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»

Отчёт
к лабораторной работе № 9
По курсу: «Функциональное и логическое программирование»
Тема: «Работа интерпретатора Lisp.»

Студент Прохорова Л. А.

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватели Толпинская Н. Б., Строганов Ю. В.

Москва.
2021 г.

Цель работы: приобрести навыки использования рекурсии и функционалов.
Задачи работы: изучить способы применения функционалов и рекурсии при обработке одноуровневых и структурированных списков.

Задание 4. Написать функцию, которая выбирает из заданного списка только те числа, которые больше 1 и меньше 10.

С использованием рекурсии

```
(defun is_between (left right a)
   (if (> a left) (< a right) (> a right))
)

(defun getfromto_r (lst left right)
   ( cond
      ((null lst) Nil)
      (
         (listp (car lst))
         (nconc
            (getfromto_r (car lst) left right)
            (getfromto_r (cdr lst) left right))
      )
      (
         (and
            (numberp (car lst))
            (is_between left right (car lst))
         )
         (cons (car lst) (getfromto_r (cdr lst) left right))
      )

      (t (getfromto_r (cdr lst) left right))
   )
)
```

Тестирование

*(*getfromto_r *'(1 2 (3 4 (2) 3) 6) 0 4)-> (1 2 3 2 3)*
*(*getfromto_r *'(1 2 w e 3 4 5) 0 4)->(1 2 3)*
*(*getfromto_r *'(1 2 w e 3 4 5) 0 4) -> (1 2 3)*

С использованием функционалов

```
(defun is_between (left right a)
   (if (> a left) (< a right) (> a right))
```

```
)

(defun make_select_between (lst left right)
   (reduce
      #'(lambda (res el)
         (cond
            ((and (numberp el) (is_between left right el)) (append res (cons el Nil)))
            ((listp el) (append res (make_select_between el left right)))
            (t res)
         )

      )lst :initial-value nil
   )
)
```

Тестирование
(make_select_between '(1 2 (3 4 (2) 3) 6) 0 4)->(1 2 3 2 3)
(make_select_between '(1 2 w e 3 4 5) 0 4)->(1 2 3)
(make_select_between '(1 2 w e 3 4 5) 0 4)->(1 2 3)

Задание 5. Написать функцию, вычисляющую декартово произведение двух своих списков аргументов. (Напомним, что A x B это множество всевозможных пар (a b), где a), где a принадлежит A, принадлежит B.)

С использованием функционалов

```
(defun decart (a b)
   (mapcan
      #'(lambda (x)
         (mapcar #'(lambda (y) (list x y)) b )
      ) a
   )
)
```

Тестирование
(decart '(1 2 3 4) '(5 6 7 8))->((1 5) (1 6) (1 7) (1 8) (2 5) (2 6) (2 7) (2 8) (3 5) (3 6) (3 7) (3 8) (4 5)(4 6) (4 7) (4 8))

С использованием рекурсии

```
(defun dec_rec(el lst)
   ( cond
      ((null lst) Nil)
      (t (cons (cons el (cons (car lst) nil)) (dec_rec el (cdr lst))))
   )
)


(defun decart_rec(a b)
   ( cond
      ( (null a) nil)
      (t (nconc (dec_rec (car a) b) (decart_rec (cdr a) b)))
   )
)
```

Тестирование

(dec_rec 3 '(a d v))->((3 A) (3 D) (3 V))

(decart_rec '(1 2 3 4) '(5 6 7 8))->((1 5) (1 6) (1 7) (1 8) (2 5) (2 6) (2 7) (2 8) (3 5) (3 6) (3 7) (3 8) (4 5) (4 6) (4 7) (4 8))

6. Почему так реализовано reduce, в чем причина?

(reduce #'+ ()) -> 0

(reduce #'* ())-> 1

Сначала функция проверяет список-аргумент. Если он пуст, возвращается значение функции при отсутствии аргументов. Также reduce использует аргумент :initial-value. Этот аргумент определяет значение, к которому будет применена функция при обработке первого элемента списка-аргумента. Если список-аргумент пуст, то будет возвращено значение initial-value. Результатом вычисления функции + без аргументов будет 0, а результатом вычисления функции * без аргументов будет 1, т.к. это нейтральные элементы для данных операций.

Задание 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.

С использованием рекурсии

```
(defun sum_of_len_r(lst res)
   ( cond
      ((null lst) res)
      ((listp (car lst)) (sum_of_len_r (cdr lst) (+ res (length (car lst)))))
      (t (sum_of_len_r (cdr lst) (+ res 1)))
   )
)

(defun sum_of_len(lst)
   (sum_of_len_r lst 0)
)
```

Тестирование
(sum_of_len '((1 2) (3 4)))->4
(sum_of_len '((1 2) 3 (2 3)))->5

С использованием функционалов

```
(defun sum_of_len_f (lst)
   (reduce
      #'(lambda (res el)
         (cond
            ((listp el) (+ res (length el)))
            (t (+ res 1))
         )
      )lst :initial-value 0
   )
)
```

Тестирование
(sum_of_len_f '((1 2) (3 4)))->4
(sum_of_len_f '((1 2) 3 4 (5 6 7)))->7

Ответы на теоретические вопросы
Классификация рекурсивных функций

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

В LISP существует классификация рекурсивных функций:

- простая рекурсия - один рекурсивный вызов в теле
- рекурсия первого порядка - рекурсивный вызов встречается несколько раз
- взаимная рекурсия - используется несколько функций, рекурсивно вызывающих друг друга.

Виды рекурсии:

- Хвостовая. Результат формируется не на выходе из рекурсии, а на входе в рекурсию, все действия выполняются до ухода на следующий шаг рекурсии.
- 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. Количество условий выхода также может зависеть от задачи.
- Взаимная.
- Рекурсии высокого порядка.

При организации рекурсии можно использовать как функции с именем, так и локально определенные с помощью лямбда выражений функции. Кроме этого, при организации рекурсии можно использовать функционалы или использовать рекурсивную функцию внутри функционала.