



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 дисциплине "Функциональное и логическое программирование"

Тема Использование управляющих структур, работа со списками

Студент Малышев И. А.

Группа ИУ7-61Б

Оценка (баллы) _____

Преподаватель: Толпинская Н. Б.

Москва — 2022 г.

Теоретические вопросы

1. Структуроразрушающие и не разрушающие структуру списка функции

Не разрушающие структуру списка функции. Данные функции не меняют сам объект-аргумент, а создают копию. К таким функциям относятся: `append`, `reverse`, `last`, `nth`, `nthcdr`, `length`, `remove`, `subst` и прочие.

Структуроразрушающие функции. Данные функции меняют сам объект-аргумент, невозможно вернуться к исходному списку. Чаще всего такие функции начинаются с префикса `n-`. К таким функциям относятся: `nreverse`, `nconc`, `nsubst` и прочие.

2. Отличие в работе функций `cons`, `list`, `append`, `nconc` и в их результате

Функция `cons` — чисто математическая, конструирует списковую ячейку, которая может вовсе и не быть списком. Является списком только в том случае, если вторым аргументом передан список.

Функция `list` — форма, принимает произвольное количество аргументов и конструирует из них список. Результат — всегда список. При нуле аргументов возвращает пустой список.

Функция `append` — форма, принимает на вход произвольное количество аргументов и для всех аргументов, кроме последнего, создает копию, ссылая при этом последний элемент каждого списка-аргумента на первый элемент следующего по порядку списка-аргумента. Копирование для последнего не делается в целях эффективности.

Практические задания

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

Листинг 1: Решение задания №1

```
1 (defun is-palindrome (lst)
2   (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.

Листинг 2: Решение задания №2

```
1 (defun set-equal (set1 set2)
2   (if (= (length set1) (length set2))
3       (and (subsetp set1 set2) (subsetp set2 set1))
4       Nil))
```

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 3: Решение задания №3

```
1 (defun get-capital (table country)
2   (cdr (assoc country table)))
3
4 (defun get-country (table capital)
5   (car (rassoc capital table)))
```

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

Листинг 4: Решение задания №4

```

1 (defun swap-first-last (lst)
2   (let ((el1 (car lst)) (last-el (car (reverse lst))))
3     (reverse (cons el1 (cdr (reverse (cons last-el (cdr lst))))))))

```

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

Листинг 5: Решение задания №5

```

1 (defun swap-two-ellement (n1 n2 lst)
2   (let ((len (length lst)) (lst-copy (copy-list lst)))
3     (and (< n1 len) (< n2 len)
4       (let ((el1 (nth n1 lst)) (el2 (nth n2 lst)))
5         (setf (nth n1 lst-copy) el2)
6         (setf (nth n2 lst-copy) el1)
7         lst-copy))))

```

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

Листинг 6: Решение задания №6

```

1 (defun swap-to-left (lst)
2   (append (cdr lst) (cons (car lst) Nil)))
3
4 (defun swap-to-right (lst)
5   (cons
6     (car (reverse lst))
7     (reverse (cdr (reverse lst)))))

```

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

Листинг 7: Решение задания №7

```
1 (defun is-double-in-set (lst double)
2   (cond ((null lst) nil)
3         ((and (eql (caar lst) (car double)) (eql (cadr lst) (cadr
4           double)))) t)
5         (t (is-double-in-set (cdr lst) double))))
6
7 (defun inner-append-double-lst (lst double)
8   (cond ((null (cdr lst)) (cons (car lst) (cons double nil)))
9         (t (cons (car lst) (inner-append-double-lst (cdr lst) double)))))
10
11 (defun append-double-lst (lst double)
12   (if (is-double-in-set lst double)
13       lst
14       (inner-append-double-lst lst double)))
```

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 8: Решение задания №8

```
1 ; a)
2 (defun first-num-mul (lst mul)
3   (cond ((null lst) nil)
4         (t (cons (* (car lst) mul) (cdr lst)))))
5
6 ; b)
7 (defun first-num-mul (lst mul)
8   (cond ((null lst) nil)
9         ((numberp (car lst)) (cons (* (car lst) mul) (cdr lst)))
10        ((listp (car lst)) (cons (first-num-mul (car lst) mul)
11                                  (first-num-mul (cdr lst) mul)))
12        (t (cons (car lst) (first-num-mul (cdr lst) mul)))))
```

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)).

Листинг 9: Решение задания №9

```
1 (defun betweenp (e1 b1 b2)
2   (< (* (- e1 b1) (- e1 b2)) 0))
3
4 (defun select-between (lst b1 b2)
5   (cond ((null lst) nil)
6         ((betweenp (car lst) b1 b2) (cons (car lst) (select-between (cdr
7   (T (select-between (cdr lst) b1 b2))))
```