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

по лабораторной работе №3

Название «Работа интерпретатора Lisp»

Дисциплина «Функциональное и логическое программирование»

Студент ИУ7-65Б _____ Бугаенко А.П.
(подпись, дата) (Фамилия И.О.)

Преподователь _____ Толпинская Н.Б.
(подпись, дата) (Фамилия И.О.)

Москва, 2022

1 Цели и задачи работы

Цель работы — приобрести навыки работы в системе Common Lisp. Задачи работы — изучить работу форм — функций, которые особым образом обрабатывают свои аргументы и особенности их работы в Lisp.

2 Теоретические вопросы

2.1 Базис Lisp

Базис Lisp составляют атомы, структуры, базовые функции и встроенные и спецальные функционалы.

2.2 Классификация функций

Функция - однозначное отображение множества значений аргументов в значение функции.
Функциональный язык - тот, который базируется на понятии функции.
Функциональность системы - предоставляемые пользователю возможности.

Классификация функций по аргументам и поведению.

— чистые функции (математические функции) - имеют фиксированное число аргументов, для определённого набора аргументов один фиксированный результат

— формы (специальные функции) - функции, принимающие на вход произвольное количество аргументов, или по-разному обрабатывающее аргументы.

— псевдо-функции - функции, обладающие побочным эффектом. Побочный эффект - событие, изменяющее сознание системы. Пример - setf, связывающее атом и значение и format, выводящее значение на экран.

— функционалы - принимают функции в качестве параметров либо в качестве возвращаемого значения выступает функция.

Классификация функций по именованию.

— именованные - есть имя, определяется через defun. Специальные символы (T, Nil) и самоопределимые атомы (натуральные, вещественные числа, строки) не могут выступать в роли функции.

— неименованные - нет имени, через lambda.

2.3 Способы создания функций

Создание именованной функции - синтаксис:
(defun имя список_аргументов лямбда-выражение)

Создание неименованной функции - синтаксис:
(lambda список_аргументов лямбда-выражение)
(lambda $(x_1, ..., x_k)$ форма)

2.4 Функции cond, if, and/or

Функция cond - (cond (test_1 body_1) (test_2 body_2) ... (test_i body_i) [(t body_i+1)]), test_i - тестирующееся выражение, body_i - выполняется, если test_i корректный.
Функция if - (if test than else)/(if test than) - если нет else, то автоматически возвращается nil.
Логические функции and/or (and test_1 test_2 ... test_n) - поочереди вычисляет test_1 — test_n, если одно из них nil, то возвращает nil, иначе возвращает последнее вычисленное значение. (or test_1 test_2 ... test_n) - поочереди вычисляет test_1 — test_n, возвращает первое вычисленное значение которое не является nil.

3 Практические задания

3.1 Задание 1

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

```
(defun even_greater_and_equal (x)
  (if (= (logand x 1) 0)
      x
      (+ x 1)))
```

3.2 Задание 2

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

```
(defun abs_inc (x)
  (if (>= x 0)
      (+ x 1)
      (- x 1)))
```

3.3 Задание 3

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

```
(defun mysort (a b)
  (if (>= a b)
      (list b a)
      (list a b)))
```

3.4 Задание 4

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

```
(defun func (a b c)
  (if (and (> b a) (> c b))
      t
  ))
```

3.5 Задание 5

Каков результат вычисления следующих выражений?

(and 'fee 'fie 'foe) → 'foe
(or 'fee 'fie 'foe) → 'fee
(or nil 'fie 'foe) → 'fie
(and nil 'fie 'foe) → nil
(and (equal 'abc 'abc) 'yes) → 'yes
(or (equal 'abc 'abc) 'yes) → t

3.6 Задание 6

Написать предикат, который принимает два числа-аргумента и возвращает T, если первое число не меньше второго.

```
1 (defun pred (a b)
2   (>= a b))
```

3.7 Задание 7

Какой из следующих двух вариантов предиката ошибочен и почему?

(defun pred1 (x)
(and (numberp x) (plusp x)))

(defun pred2 (x)
(and (plusp x)(numberp x)))

Ошибочен второй предикат, так как если x - не число, то возникнет ошибка, поскольку проверка на число проходит после plusp.

3.8 Задание 8

Решить задачу 4, используя для ее решения конструкции IF, COND, AND/OR.

С if:

```
1 (defun func (a b c)
2   (if (and (> b a) (> c b))
3       t
4   ))
```

C cond:

```
(defun func (a b c)
  (cond ((and (> b a) (> c b) t))))
```

3.9 Задание 9

Переписать функцию how-alike, приведенную в лекции и использующую COND, используя только конструкции IF, AND/OR.

Реализация с cond

```
(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)))
```

Реализация с if

```
(defun how-alike (x y)
  (if (if (= x y)
          (equal x y))
      'the_same
      (if (if (oddp x)
              (oddp y))
          'both_odd
          (if (if (evenp x)
                  (evenp y))
              'both_even
              'difference))))
```

Реализация с and/or

```
(defun how-alike (x y)
  (or (and (or (= x y) (equal x y)) 'the_same)
  (and (oddp x) (oddp y) 'both_odd)
  (and (evenp x) (evenp y) 'both_even)
  'difference))
```