>= x board_left) (<= x board_right) (list x)))
26                                lst))
27 )
```