|  |  |
| --- | --- |
| **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»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Лабораторная работа № 9**

|  |  |
| --- | --- |
| **Тема** Использование функционалов и рекурсии  **Дисциплина** Функциональное и логическое программирование  **Студент** Потапчук А.А.  **Группа** ИУ7-63Б  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Строганов Ю.В. |  |

Москва.

2020 г.

**Цель работы:** приобрести навыки использования функционалов и рекурсии.

**Задачи работы:** изучить работу и методы использования отображающих функционалов: mapcar, maplist, reduce и др., изучить способы организации хвостовой рекурсии, сравнить эффективность.

**Практическая часть:**

**Лаб. работа № 5**

2. Написать предикат set-equal, который возвращает t, если два его множество-

аргумента содержат одни и те же элементы, порядок которых не имеет значения.

*Решение:*

(defun my\_member (el lst) ;

(cond ( (null lst) nil )

( (equal el (car lst)) t )

( t (my\_member el (cdr lst)) )

)

**)**

(defun set-equal (s1 s2)

(cond ((null s1) t)

((my\_member (car s1) s2) (set-equal (cdr s1) s2))

(t nil)

)

*)*

3. Напишите необходимые функции, которые обрабатывают таблицу из точечных пар:

(страна. столица), и возвращают по стране - столицу, а по столице — страну.

*Решение (table – таблица, являющаяся списокм точечных пар вида*

*(<страа> . <столица>)):*

(defun get-country(capital table)

(cond ((null table) nil)

((equal (cdar table) capital) (caar table))

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

)

)

(defun get-capital(country table)

(cond ((null table) nil)

((equal (caar table) country) (cdar table))

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

)

)

7. Напишите функцию, которая умножает на заданное число-аргумент все числа

из заданного списка-аргумента, когда

a) все элементы списка --- числа,

6) элементы списка **-- любые объекты**.

*Решение (mult - множитель):*

*а)* (defun num-list-mult (lst mult)

(mapcar

#'(lambda (num) (\* mult num))

lst

)

)

б) (defun list-mult (lst mult)

(mapcar

#'(lambda (list-item)

(cond ((null list-item) nil)

((numberp list-item) (\* mult list-item))

((not (atom list-item)) (list-mult list-item mult))

)

)

lst

)

)

**Лаб. работа № 6**

2. Напишите функцию, которая уменьшает на 10 все числа из списка

аргумента этой функции.

*Решение: эта функция аналогична функции из задания выше (с точностью до определения лямбда-функции). Если принципиально – напишу, но сейчас считаю, что, если задание выше решено правильно, этого можно не делать.*

3. Написать функцию, которая возвращает первый аргумент списка -аргумента.

который сам является непустым списком.

*Решение:*(defun first-list (lst)

(cond ((null lst) nil)

((not (atom (car lst))) (car lst))

(T (first-list (cdr lst)))

)

)

4. Написать функцию, которая выбирает из заданного списка только те числа,

которые больше 1 и меньше 10.

(Вариант: между двумя заданными границами. )

*Решение (floor ceil – соответственно нижняя и верхняя границы, из которых выбираем числа):*

(defun numbers-between (floor ceil lst)

(remove nil

(mapcar

#'(lambda (list-item)

(cond ((null list-item) nil)

((and (numberp list-item) (< list-item ceil) (> list-item floor)) list-item)

((not (atom list-item)) (numbers-between floor ceil list-item))

)

)

lst

)

)

)

5. Написать функцию, вычисляющую декартово произведение двух своих списков-аргументов. ( Напомним, что А х В это множество всевозможных пар (a b), где а принадлежит А, принадлежит В.)

*Решение:*

(defun decart (X Y)

(mapcan #'

(lambda (x)

(mapcar #'

(lambda (y) (list x y))

Y

)

)

X

)

)

6. Почему так реализовано reduce, в чем причина?

(reduce #\*+0) -> 0

(reduce #\*+ ()) -> 0

Думаю, что эта ф-я в основном применяется для численных вычислений, поэтому значение начальное значение аккумулятора по-умолчанию равно именно числу 0.