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 № 6

Тема Рекурсивные функции

Студент Сушина А.Д.

Группа ИУ7-61б

Оценка (баллы) _____

Преподаватель Толпинская Н.Б.

Москва.
2020 г

Цель работы: приобрести навыки организации рекурсии в lisp.

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

1. Переписать функцию how-alike, приведенную в лекции и не использующую COND, используя конструкция IF, AND/OR.

Функция из лекции:

```
(defun how_alike (x y)
    (cond ((or (= x y) (equal x y)) `the_same)
        ((and (oddp x) (oddp y)) `both_odd)
        ((and (evenp x) (evenp y)) `both_even)
        (t `difference)))
```

Перепишем:

```
(defun how_alike2 (x y)
    (if (or (= x y) (equal x y))
        'the_same
        (if (and (oddp x) (oddp y))
            'both_odd
            (if (and (evenp x) (evenp y))
                'both_even
                'defference))))
```

2. (дополнительно) Дано два списка: название стран (4 шт) и список столиц.

Создать

а) список из двухэлементных списков

```
(defun make_list (lst1 lst2) (mapcar #'(lambda (x y) (cons x (cons y nil))) lst1 lst2))
```

> **(make_list '(moscow kiev) '(russia ukraine))** → ((MOSCOW RUSSIA) (KIEV UKRAINE))

б) список точечных пар.

```
(defun make_pair (lst1 lst2) (mapcar 'cons lst1 lst2))
```

> **(make_pair '(moscow kiev) '(russia ukraine))** →((MOSCOW . RUSSIA) (KIEV . UKRAINE))

По созданным спискам:

1 - по столице найти страну

2 - по стране найти столицу

```
(defun find_by_func(val lst)
   (find-if (lambda (x) (not (null x)))
         (mapcar (lambda (pair)
                  (cond
                     ((equal (car pair) val)
                      (cond ((atom (cdr pair)) (cdr pair))
                            (t (cadr pair))))
                     ((equal (cond ((atom (cdr pair)) (cdr pair))
                                   (t (cadr pair)))
                             val)
                      (car pair))
                     )
                  )
                 lst))
)
```

Функция find_by_func работает и для списков из двух элементов, и для точечных пар. Обеспечивает поиск в обе стороны. И страны по столице, и столицы по стране.

> (find_by_func 'russia (make_pair '(moscow kiev) '(russia ukraine))) → MOSCOW

> (find_by_func 'moscow (make_pair '(moscow kiev) '(russia ukraine))) → RUSSIA

Ответы на вопросы:

1) способы определения функции

Определение функций в Lisp происходит с помощью оператора defun. На вход подается три или более параметров: имя функции, аргументы и одно или более выражений, которые составляют тело функции.

Например: (defun sum(arg1, arg2) (+ arg1 arg2))

Также функций можно определять через лямбда выражения. Лямбда выражение — это список, содержащий в себе слово lambda и список аргументов и следующие за ним тело функции, состоящее из 0 или более выражений.

>(lambda (x y) (+ x y))

2) вызов функции, блокировка работы

Обращении к функции или при вызове apply для лямбда выражений запускается функция eval, выполняющая обработку программы(s выражения). Для блокировки выполнения обычно используется функция ‘ (quote). Если eval была применена явно, блокировка quote не сработает, так как eval обеспечивает дополнительный вызов интерпретатора (При этом вызов может производится внутри вычисляемого S-выражения).

3) глобальные и локальные символьные атомы

Глобальные символьные атомы — значение устанавливается с помощью setf. Область видимости весь код следующий после определения.

Локальные значение — значение устанавливается с помощью let(let*) Область видимости является тело функции, в которой определена переменная.