**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

**1 Теоретическая часть** . . . . . . . . . . . . . . . . . . . . . . . . . . **3**

1.1 Базис языка . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.2 Классификация функций . . . . . . . . . . . . . . . . . . . . . 3

1.3 Способы создания функций . . . . . . . . . . . . . . . . . . . . 4 1.4 Функции car и cdr . . . . . . . . . . . . . . . . . . . . . . . . . 4

* 1. Функции eq, eql, equal, equalp . . . . . . . . . . . . . . . . . . . 5
  2. Назначение и отличие list от cons . . . . . . . . . . . . . . . . . 7

**2 Практическая часть** . . . . . . . . . . . . . . . . . . . . . . . . . . **8**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

# Теоретическая часть

## 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).

## 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 синтак- сического 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, eql, equal, equalp.

## 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

( lambda ( a b ) ( **s q r t** ( + (\* a a ) (\* b b ) ) ) )

1

## Именованная функция

Синтаксис:

(defun <имя функции> <*λ*-выражение>) Пример:

Листинг 1.2 — Пример определения именованной функции

( **defun** hyp ( a b ) ( **s q r t** ( + (\* a a ) (\* b b ) ) ) )

1

## 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 |

## Функции eq, eql, equal, equalp

(eq x y) является истиной тогда и только тогда, когда, x и y являют- ся 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 чис- ла 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 удо- влетворяют предикату char-equal, который игнорирует регистр и другие ат- рибуты символов, если они числа и имеют одинаковое значение, даже если числа разных типов, если они включает в себя элементы, которые также рав- ны equalp.

Листинг 1.3 — Примеры

( **eq** ’ a ’ b ) i s f a l s e .

( **eq** ’ a ’ a ) i s t r u e .

( **eq** 3 3 ) might be t r u e **or** f a l s e , depending on t h e i m p l e m e n t a t i o n . ( **eq** 3 3 . 0 ) i s f a l s e .

1

2

3

4

5 ( **eq** 3 . 0 3 . 0 ) might be t r u e **or** f a l s e , depending on t h e i m p l e m e n t a t i o n .

6 ( **eq** # c ( 3 − 4 ) # c ( 3 − 4 ) )

7 might be t r u e **or** f a l s e , depending on t h e i m p l e m e n t a t i o n .

8 ( **eq** # c ( 3 − 4 . 0 ) # c ( 3 − 4 ) ) i s f a l s e .

9 ( **eq** ( **cons** ’ a ’ b ) ( **cons** ’ a ’ c ) ) i s f a l s e .

10 ( **eq** ( **cons** ’ a ’ b ) ( **cons** ’ a ’ b ) ) i s f a l s e .

11 ( **eq** ’ ( a . b ) ’ ( a . b ) ) might be t r u e **or** f a l s e .

12 ( progn ( s e t q x ( **cons** ’ a ’ b ) ) ( **eq** x x ) ) i s t r u e .

13 ( progn ( s e t q x ’ ( a . b ) ) ( **eq** x x ) ) i s t r u e .

14 ( **eq** # \A # \A) might be t r u e **or** f a l s e , depending on t h e i m p l e m e n t a t i o n .

15 ( **eq** ” Foo ” ” Foo ” ) might be t r u e **or** f a l s e .

16 ( **eq** ” Foo ” ( copy−seq ” Foo ” ) ) i s f a l s e .

17 ( **eq** ”FOO” ” foo ” ) i s f a l s e .

18

19

20 ( **e q l** ’ a ’ b ) i s f a l s e .

21 ( **e q l** ’ a ’ a ) i s t r u e .

22 ( **e q l** 3 3 ) i s t r u e .

23 ( **e q l** 3 3 . 0 ) i s f a l s e .

24 ( **e q l** 3 . 0 3 . 0 ) i s t r u e .

25 ( **e q l** # c ( 3 − 4 ) # c ( 3 − 4 ) ) i s t r u e .

26 ( **e q l** # c ( 3 − 4 . 0 ) # c ( 3 − 4 ) ) i s f a l s e .

27 ( **e q l** ( **cons** ’ a ’ b ) ( **cons** ’ a ’ c ) ) i s f a l s e .

28 ( **e q l** ( **cons** ’ a ’ b ) ( **cons** ’ a ’ b ) ) i s f a l s e .

29 ( **e q l** ’ ( a . b ) ’ ( a . b ) ) might be t r u e **or** f a l s e .

30 ( progn ( s e t q x ( **cons** ’ a ’ b ) ) ( **e q l** x x ) ) i s t r u e .

31 ( progn ( s e t q x ’ ( a . b ) ) ( **e q l** x x ) ) i s t r u e .

