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 курсу: «Функциональное и логическое программирование»

Рекурсивные функции в Lisp

Студент: Набиев Ф.М.
Группа: ИУ7-63Б
Преподаватель: Толпинская Н.Б.

2020 г.

ВВЕДЕНИЕ

Целью работы является приобретение навыков организации рекурсии.

Задачи работы: изучить способы организации хвостовой, дополняемой, множественной, взаимной рекурсии и рекурсии более высокого порядка в Lisp.

1 Теоретические сведения

— Способы организации повторных вычислений в Lisp: функционалы, рекурсия.

— Что такое рекурсия? Классификация рекурсивных функций в Lisp: рекурсия — это ссылка на определяемый объект во время его определения; классификация: простая рекурсия - один рекурсивный вызов в теле; рекурсия первого порядка - рекурсивный вызов встречается несколько раз; взаимная рекурсия - используется несколько функций, рекурсивно вызывающих друг друга.

— Различные способы организации рекурсивных функций и порядок их реализации: хвостовая (результат формируется не на выходе из рекурсии, а на входе в рекурсию, все действия выполняя до ухода на следующий шаг рекурсии), по нескольким параметрам, дополняемая (при обращении к рекурсивной функции используется дополнительная функция не в аргументе вызова , а вне его), множественная (на одной ветке происходит сразу несколько рекурсивных вызовов).

— Способы повышения эффективности реализации рекурсии: применять хвостовую организацию рекурсии.

2 Практическая часть

2.1 Задание №1

Условие:

Пусть list-of-lists список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов list-of-lists.

В литинге 2.1 приведен текст решения данной задачи на языке Common Lisp.

Листинг 2.1 — Задание №1

```
1 (defun lvl2len (lol)
2   (cond ((null lol) 0)
3         (t (+ (length (car lol))
4               (lvl2len (cdr lol))))))
5
6 (lvl2len nil) ; 0
7 (lvl2len '((1 2) (3 4))) ; 4
```

2.2 Задание №2

Условие:

Написать рекурсивную версию функции вычисления суммы чисел заданного списка.

В литинге 2.2 приведен текст решения данной задачи на языке Common Lisp.

Листинг 2.2 — Задание №2

```
1 (defun reg-add (lst)
2   (cond ((null lst) 0)
3         (t (+ (cond ((numberp (car lst)) (car lst))
4                     (t 0))
5               (reg-add (cdr lst))))))
6
7 (reg-add '(2 4 6)) ; 12
8 (reg-add '(2 4 word 6 nil)) ; 12
```

```
 9 (reg-add '(word nil)) ; 0
10 (reg-add nil) ; 0
```

2.3 Задание №3

Условие:

Написать рекурсивную версию функции nth.

В литинге 2.3 приведен текст решения данной задачи на языке Common Lisp.

Листинг 2.3 — Задание №3

```
1 (defun recnth (idx lst)
2   (cond ((or (= 0 idx)
3              (null lst)) (car lst))
4         (t (recnth (- idx 1) (cdr lst)))))
5
6 (recnth 0 '(1 2 3)) ; 1
7 (recnth 2 '(1 2 3)) ; 3
8 (recnth 1 '(1 2 3)) ; 2
9 (recnth 5 '(1 2 3)) ; NIL
```

2.4 Задание №4

Условие:

Написать рекурсивную функцию, которая возвращает t когда все элементы списка нечетные.

В литинге 2.4 приведен текст решения данной задачи на языке Common Lisp.

Листинг 2.4 — Задание №4

```
1 (defun alloddr (lst)
2   (cond ((null lst) t)
3         ((evenp (car lst)) nil)
4         (t (alloddr (cdr lst)))))
5
```

```
6 (alloddr '(1 2 3)) ; nil
7 (alloddr '(1 3 5)) ; nil
```

2.5 Задание №5

Условие:

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

В литинге 2.5 приведен текст решения данной задачи на языке Common Lisp.

Листинг 2.5 — Задание №5

```
1 (defun latest (lst)
2   (cond ((null (cdr lst)) (car lst))
3         (t (latest (cdr lst)))))
4
5 (latest nil) ; NIL
6 (latest '(1)) ; 1
7 (latest '(1 2 3 (4 5))) ; (4 5)
```

2.6 Задание №6

Условие:

Написать рекурсивную функцию, относящуюся к дополняемой рекурсии с одним тестом завершения, которая вычисляет сумму всех чисел:

а) от 0 до n-ого аргумента функции;

б) от n-аргумента функции до последнего >= 0;

в) от n-аргумента функции до r-аргумента с шагом d.

