|  |  |
| --- | --- |
| 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 №10*

*По предмету: «Функциональное и логическое программирование»*

**Тема:** Определение функций пользователя.

Студент: Лаврова А. А.,

Группа: ИУ7-65Б

Преподаватель: Толпинская Н. Б.

Москва, 2020 г.

№7

Пусть list-of-list список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов list-of-list, т.е. например для аргумента ((1 2) (3 4)) -> 4.

(defun func (list-of-lists)

(reduce #'+

(mapcar (lambda(x)

(if (listp x) (func x) 1)

)

list-of-lists

)

)

)

Пример:

> (func '((1 2 3) ((4)) 5))

5

> (func '(1 2 3 4 5))

5

№8

Написать рекурсивную версию (с именем reg-add) вычисления суммы чисел

заданного списка.

Например: (reg-add (2 4 6)) -> 12

(defun rec\_add\_inner (lst sum)

(let ( (head (car lst))

(tail (cdr lst))

)

(cond ((null lst) sum)

((listp head) (rec\_add\_inner tail (rec\_add\_inner head sum)) )

((numberp head) (rec\_add\_inner tail (+ sum head)) )

(t (rec\_add\_inner tail sum))

)

)

)

(defun rec\_add (lst)

(if (eq lst nil)

nil

(rec\_add\_inner lst 0)

)

)

Пример:

> (rec\_add '(1 2 3 4))

10

> (print (rec\_add '(1 2 (3 (4)))))

10

№9

Написать рекурсивную версию с именем recnth функции nth.

(defun recnth (n lst)

(if (eq n 0) (car lst) (recnth (- n 1) (cdr lst))

)

)

Пример:

> (recnth 3 ‘(10 20 30 40 50))

40

№10

Написать рекурсивную функцию alloddr, которая возвращает t когда все

элементы списка нечетные.

(defun alloddr (lst)

(let ((head (car lst))

(tail (cdr lst))

)

(cond ((null lst) t)

((listp head)

(and (alloddr head) (alloddr tail))

)

((not (numberp head)) nil)

((evenp head) nil)

(t (alloddr tail))

)

)

)

Пример:

> (alloddr ‘(1 2 3 4 5))

NIL

> (alloddr '(1 3 5 7))

T

№11

Написать рекурсивную функцию, относящуюся к хвостовой рекурсии с одним тестом завершения, которая возвращает последний элемент списка - аргументы.

(defun last\_elem (curr)

(if (eq (cdr curr) nil)

(car curr)

(last\_elem (cdr curr))

)

)

Пример:

> (last\_elem '(1 2 3))

3

№12

Написать рекурсивную функцию, относящуюся к дополняемой рекурсии с  
одним тестом завершения, которая вычисляет сумму всех чисел от 0 до n-ого аргумента функции.

Вариант:

1) от п-аргумента функции до последнего >= 0,

2) от п-аргумента функции до т-аргумента с шагом d.

(defun sum\_all\_rec (lst n sum)

(cond

((or (null lst) (eq n 0)) sum)

(t (sum\_all\_rec (cdr lst) (- n 1) (+ sum (car lst))))

)

)

(defun sum\_all (lst n)

(sum\_all\_rec lst n 0))

Пример:

> (sum\_all '(1 2 3 4 5) 0)

0

> (sum\_all '(1 2 3 4 5) 10)

15

№13

Написать рекурсивную функцию, которая возвращает последнее нечетное  
число из числового списка, возможно создавая некоторые вспомогательные функции.

(defun get\_last\_odd\_inner (curr value)

(cond ((eq curr nil) value)

((oddp (car curr)) (get\_last\_odd\_inner (cdr curr) (car curr)))

(t (get\_last\_odd\_inner (cdr curr) value))

)

)

(defun get\_last\_odd (lst)

(get\_last\_odd\_inner lst nil)

)

Пример:

> (get\_last\_odd ‘(1 2 3 4 5))

5

> (get\_last\_odd ‘(1 2 3 4))

3

> (get\_last\_odd ‘(0 2 4 6))

NIL

№14

Используя cons-дополняемую рекурсию с одним тестом завершения,  
написать функцию которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке.

(defun square\_all (lst)

(cond

((null lst) nil)

(t (cons (\* (car lst) (car lst)) (square\_all (cdr lst))))

)

)

Пример:

> (square\_all '(1 2 3))

(1 4 9)

№15

Написать функцию с именем select-odd, которая из заданного  
списка выбирает все нечетные числа.

(defun select-odd-recur (lst)

(cond

((null lst) nil)

(t (cons (if (oddp (car lst)) (car lst) Nil) (select-odd-recur (cdr lst))))

)

)

(defun select-odd (lst)

(remove nil (select\_all\_odd lst))

)

Пример:

> (select-odd '(1 0 3 90 48 79))

(1 3 79)

**Теоретическая часть**

1. Способы организации повторных вычислений в Lisp

Функционал – функция, аргументом которого является функция.

Рекурсия - это ссылка на определяемый объект во время его определения.

1. Различные способы использования функционалов

Функционалы используются для реализации повторных вычислений или для обработки списковых таблиц.

Классификация функционалов:

* Применяющие
* Отображающие

1. Что такое рекурсия? Способы организации рекурсивных функций  
   Рекурсия — это ссылка на определяемый объект во время его  
   определения. Виды рекурсии: хвостовая, дополняемая, множественная,  
   взаимная и рекурсии высокого порядка.
2. Способы повышения эффективности реализации рекурсии  
   Организация хвостовой рекурсии повышает эффективность, однако для этого может потребоваться использование дополнительных параметров. Такая рекурсия может быть заменена на итерацию.