32 ( **e q l** # \A # \A) i s t r u e .

33 ( **e q l** ” Foo ” ” Foo ” ) might be t r u e **or** f a l s e .

34 ( **e q l** ” Foo ” ( copy−seq ” Foo ” ) ) i s f a l s e .

35 ( **e q l** ”FOO” ” foo ” ) i s f a l s e .

36

37

38 ( **equal** ’ a ’ b ) i s f a l s e .

39 ( **equal** ’ a ’ a ) i s t r u e .

40 ( **equal** 3 3 ) i s t r u e .

41 ( **equal** 3 3 . 0 ) i s f a l s e .

42 ( **equal** 3 . 0 3 . 0 ) i s t r u e .

43

( **equal** ( **equal** ( **equal** ( **equal** ( **equal** ( progn ( progn ( **equal** ( **equal** ( **equal**

( **equal**

# c ( 3 − 4 ) # c ( 3 − 4 ) ) i s t r u e .

# c ( 3 − 4 . 0 ) # c ( 3 − 4 ) ) i s f a l s e .

( **cons** ( **cons** ’ ( a .

( s e t q

( s e t q

’ a

’ a

b )

’ b )

’ b )

’ ( a

( **cons**

( **cons**

. b ) )

’ a

’ a i s

x ( **cons** ’ a ’ b ) )

’ c ) ) i s f a l s e .

’ b ) ) i s t r u e . t r u e .

( **equal** x x ) ) i s t r u e .

x ’ ( a . b ) ) ( **equal** x x ) ) i s t r u e .

# \A # \A) i s t r u e .

” Foo ”

” Foo ”

”FOO”

” Foo ” ) i s

( copy−seq ” foo ” ) i s

t r u e .

” Foo ” ) ) i s t r u e . f a l s e .

44

45

46

47

48

49

50

51

52

53

## Назначение и отличие list от cons

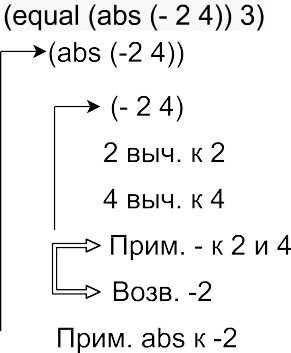
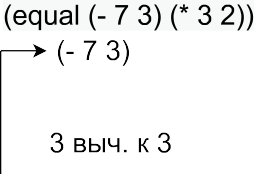
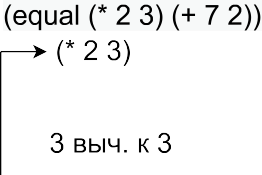
cons — функция конструирования точечной пары, на вход получает 2 значения и делает из них точечную пару.

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

Вызовы (list 1 2 3 4) и (cons 1 (cons 2 (cons 3 (cons 4 Nil)))) эквивалент- ны, то есть дают одинаковый результ

# Практическая часть

## Задание 1



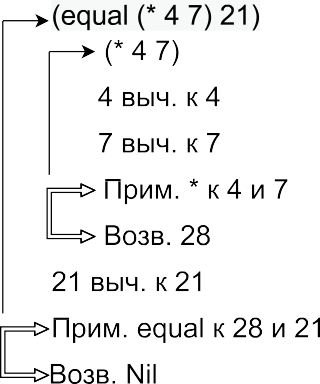
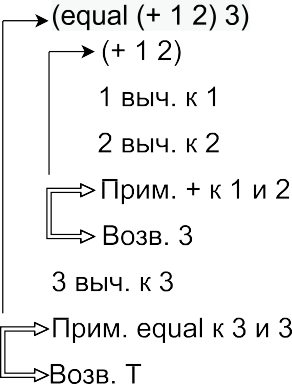
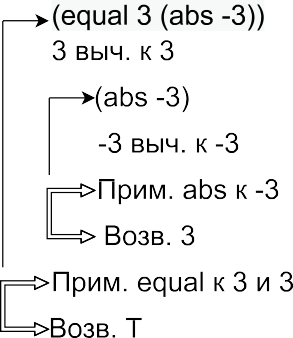


Рис. 2.1 – Часть 1



## Задание 2

Листинг 2.1 — Выражение 1

( **defun** calc\_hyp ( a b ) ( **s q r t** ( + (\* a a ) (\* b b ) ) ) )

1





Рис. 2.2 – ss



## Задание 3

Листинг 2.2 — Выражение 1

( **l i s t** ’ a c ) *; The v a r i a b l e C i s unbound . ; ( l i s t ’ a ’ c )*

1

Листинг 2.3 — Выражение 2

( **cons** ’ a ( b c ) ) *; The v a r i a b l e C i s unbound . ; ( cons ’ a ’ ( b c ) )*

