|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

*к лабораторной работе №7*

*По курсу: «Функциональное и логическое программирование»*

**Тема: «**Использование управляющих структур, модификация списков».

Студент: Якуба Д.В.

Группа: ИУ7-63Б

Преподаватели: Толпинская Н. Б.,

Строганов Ю. В.

Москва, 2021 г.

# Практическая часть

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

Решение:

(*defun* isPal (lst)

    (equal lst (reverse lst)))

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

Решение:

(*defun* set-equal (stF stS)

    (equal (sort stF #'>) (sort stS #'>)))

Задание 3. Напишите необходимые функции, которые обрабатывают таблицу из точечных пар: (страна.столица), и возвращают по стране – столицу, а по столице – страну.

Решение:

(*defun* find-country (table capital)

    (*cond* ((< (length table) 1) '(Такой столицы нет в таблице))

          ((eq (cdar table) capital) (caar table))

          (t (find-country (cdr table) capital))))

(*defun* find-capital (table country)

    (*cond* ((< (length table) 1) '(Такой страны нет в таблице))

          ((eq (caar table) country) (cdar table))

          (t (find-capital (cdr table) country))))

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

Решение:

(*defun* swap-first-last (lst)

    (*and*

        (*setf* (first lst) (cons (first lst) (car (last lst))))

        (*setf* (car (last lst)) (car (first lst)))

        (*setf* (first lst) (cdr (first lst))))

        lst)

(*defun* swap-first-last (lst)

    (let

        ((temp (car lst))

        (lastEl (last lst)))

        (*setf* (car lst) (car lastEl))

        (*setf* (car lastEl) temp)

        lst))

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

(*defun* swap-two-elements (lst indF indS)

    (*and*

        (*and* (>= indF 0) (>= indS 0) (< indF (length lst)) (< indS (length lst)))

        (*setf* (nth indF lst) (cons (nth indF lst) (nth indS lst)))

        (*setf* (nth indS lst) (car (nth indF lst)))

        (*setf* (nth indF lst) (cdr (nth indF lst))))

        lst)

(*defun* swap-two-elements (lst indF indS)

    (let

        ((fEl (*nthcdr* indF lst))

        (temp nil)

        (sEl (*nthcdr* indS lst)))

        (*and*

            (*and* (>= indF 0) (>= indS 0) (< indF (length lst)) (< indS (length lst)))

            (*setf* temp (car fEl))

            (*setf* (car fEl) (car sEl))

            (*setf* (car sEl) temp))

        lst))

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

Решение:

(*defun* swap-to-left (lst)

    (let ((out nil))

    (*setf* out (copy-list (cdr lst)))

    (*setf* (cdr (last out)) (cons (first lst) nil))

    out))

(*defun* swap-to-right (lst)

    (let ((out nil))

    (*setf* out (copy-list lst))

    (*setf* (cdr (last out)) out)

    (*setf* out (cdr (*nthcdr* (- (length lst) 2) out)))

    (*setf* (cdr (*nthcdr* (- (length lst) 1) out)) nil)

    out))

# Теоретическая часть

1. Способы определения функций.

Первый способ:

(defun \*имя функции\* (\*список параметров\*) (

\*тело функции\*

)

)

Пример:

(*defun* findCat (gip cat)

    (sqrt ( - (\* gip gip) (\* cat cat))))

(findCat 5 4) -> 3.0;

Второй способ:

(lambda (\*список аргументов\*)) (\*тело функции\*))

Пример:

((lambda (a) (\* a 3)) 4) -> 12

lamda-функции называются «безымянными». Суть такой функции состоит в том, что задается алгоритм вычисления, но не задается имени функции. Подобную функцию можно применить к списку аргументов и сразу получить результат.

2. Варианты и методы модификации элементов списка.

Существует два вида функций работы со списками: структуроразрушающие и не разрушающие структуру функции.

Структуроразрушающими называются функции, при помощи которых можно вносить изменения во внутреннюю структуру уже существующих выражений.

К структуроразрушающим функциям относят: nreverse, rplaca, rplacd, nconc, nsubst и т.д.

К не разрушающим структуру функциям относят: append, reverse, remove и т.д.