

**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 № 2  
по дисциплине «Функциональное и логическое  
программирование»**

**Тема Определение функций пользователя.**

**Студент Фам Минь Хиеу**

**Группа ИУ7-62Б**

**Преподаватель Толпинская Н.Б., Строганов Ю.В.**

Москва, 2024

# СОДЕРЖАНИЕ

<b>1</b>	<b>Теоретическая часть . . . . .</b>	<b>3</b>
1.1	Базис языка . . . . .	3
1.2	Классификация функций . . . . .	3
1.3	Способы создания функций . . . . .	4
1.4	Функции <code>car</code> и <code>cdr</code> . . . . .	4
1.5	Функции <code>eq</code> , <code>eq1</code> , <code>equal</code> , <code>equalp</code> . . . . .	5
1.6	Назначение и отличие <code>list</code> от <code>cons</code> . . . . .	7
<b>2</b>	<b>Практическая часть . . . . .</b>	<b>8</b>
2.1	Задание 1 . . . . .	8
2.2	Задание 2 . . . . .	8
2.3	Задание 3 . . . . .	10
2.4	Задание 4 . . . . .	10
2.5	Задание 5 . . . . .	11
2.6	Задание 6 . . . . .	11
2.7	Задание 7 . . . . .	12
2.8	Задание 8 . . . . .	12

# 1 Теоретическая часть

## 1.1 Базис языка

Базис состоит из:

1. структуры, атомы;
2. встроенные (примитивные) функции (atom, eq, cons, car, cdr);
3. специальные функции и функционалы, управляющие обработкой структур, представляющих вычислимые выражения (quote, cond, lambda, label, eval).

## 1.2 Классификация функций

Функции в Lisp классифицируют следующим образом:

- чистые математические функции;
- рекурсивные функции;
- специальные функции — формы (сегодня 2 аргумента, завтра - 5);
- псевдофункции (создают эффект на внешнем устройстве);
- функции с вариативными значениями, из которых выбирается 1;
- функции высших порядков — функционал: используется для синтаксического управления программ (абстракция языка).

По назначению функции разделяются следующим образом:

- конструкторы — создают значение (cons, например);

- селекторы — получают доступ по адресу (`car`, `cdr`);
- предикаты — возвращают `Nil`, `T`.
- функции сравнения – такие как: `eq`, `eq1`, `equal`, `equalp`.

## 1.3 Способы создания функций

Функции в Lisp можно задавать следующими способами:

### Lambda-выражение

Синтаксис:

(`lambda` <λ-список> форма)

Пример:

Листинг 1.1 — Функция определенная Lambda-выражением

```
1 (lambda (a b) (sqrt (+ (* a a) (* b b))))
```

### Именованная функция

Синтаксис:

(`defun` <имя функции> <λ-выражение>)

Пример:

Листинг 1.2 — Пример определения именованной функции

```
1 (defun hyp (a b) (sqrt (+ (* a a) (* b b))))
```

## 1.4 Функции `car` и `cdr`

`car` — функция получения первого элемента точечной пары.

Примеры:

`cdr` — функция получения второго элемента точечной пары.

S-выражение	Результат выполнения car
(A . B)	A
((A . B) . C)	(A . B)
A	ошибка

S-выражение	Результат выполнения cdr
(A . B)	B
(A . (B . C))	(B . C)
A	ошибка

## 1.5 Функции eq, eql, equal, equalp

(eq x y) является истиной тогда и только тогда, когда, x и y являются идентичными объектами. (В реализациях, x и y обычно равны eq тогда и только тогда, когда обращаются к одной ячейке памяти.)

Предикат eql истинен, если его аргументы равны eq, или если это числа одинакового типа и с одинаковыми значениями, или если это одинаковые буквы.

Предикат equal истинен, если его аргументы это структурно похожие (изоморфные) объекты. Грубое правило такое, что два объекта равны equal тогда и только тогда, когда одинаково их выводимое представление. Числа и буквы сравниваются также как и в eql. Символы сравниваются как в eq. Объекты, которые содержат другие элементы, будут равны equal, если они принадлежат одному типу и содержащиеся элементы равны equal.

