



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»

Отчет по лабораторной работе №3 по курсу "Функциональное и логическое программирование"

Тема Работа интерпретатора Lisp

Студент Цветков И.А.

Группа ИУ7-63Б

Оценка (баллы) _____

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

Москва — 2022 г.

1 Практические задания

1.1 Задание 1

Условие: написать функцию, которая принимает целое число и возвращает первое четное число, не меньшее аргумента.

```
1 (defun find_even (num) (if (evenp num) num (+ num 1)))
2 ; (find_even 5) -> 6
3 ; (find_even 6) -> 6
```

1.2 Задание 2

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

```
1 (defun inc_module (num) (if (< num 0) (- num 1) (+ num 1)))
2 ; (inc_module 5) -> 6
3 ; (inc_module -5) -> -6
4 ; (inc_module 0) -> 1
```

1.3 Задание 3

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

```
1 (defun mini_sort (a b) (if (> a b) (list b a) (list a b)))
2 ; (mini_sort 2 1) -> (2 1)
```

1.4 Задание 4

Условие: написать функцию, которая принимает три числа и возвращает Т только тогда, когда первое число расположено между вторым и третьим.

```
1 (defun is_middle (a b c) (cond ((and (> a b) (< a c)) T) ((and
    (> a c) (< a b)) T)))
```

1.5 Задание 5

Условие: каков результат вычисления следующих выражений?

```
1 (and 'fee 'fie 'foe) ; FOE
2 (or Nil 'fie 'foe) ; FIE
3 (and (equal 'abc 'abc) 'yes) ; YES
4 (or 'fee 'fie 'foe) ; FEE
5 (and nil 'fie 'foe) ; Nil
6 (or (equal 'abc 'abc) 'yes) ; T
```

1.6 Задание 6

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

```
1 (defun predicate (a b) (>= a b))
2 ; (predicate 2 1) -> T
```

1.7 Задание 7

Условие: какой из следующих двух вариантов предиката ошибочен и почему?

```
1 (defun pred1 (x) (and (numberp x) (pluss x))) ; Ok
2 (defun pred2 (x) (and (pluss x)(numberp x))) ; Если x не число,
    то функция pluss вернет ошибку <Not a number>
```

1.8 Задание 8

Условие: решить задачу 4, используя для ее решения конструкции IF, COND, AND/OR.

```
1      ; if
2      (defun between_if (a b c) (if (< b a c) (< b a c) (> b a c)))
3
4      ; cond
5      (defun between_cond (a b c) (cond ((< b a c) T) ((< c a b) T)))
6
7      ; and/or
8      (defun between_andor (a b c) (or (< b a c) (< c a b)))
```

1.9 Задание 9

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

Исходная функция how_alike представлена ниже.

```
1      (defun how_alike(x y)
2      (cond ((or (= x y) (equal x y)) 'the_same)
3              ((and (oddp x) (oddp y)) 'both_odd)
4              ((and (evenp x) (evenp y)) 'both_even)
5              (t 'difference)))
```

Функция how_alike через if представлена ниже.

```
1      (defun how_alike(x y)
2      (if (or (= x y) (equal x y)) 'the_same
3          (if (and (oddp x) (oddp y)) 'both_odd
4              (if (and (evenp x) (evenp y)) 'both_even 'difference))))
```

Функция how_alike через and/or представлена ниже.

```
1      (defun how_alike(x y)
2      (or (and (or (= x y) (equal x y)) 'the_same)
3          (and (and (oddp x) (oddp y)) 'both_odd)
4          (and (and (evenp x) (evenp y)) 'both_even)
5          'difference))
```

2 Ответы на вопросы к лабораторной работе

2.1 Базис Lisp.

Базис языка – минимальный набор конструкций языка и структур данных, с помощью которых можно решить любую задачу.

Базис состоит из:

1. атомы и структуры (представляющиеся бинарными узлами);
2. базовые (несколько) функций и функционалов: встроенные примитивные функции (`atom`, `eq`, `cons`, `car`, `cdr`); специальные функции и функционалы (`quote`, `cond`, `lambda`, `eval`, `apply`, `funcall`).

2.2 Классификация функций.

Функции в Lisp классифицируют следующим образом:

- «чистые» математические функции;
- рекурсивные функции;
- специальные функции — формы (от вызова к вызову может меняться количество аргументов, или обрабатываться по-разному);
- псевдофункции (создают эффект на внешнем устройстве);
- функции с вариативными значениями, из которых выбирается 1;
- функционалы (в качестве аргумента – функция, или возвращает в качестве результата функцию).

По назначению функции разделяются следующим образом:

1. конструкторы — создают значение (`cons`, например);

2. селекторы — получают доступ по адресу (`car`, `cdr`);
3. предикаты — возвращают `Nil`, `T`.

2.3 Способы создания функций.

Построить функцию можно с помощью Lambda-выражения (базисный способ)

Lambda-определение безымянной функции:

```
1 (lambda <Lambda-список> <форма>),
```

где Lambda-список – список аргументов, а форма – тело функции.

Lambda-вызов функции:

```
1 (<Lambda-выражение> <формальные параметры>)
```

Функции с именем. В таких функциях `defun` связывает символьный атом с Lambda-определением:

```
1 (defun f <Lambda-выражение>)
```

Упрощенное определение:

```
1 (defun f(x1, ..., xk) (<формы>))
```

2.4 Работа функций `Cond`, `if`, and/or.

`cond` – средство разветвления вычислений. Условное выражение вычисляется по следующим правилам:

1. Последовательно вычисляются условия ветвей до тех пор, пока не встретится выражение-форма, значение которой отлично от `Nil`.
2. Затем вычисляется выражение соответствующей ветви и его значение возвращается в качестве значения функции.
3. Если все условия имеют значение `Nil`, то значением условного выражения становится `Nil`.

1 `(cond (p1 e1) (p2 e2) ... (pn en))`

`if` – макрофункция. Если значение выражения `C` отлично от `Nil`, то вычисляется выражение `E1`, иначе значение `E2`.

1 `(if C E1 E2)`

`or` – функция-дизъюнкции.

1 `(or e1 e2 ... en)`

При выполнении вызова последовательно вычисляются аргументы `ei` (слева направо) – до тех пор, пока не встретится значение `ei`, отличное от `Nil`. В этом случае вычисление прерывается и значение функции равно значению этого `ei`. Если же вычислены значения всех аргументов `ei`, и оказалось, что они равны `Nil`, то результирующее значение функции равно `Nil`.

`and` – функция-конъюнкции.

1 `(and e1 e2 ... en)`

При вычислении этого функционального обращения последовательно слева направо вычисляются аргументы функции `ei` – до тех пор, пока не встретится значение, равное `Nil`. В этом случае вычисление прерывается и значение функции равно `Nil`. Если же были вычислены все значения `ei` и оказалось, что все они отличны от `Nil`, то результирующим значением функции `and` будет значение последнего выражения `en`.