1

Листинг 2.4 — Выражение 3

( **cons** ’ a ’ ( b c ) ) *; ( A B C)*

1

Листинг 2.5 — Выражение 4

( c a d d r ( 1 2 3 4 5 ) ) *; Compile−time e r r o r : i l l e g a l f u n c t i o n c a l l ; ( caddr ’ ( 1 2 3 4 5 ) )*

1

Листинг 2.6 — Выражение 5

( **cons** ’ a ’ b ’ c ) *; i n v a l i d number o f arguments : 3 ; ( cons ’ ( a b ) ’ c ) ( cons ’ a ’ ( b c ) )*

1

Листинг 2.7 — Выражение 6

( **l i s t** ’ a ( b c ) ) *; The v a r i a b l e C i s unbound . ; ( l i s t ’ a ’ ( b c ) )*

1

Листинг 2.8 — Выражение 7

( **l i s t** a ’ ( b c ) ) *; The v a r i a b l e A i s unbound . ; ( l i s t ’ a ’ ( b c ) )*

1

Листинг 2.9 — Выражение 8

( **l i s t** ( + 1 ’ ( **l e n g t h** ’ ( 1 2 3 ) ) ) ) *;*

*not o f t y p e NUMBER; ( l i s t (+ 1*

*The value ( LENGTH ’ ( 1 2 3 ) ) i s*

*( l e n g t h*

*’ ( 1 2 3 ) ) ) )*

1

## Задание 4

Листинг 2.10 — Выражение 1

( **defun** l onger\_then ( f s ) (> ( **length** f ) ( **length** s ) ) )

1

## Задание 5

Листинг 2.11 — Выражение 1

( **cons** 3 ( **l i s t** 5 6 ) ) *; ( 3 5 6 )*

1

Листинг 2.12 — Выражение 2

( **l i s t** 3 ’ from 9 ’ l i v e s ( − 9 3 ) ) *; ( 3 FROM 9 LIVES 6 )*

1

Листинг 2.13 — Выражение 3

( + ( **l e n g t h** f o r 2 t o o ) ) ( c a r ’ ( 21 22 23 ) ) ) *; The v a r i a b l e FOR i s unbound .*

1

Листинг 2.14 — Выражение 4

( c d r ’ ( **cons** i s s h o r t f o r ans ) ) *; ( IS SHORT FOR ANS)*

1

Листинг 2.15 — Выражение 5

( c a r ( **l i s t** one two ) ) *; The v a r i a b l e ONE i s unbound .*

1

Листинг 2.16 — Выражение 6

( **cons** 3 ’ ( **l i s t** 5 6 ) ) *; ( 3 LIST 5 6 )*

1

Листинг 2.17 — Выражение 7

( c a r ( **l i s t** ’ one ’ two ) ) *;ONE*

1

## Задание 6

Листинг 2.18 — Выражение 1

( mystery ( one two ) ) *; The v a r i a b l e TWO i s unbound .*

1

Листинг 2.19 — Выражение 2

( mystery ( **l a s t** one two ) ) *; The v a r i a b l e ONE i s unbound .*

1

Листинг 2.20 — Выражение 3

( mystery f r e e ) *; The v a r i a b l e FREE i s unbound .*

1

Листинг 2.21 — Выражение 4

( mystery one ’ two ) ) *; The v a r i a b l e ONE i s unbound .*

1

## Задание 7

Листинг 2.22 — Выражение 1

( **defun** f − t o − c ( temp ) (\* ( / 5 9 ) ( − temp 3 2 . 0 ) ) )

1

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

## Задание 8

Листинг 2.23 — Выражение 1

( **l i s t** ’ **cons** t NIL ) *; ( CONS T NIL )*

1

Листинг 2.24 — Выражение 2

( **e v a l** ( **e v a l** ( **l i s t** ’ **cons** t NIL ) ) ) *; The f u n c t i o n COMMON−LISP: T i s u n d e f i n e d .*

1

Листинг 2.25 — Выражение 3

( **apply** # **cons** ” ( t NIL ) ) ; i l l e g a l complex number f o r m a t : #CONS

1

Листинг 2.26 — Выражение 4

( **l i s t** ’ **e v a l** NIL ) *; ( EVAL NIL )*

1

Листинг 2.27 — Выражение 5

( **e v a l** ( **l i s t** ’ **cons** t NIL ) ) *; ( T )*

1

Листинг 2.28 — Выражение 6

( **e v a l** NIL ) *; NIL*

1

Листинг 2.29 — Выражение 7

( **e v a l** ( **l i s t** ’ **e v a l** NIL ) ) *; NIL*

1