Два объекта равны equalp, если они равны equal, если они буквы и удовлетворяют предикату char-equal, который игнорирует регистр и другие атрибуты символов, если они числа и имеют одинаковое значение, даже если числа разных типов, если они включает в себя элементы, которые также равны equalp.

### Листинг 1.3 — Примеры

```

1 (eq 'a 'b) is false .
2 (eq 'a 'a) is true .
3 (eq 3 3) might be true or false , depending on the implementation .
4 (eq 3 3.0) is false .

```

```

5 (eq 3.0 3.0) might be true or false , depending on the
   implementation .
6 (eq #c(3 -4) #c(3 -4))
7 might be true or false , depending on the implementation .
8 (eq #c(3 -4.0) #c(3 -4)) is false .
9 (eq (cons 'a 'b) (cons 'a 'c)) is false .
10 (eq (cons 'a 'b) (cons 'a 'b)) is false .
11 (eq '(a . b) '(a . b)) might be true or false .
12 (progn (setq x (cons 'a 'b)) (eq x x)) is true .
13 (progn (setq x '(a . b)) (eq x x)) is true .
14 (eq #\A #\A) might be true or false , depending on the
   implementation .
15 (eq "Foo" "Foo") might be true or false .
16 (eq "Foo" (copy-seq "Foo")) is false .
17 (eq "FOO" "foo") is false .
18
19
20 (eql 'a 'b) is false .
21 (eql 'a 'a) is true .
22 (eql 3 3) is true .
23 (eql 3 3.0) is false .
24 (eql 3.0 3.0) is true .
25 (eql #c(3 -4) #c(3 -4)) is true .
26 (eql #c(3 -4.0) #c(3 -4)) is false .
27 (eql (cons 'a 'b) (cons 'a 'c)) is false .
28 (eql (cons 'a 'b) (cons 'a 'b)) is false .
29 (eql '(a . b) '(a . b)) might be true or false .
30 (progn (setq x (cons 'a 'b)) (eql x x)) is true .
31 (progn (setq x '(a . b)) (eql x x)) is true .
32 (eql #\A #\A) is true .
33 (eql "Foo" "Foo") might be true or false .
34 (eql "Foo" (copy-seq "Foo")) is false .
35 (eql "FOO" "foo") is false .
36
37
38 (equal 'a 'b) is false .
39 (equal 'a 'a) is true .
40 (equal 3 3) is true .
41 (equal 3 3.0) is false .
42 (equal 3.0 3.0) is true .

```

```

43 (equal #c(3 -4) #c(3 -4)) is true .
44 (equal #c(3 -4.0) #c(3 -4)) is false .
45 (equal (cons 'a 'b) (cons 'a 'c)) is false .
46 (equal (cons 'a 'b) (cons 'a 'b)) is true .
47 (equal '(a . b) '(a . b)) is true .
48 (progn (setq x (cons 'a 'b)) (equal x x)) is true .
49 (progn (setq x '(a . b)) (equal x x)) is true .
50 (equal #\A #\A) is true .
51 (equal "Foo" "Foo") is true .
52 (equal "Foo" (copy-seq "Foo")) is true .
53 (equal "FOO" "foo") is false .

```

## 1.6 Назначение и отличие list от cons

cons — функция конструирования точечной пары, на вход получает 2 значения и делает из них точечную пару.

list — функция конструирования списка. На вход получает произвольное количество элементов и делает из них список.

