



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 №3 по курсу "Функциональное и логическое программирование"

Тема Работа интерпретатора Lisp

Студент Ковалец К. Э.

Группа ИУ7-63Б

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

Москва — 2022 г.

1 Практические задания

1.1 Задание 1

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

Листинг 1.1 – Решение задания 1

```
1 (defun first-even (num)
2   (if (evenp num) num (+ num 1)))
3
4 ;; (FIRST-EVEN 1) -> 2
5 ;; (FIRST-EVEN 2) -> 2
```

1.2 Задание 2

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

Листинг 1.2 – Решение задания 2

```
1 (defun module-plus (num)
2   (+ num (if (> num 0) 1 -1)))
3
4 ;; (MODULE-PLUS 1) -> 2
5 ;; (MODULE-PLUS -1) -> -2
```

1.3 Задание 3

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

Листинг 1.3 – Решение задания 3

```
1 (defun ascending-list (num1 num2)
2   (if (> num1 num2) (list num2 num1) (list num1 num2)))
3
4 ;; (ASCENDING-LIST 1 2) -> (1 2)
5 ;; (ASCENDING-LIST 2 1) -> (1 2)
```

1.4 Задание 4

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

Листинг 1.4 – Решение задания 4

```
1 (defun between (num1 num2 num3)
2   (or (and (> num1 num2) (< num1 num3))
3       (and (< num1 num2) (> num1 num3))))
4
5 ;; (BETWEEN 2 1 3) -> T
6 ;; (BETWEEN 2 3 1) -> T
7 ;; (BETWEEN 1 2 3) -> NIL
8 ;; (BETWEEN 4 3 2) -> NIL
```

1.5 Задание 5

Каков результат вычисления следующих выражений?

Листинг 1.5 – Решение задания 5

```
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

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

Листинг 1.6 – Решение задания 6

```
1 (defun predicate (num1 num2)
2   (>= num1 num2))
3
4 ;; (PREDICATE 1 2) -> NIL
5 ;; (PREDICATE 2 1) -> T
```

1.7 Задание 7

Какой из следующих двух вариантов предиката ошибочен и почему?

- **numberp** — возвращает Т, если значение аргумента — числовой атом, Nil иначе;
- **plusp** — возвращает Т, если аргумент больше нуля, Nil иначе.

Листинг 1.7 – Решение задания 7

```
1 (defun pred1 (x)
2   (and (numberp x) (plusp x))) ;; OK
3
4 (defun pred2 (x)
5   (and (plusp x) (numberp x))) ;; ERROR
6
7 ;; (PRED1 1) -> T
8 ;; (PRED2 1) -> T
9
10 ;; (PRED1 'Hello) -> NIL
11 ;; (PRED2 'Hello) -> The value HELLO is not of type NUMBER
```

Во втором случае при попытке проверить, является ли аргумент больше нуля, может возникнуть ошибка, если аргумент не является числовым атомом, так как при вызове функции **and** аргументы вычисляются слева направо до первого вернувшего Nil

1.8 Задание 8

Решить задачу 4, используя для ее решения конструкции IF, COND, AND/OR. Написать функцию, которая принимает три числа и возвращает T только тогда, когда первое число расположено между вторым и третьим.

Листинг 1.8 – Решение задания 8

```
1 ;; С использованием IF
2
3 (defun between (num1 num2 num3)
4   (if (> num1 num2)
5       (< num1 num3) (> num1 num3)))
6
7 ;; С использованием COND
8
9 (defun between (num1 num2 num3)
10  (cond ((> num1 num2) (< num1 num3))
11        ((< num1 num2) (> num1 num3))))
12
13 ;; С использованием AND/OR
14
15 (defun between (num1 num2 num3)
16  (or (and (> num1 num2) (< num1 num3))
17      (and (< num1 num2) (> num1 num3))))
18
19 ;; (BETWEEN 2 1 3) -> T
20 ;; (BETWEEN 2 3 1) -> T
21 ;; (BETWEEN 1 2 3) -> NIL
22 ;; (BETWEEN 4 3 2) -> NIL
```

1.9 Задание 9

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

Листинг 1.9 – Функция how-alike из лекции

```
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)))
```

Листинг 1.10 – Решение задания 9

```
1 ;; С использованием IF
2
3 (defun how_alike(x y)
4   (if (or (= x y) (equal x y)) 'the_same
5       (if (and (oddp x) (oddp y)) 'both_odd
6           (if (and (evenp x) (evenp y)) 'both_even 'difference))))
7
8 ;; С использованием AND/OR
9
10 (defun how_alike(x y)
11   (or (and (or (= x y) (equal x y)) 'the_same)
12       (and (and (oddp x) (oddp y)) 'both_odd)
13       (and (and (evenp x) (evenp y)) 'both_even)
14       'difference))
15
16 ;; (HOW_ALIKE 4 4) -> THE_SAME
17 ;; (HOW_ALIKE 4 6) -> BOTH_EVEN
18 ;; (HOW_ALIKE 1 3) -> BOTH_ODD
19 ;; (HOW_ALIKE 4 5) -> DIFFERENCE
```

2 Ответы на теоретические 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 языка Lisp содержит:

- атомы и структуры (представляющиеся бинарными узлами);
- базовые функции и функционалы:
 - 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 (прнимают произвольное число аргументов или по разному обрабатывают аргументы);
- псевдофункции (создают эффект на внешнем устройстве);
- функции с вариативными значениями, из которых выбирается одно;
- функции высших порядков – функционалы (используются для построения синтаксически управляемых программ).

По назначению функции разделяются следующим образом:

- конструкторы — создают значение (`cons`, `list`);
- селекторы — получают доступ по адресу (`car`, `cdr`);
- предикаты — возвращают `Nil`, `T`.

2.3 Способы создания функций

Функцией называется правило, по которому каждому значению одного или нескольких аргументов ставится в соответствие конкретное значение результата.

- В Lisp можно определить функцию без имени с помощью **λ -выражений**. Lambda-определение безымянной функции:

```
(lambda <lambda-список> <форма>)
```

Lambda-вызов функции:

```
(<lambda-выражение> <формальные параметры>)
```

- Также в Lisp можно определить функцию с именем с помощью **defun**. В таких функциях `defun` связывает символьный атом с Lambda-определением:

```
(defun f <lambda-выражение>)
```

Упрощенное определение:

```
(defun f(arg1, ..., argN) <формы>)
```


2.4 Работа функций Cond, if, and/or

- **cond**

```
1 (cond (test1 body1)
2       (test2 body2)
3       ...
4       (testN bodyN)
5       [(T else-body)])
```

Список аргументов обрабатывается последовательно: вычисляется выражение `testi`, и если не `Nil`, то вычисляется `bodyi`, и работа функции завершается, если ни один тест не выполнен, то возвращается `Nil`, можно организовать ветку «else», явно указав в качестве `test` - `T`.

- **if**

```
1 (if test T-body F-body)
```

Работа функции `if` очевидна, с учетом, что всё что не `nil`, то `T`. Результат теста может быть как атомом (не обязательно `Nil`) так и списком. В зависимости от `test`, будет вычислен либо один либо другой аргумент.

- **and**

```
1 (and arg1 arg2 ... argN)
```

Функция `and` вычисляет аргументы, пока не станет очевидным результат (появится первый `nil`). Как только станет очевиден результат — возвращается последнее вычисленное значение.

- **or**

```
1 (or arg1 arg2 ... argN)
```

Функция `or` вычисляет аргументы, пока не станет очевидным результат (появится первый не `nil`). Как только станет очевиден результат — возвращается последнее вычисленное значение.