Министерство образования Российской Федерации
Московский Государственный Технический Университет
им. Н.Э. Баумана

Отчет по лабораторной работе №6
По курсу «Функциональное и логическое программирование»

Студент Медведев А.В.
Группа ИУ7-62
Преподаватель Толпинская Н.Б.

Москва, 2019

Теоретическая часть

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))

>(sum(1 5))

6

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 и блокировка

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.

```
>(setf a 9)
>a
9
```

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

```
>(let ((x 1) (y 2))
        (+ x y))
3
```

Практическая часть

2.7) Написать функцию, которая переводит температуру в системе Фаренгейта в температуру по Цельсию (defun f-to-c (temp) ...). Как бы назывался роман Р. Брэдбери «+451 по Фаренгейту" в системе по Цельсию?

Формулы:

- c = 5/9*(f-320)
- f = 9/5*c+32.0.

```
(defun f-to-c (temp)
        (* (/ 5.0 9.0) (- temp 320)))
```

Роман назывался бы «+72.77778 по Цельсию».

2.8). Что получится при вычислении каждого из выражений?

1. **(list 'cons t NIL)**
   (CONS T NIL)

2. **(eval (list 'cons t NIL))**
   (T)

3. **(eval (eval (list 'cons t NIL)))**
   Ошибка. Функция «T» не определена.

4. **(apply #'cons '(t NIL))**
   (T)

5. **(eval NIL)**
   NIL

6. **(list 'eval NIL)**
   (EVAL NIL)

7. **(eval (list 'eval NIL))**
   NIL

3.1. Написать функцию, которая принимает целое число и возвращает первое четное число, не меньшее аргумента.

```
(defun first-even (x)
        (cond ((evenp x)(+ x 2))
                        (T (+ x 1))))
(lambda (x) (cond ((evenp x) (+ x 2)) (T (+ x 1))))
```

```
* (first-even 11)
12
* (first-even 12)
14
```

3.2 Написать функцию, которая принимает число и возвращает число того же знака, но с модулем на 1 больше модуля аргумента.

```
(defun custom_abs (x)
        (+ x (cond ((> x  0) 1)
                        (T -1)))
* (custom_abs 10)
11
* (custom_abs -10)
-11
```

3.3. Написать функцию, которая принимает два числа и возвращает список из этих чисел, расположенный по возрастанию.

```
(defun sorted_list (x y)
        (cond ((< x  y) (list x y))
                        (T (list y x)))
* (sorted_list 12 11)
(11 12)
* (sorted_list 11 12)
(11 12)
```

3.4. Написать функцию, которая принимает три числа и возвращает T только тогда, когда первое число расположено между вторым и третьим.

```
(defun between  (x y z)
        (or (and (> x y) (< x z))
                (and (> x z) (< x y))
))
```

** (between 1 2 3)*
NIL
** (between 2 1 3)*
T
** (between 3 1 2)*
NIL
** (between 2 3 1)*
T

3.5. Каков результат следующих выражений?

a. **(and `fee `fie `foe)**

   FOE

b. **(or `fee `fie `foe)**

   FEE

c. **(and (equal `abc `abc) `yes)**

   YES

d. **(or (equal `abc `abc) `yes)**

   T

e. **(or nil `fi `foe)**

   FI

f. **(and nil `fie `foe)**

   NIL