Вызовы (list 1 2 3 4) и (cons 1 (cons 2 (cons 3 (cons 4 Nil)))) эквивалентны, то есть дают одинаковый результат

## 2 Практическая часть

### 2.1 Задание 1

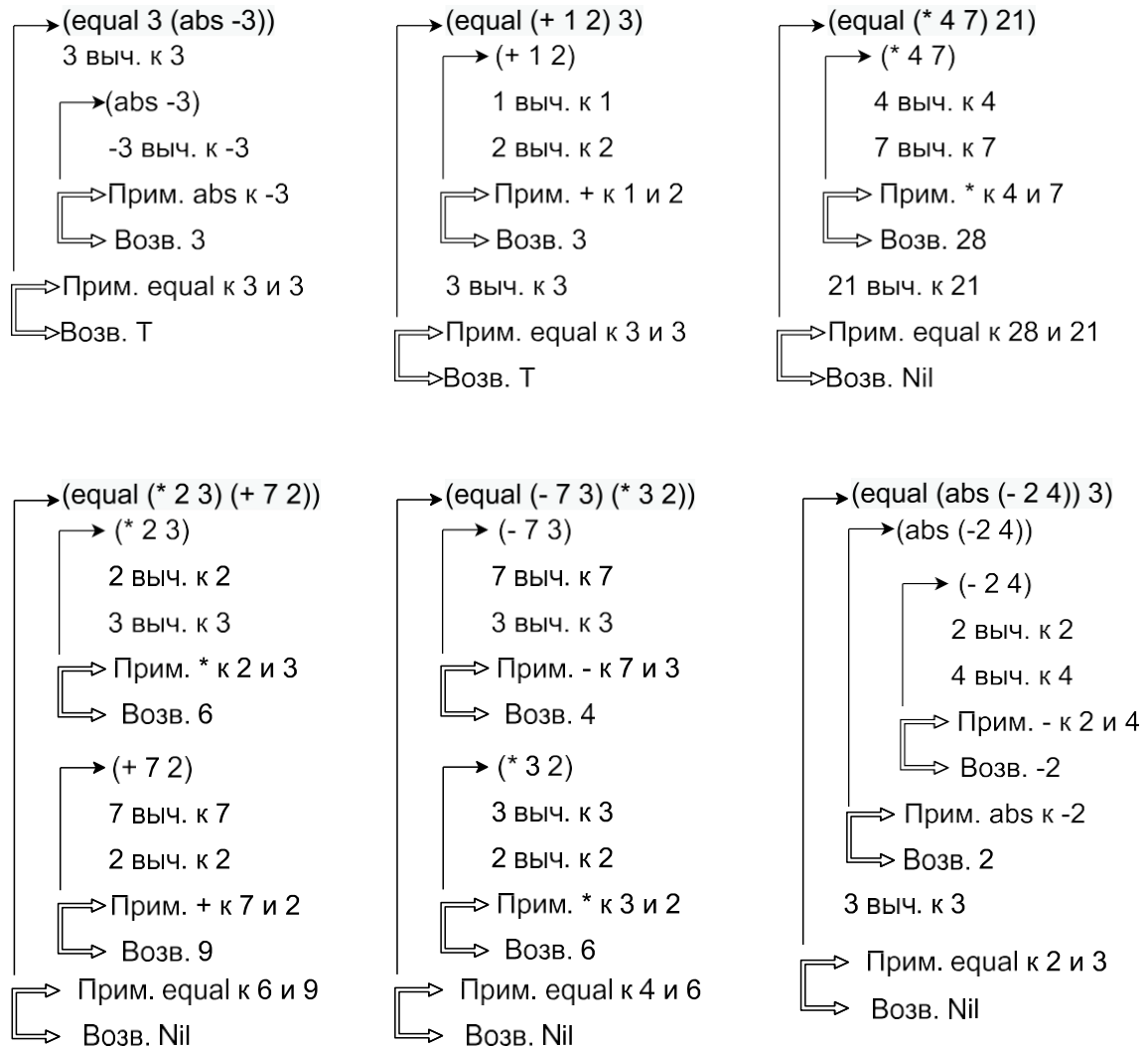


Рис. 2.1 – Часть 1

### 2.2 Задание 2

Листинг 2.1 — Выражение 1

```
(defun calc_hyp (a b) (sqrt (+ (* a a) (* b b))))
```





Рис. 2.2 – ss

## 2.3 Задание 3

Листинг 2.2 — Выражение 1

```
1 (list 'a c) ;The variable C is unbound.; (list 'a 'c)
```