В литинге 2.6 приведен текст решения данной задачи на языке Common Lisp.

Листинг 2.6 — Задание №6

```
;;; 1)
(defun sum-first (nums n)
  (cond ((or (= 0 n)
             (null nums)) 0)
        (t (+ (car nums)
              (sum-first (cdr nums) (- n 1))))))

;;; 2)
(defun help1 (nums n)
  (cond ((or (null nums) (= 0 n)) 0)
        (t (+ (car nums)
              (help1 (cdr nums) (- n 1))))))

(defun help2 (nums n)
  (cond ((or (null nums) (= 0 n)) 0)
        ((> (car nums) 0) (help1 nums n))
        (t (help2 (cdr nums) (- n 1)))))

(defun sum-to-last-pos (nums n)
  (help2 (reverse nums) (- (length nums) n)))

;;; 3)
(defun sum-range (nums lo hi h)
  (cond ((or (= 0 hi)
             (null nums)) 0)
        (t (+ (sum-range (cdr nums)
                         (cond ((= lo 0) h)
                               (t (- lo 1)))
                         (- hi 1)
                         h)
              (cond ((> lo 0) 0)
                    (t (car nums)))))))

(setq nums '(0 1 2 3 4 5 6 7 8 9))

(sum-first nums 5) ; 10
(sum-first nums 0) ; 0
(sum-first nums 1) ; 0
(sum-first nums 2) ; 1
(sum-first nums 10) ; 45

(sum-to-last-pos nums 5) ; 35
(sum-to-last-pos '(-1 2 3 4 -1) 2) ; 7
(sum-to-last-pos (concatenate 'list nums '(-10 5 -7 -9 -2)) 5) ; 30

(sum-range nums 1 9 2) ; 12
```

```
47 (sum-range nums 0 10 0) ; 45
48 (sum-range nums 0 10 1) ; 20
49 (sum-range nums 1 10 1) ; 25
```

2.7 Задание №7

Условие:

Написать рекурсивную функцию, которая возвращает последнее нечетное число из числового списка.

В литинге 2.7 приведен текст решения данной задачи на языке Common Lisp.

Листинг 2.7 — Задание №7

```
1  (defun help1 (lst lodd)
2    (cond ((null lst) lodd)
3          (t (help1 (cdr lst)
4                    (cond ((oddp (car lst)) (car lst))
5                          (t lodd))))))
6
7  (defun lastest-odd (lst)
8    (help1 lst nil))
9
10 (lastest-odd nil) ; NIL
11 (lastest-odd '(2 4 6 8)) ; NIL
12 (lastest-odd '(1 2 3 4 5 6 7 8 9 -1 10)) ; -1
```

2.8 Задание №8

Условие:

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.

В литинге 2.8 приведен текст решения данной задачи на языке Common Lisp.

Листинг 2.8 — Задание №8

```
(defun sqr-nums (nums)
  (cond ((null nums) nil)
        (t (cons (* (car nums) (car nums))
                 (sqr-nums (cdr nums))))))

(sqr-nums nil) ; NIL
(sqr-nums '(1 2 -3 4 5)) ; (1 4 9 16 25)
```

2.9 Задание №9

Условие:

Написать функцию, которая:

а) выбирает из списка чисел все чётные;

б) выбирает из списка чисел все нечётные;

в) суммирует все четные числа списка;

г) суммирует все нечетные числа списка.

В литинге 2.9 приведен текст решения данной задачи на языке Common Lisp.

Листинг 2.9 — Задание №9

```
(defun filter (predicate lst)
  (cond ((null lst) nil)
        (t (cond ((funcall predicate (car lst))
                  (cons (car lst) (filter predicate (cdr lst))))
                 (t (filter predicate (cdr lst)))))))

(defun select-odd (nums)
  (filter #'oddp nums))

(defun select-even (nums)
  (filter #'evenp nums))

(defun sum-all-odd (nums)
  (reduce #'+ (select-odd nums)))

(defun sum-all-even (nums)
```

```
17    (reduce #'+ (select-even nums)))
18
19  (setq digits '(0 1 2 3 4 5 6 7 8 9))
20
21  (select-odd digits) ; (1 3 5 7 9)
22  (sum-all-odd digits) ; 25
23
24  (select-even digits) ; (0 2 4 6 8)
25  (sum-all-even digits) ; 20
```