|  |  |
| --- | --- |
| 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» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 8 |

**Дисциплина:** Функциональное и логическое программирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-62Б |  |  | Е.В. Брянская |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Н.Б.Толпинская |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2021

1. Напишите функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда:
   1. Все элементы списка – числа
   2. Элементы списка – любые объекты.

(defun mult (lst n)

(mapcar #'(lambda (x) (\* x n)) lst))

(defun mult-all (lst n)

(mapcar #'(lambda (x)

(cond ((numberp x)(\* x n))

((listp x)(mult-all x n))

(t x)))

lst))

1. Напишите функцию select-between, которая из списка аргумента, содержащего только числа, выбирает только те, которые расположены между двумя указанными границами-аргументами и возвращает их в виде списка (упорядоченного по возрастанию).

(defun find-elements (lst left right)

(remove-if #'(lambda (x) (null x))

(mapcar #'(lambda (x)

(if (< left x right)

x)) lst)))

(defun find-min (lst)

(setf temp (car lst))

(mapcar #'(lambda (x)

(if (> temp x)

(setf temp x))) lst)

temp)

(defun set-element (new old lst)

(cond ((= (car lst) old)(rplaca lst new))

(t (set-element new old (cdr lst))))

lst)

(defun my-sort (lst)

(maplist #'(lambda (x)

(and (setf temp (find-min x))

(set-element (car x) temp x)

(rplaca x temp))) lst)

lst)

(defun select-between (lst b1 b2)

(cond ((null lst) nil)

((not (and (numberp b1) (numberp b2)))(and (print "ERROR: wrong format of borders") nil))

((= b1 b2)(and (print "ERROR: wrong format of borders (equal)") nil))

((> b1 b2)(my-sort (find-elements lst b2 b1)))

((> b2 b1)(my-sort (find-elements lst b1 b2)))))

1. Что будет результатом (mapcar `вектор `(570-40-8))?

Ошибка, так как функция «вектор» не определена.

1. Напишите функцию, которая уменьшает на 10 все числа из списка-аргумента этой функции.

(defun f1 (lst)

(mapcar #'(lambda (x) (- x 10)) lst))

(defun f2 (lst)

(cond ((null lst) nil)

(t (cons (- (car lst) 10) (f2 (cdr lst))))))

1. Написать функцию, которая возвращает первый аргумент списка-аргумента, который сам является непустым списком.

(defun f (lst)

(cond ((null lst) nil)

((and (listp (car lst)) (> (length (car lst)) 0)) (car lst))

(t (f (cdr lst)))))

1. Найти сумму числовых элементов смешанного структурированного списка.

(defun sum-list (lst sum)

(cond ((null lst) sum)

((numberp (car lst)) (sum-list (cdr lst) (+ sum (car lst))))

((listp (car lst)) (sum-list (cdr lst) (sum-list (car lst) sum)))

(t (sum-list (cdr lst) sum))))

(defun find-sum-list(lst)

(if (null lst)

nil

(sum-list lst 0)))

**Вопросы**

1. Порядок работы и варианты использования функционалов.

Функционалы – функции, которые в качестве аргумента принимают другую функцию.

Бывают:

1. Применяющие (однократное применение функции к аргументам)

* (apply #’func arg\_list)
* (funcall #’func arg1 arg2 … argN)

1. Отображающие (многократное применение функции)

* (mapcar #’func ‘(x1 x2 … xN)) -> ((func x1)(func x2) … (func xN))
  + Получается N штук результатов, и mapcar объединяет эти результаты в один список с помощью list
  + Если функция принимает несколько аргументов, в таком случае, на вход подаются несколько списков. mapcar выбирает первые элементы из списков, применяет к ним функцию func, далее берет вторые и так далее. Допустимо подавать списки разной длины, в такой ситуации, работа функции завершается тогда, когда будет обработан самый коротки список.
* (maplist #’func lst)
  + Применяет функцию func к lst целиком, затем к хвосту, далее к хвосту хвоста и т.д.
  + Результаты объединяются в один список с помощью функции cons.
* (find-if #’predicat lst)
  + Находит первый элемент, который удовлетворяет предикату
  + Применяется поэлементно
  + Если находит элемент, то работа сразу прекращается
* (remove-if #’predicat lst)
  + Удаляет из списка элементы, которые удовлетворяют предикату
  + Не разрушает структуру
* (reduce #’func lst)
  + Применяет функцию func каскадно.
* (every #’predicat lst)
  + Возвращает T/Nil в зависимости от того, все ли элементы списка удовлетворяют предикату
* И т.д.