Листинг 2.3 — Выражение 2

```
1 (cons 'a (b c)) ;The variable C is unbound.; (cons 'a '(b c))
```

Листинг 2.4 — Выражение 3

```
1 (cons 'a '(b c)) ;(A B C)
```

Листинг 2.5 — Выражение 4

```
1 (caddr (1 2 3 4 5)) ;Compile-time error: illegal function call; (  
  caddr '(1 2 3 4 5))
```

Листинг 2.6 — Выражение 5

```
1 (cons 'a 'b 'c) ;invalid number of arguments: 3; (cons '(a b) 'c)  
  (cons 'a '(b c))
```

Листинг 2.7 — Выражение 6

```
1 (list 'a (b c)) ;The variable C is unbound.; (list 'a '(b c))
```

Листинг 2.8 — Выражение 7

```
1 (list a '(b c)) ;The variable A is unbound.; (list 'a '(b c))
```

Листинг 2.9 — Выражение 8

```
1 (list (+ 1 '(length '(1 2 3)))) ; The value (LENGTH '(1 2 3)) is  
  not of type NUMBER; (list (+ 1 (length '(1 2 3))))
```

## 2.4 Задание 4

Листинг 2.10 — Выражение 1

```
1 (defun longer_than(f s)(> (length f) (length s)))
```

## 2.5 Задание 5

Листинг 2.11 — Выражение 1

```
1 (cons 3 (list 5 6)) ;(3 5 6)
```

Листинг 2.12 — Выражение 2

```
1 (list 3 'from 9 'lives (- 9 3)) ;(3 FROM 9 LIVES 6)
```

Листинг 2.13 — Выражение 3

```
1 (+ (length for 2 too)) (car '(21 22 23))) ;The variable FOR is  
unbound.
```

Листинг 2.14 — Выражение 4

```
1 (cdr '(cons is short for ans)) ;(IS SHORT FOR ANS)
```

Листинг 2.15 — Выражение 5

```
1 (car (list one two)) ;The variable ONE is unbound.
```

Листинг 2.16 — Выражение 6

```
1 (cons 3 '(list 5 6)) ;(3 LIST 5 6)
```

Листинг 2.17 — Выражение 7

```
1 (car (list 'one 'two)) ;ONE
```

## 2.6 Задание 6

Листинг 2.18 — Выражение 1

```
1 (mystery (one two)) ;The variable TWO is unbound.
```

Листинг 2.19 — Выражение 2

```
1 (mystery (last one two)) ;The variable ONE is unbound.
```

Листинг 2.20 — Выражение 3

```
1 (mystery free) ;The variable FREE is unbound.
```

#### Листинг 2.21 — Выражение 4

```
1 (mystery one 'two)) ;The variable ONE is unbound.
```

## 2.7 Задание 7

#### Листинг 2.22 — Выражение 1

```
1 (defun f-to-c (temp) (* (/ 5 9) (- temp 32.0)))
```

При переводе из 451 по Фаренгейту в Цельсии получится 232.77779 градуса.

## 2.8 Задание 8

#### Листинг 2.23 — Выражение 1

```
1 (list 'cons t NIL) ;(CONS T NIL)
```

#### Листинг 2.24 — Выражение 2

```
1 (eval (eval (list 'cons t NIL))) ;The function COMMON-LISP:T is  
undefined.
```

#### Листинг 2.25 — Выражение 3

```
1 (apply #cons "(t NIL)) ;illegal complex number format: #CONS
```

#### Листинг 2.26 — Выражение 4

```
1 (list 'eval NIL) ;(EVAL NIL)
```

#### Листинг 2.27 — Выражение 5

```
1 (eval (list 'cons t NIL)) ;(T)
```

#### Листинг 2.28 — Выражение 6

```
1 (eval NIL) ;NIL
```

#### Листинг 2.29 — Выражение 7

```
1 (eval (list 'eval NIL)) ;NIL
```