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 к лабораторной работе №3 по курсу: «Функциональное и логическое программирование»

Тема: **Определение функций пользователя**

Студент группы **ИУ7-62Б** **А.П. Сорокин**
(И.О. Фамилия)

Преподаватель **Н.Б. Толпинская**
(И.О. Фамилия)

2020 г.

СОДЕРЖАНИЕ

1 Цели и задачи

Цель работы: приобрести навыки создания и использования функций пользователя в Lisp.

Задачи работы: изучить способы создания и использования именованных и неименованных функций пользователя для обработки списков.

2 Теоретическая часть

Базис - это это минимально необходимый набор конструкций, с помощью которого можно реализовать задачу.

Классификация функций:

- чистые (математические): принимают строго определённое число аргументов и возвращают одно значение
- формы: могут принимать разное число аргументов, в зависимости от чего по-разному себя ведёт
- функционалы: принимают функциональные описания.

Классификация базовых функций Lisp:

- функции-селекторы (функции доступа к данным): car, cdr
- функции-конструкторы: cons
- функции-предикаты: atom, Null, lisp и т. д.
- функции-сравнения: eq, eql, =, equal, equalp

Списки в оперативной памяти представляются с помощью списковых ячеек. Списковая ячейка состоит из двух частей - полей first и rest. В этих полях хранятся два указателя: в поле first указатель ссылается на голову (car), в поле rest - на хвост (cdr). Эти указатели могут ссылаться на другую списочную ячейку или на атом.

Функции CAR и CDR являются базовыми функциями-селекторами. **CAR** принимает в качестве аргумента точечную пару или список. Функция возвращает голову списка. В случае точечной пара или непустого списка функ-

ция вернёт первый элемент, в случае пустого списка - nil.

CDR также принимает в качестве аргумента точечную пару или список. Функция возвращает хвост списка, т.е. список, состоящий из всех элементов, кроме первого. Если в списке меньше двух элементов, то функция возвращает nil.

Функции LIST и CONS являются функциями-конструкторами, причём функция CONS является базовой, а LIST - нет.

CONS создает списочную ячейку и устанавливает два указателя на принимаемые два аргумента.

LIST принимает переменное число аргументов и возвращает список, элементами которого являются аргументы функции.

Пример использования функций CONS и LIST для создания списка (a b c):

Листинг 1 – Создание списка (a b c)

```
1    (cons 'a (cons 'b (cons 'c nil)))
2    (list 'a 'b 'c)
```

3 Практическая часть

3.1 Задание 1

Составить диаграмму вычисления для указанных выражений.

Задание 1.1. (equal 3 (abs - 3))

⟶ (equal 3 (abs - 3))

- вычисление 3 к 3

⟶ (abs - 3)

⟶ -3

- вычисление 3 к 3

⟹ применение - к 3

⟹ -3

⟹ применение abs к -3

⟹ 3

⟹ применение equal к 3, 3

⟹ T

Задание 1.2. (equal (+ 1 2) 3)

⟶ (equal (+ 1 2) 3):

⟶ (+ 1 2)

- вычисление 1 к 1
- вычисление 2 к 2

⟹ применение + к 1, 2

⟹ 3

⟹ применение equal к 3, 3

⟹ T

Задание 1.3. (equal (* 4 7) 21)

⟶ (equal (* 4 7) 21)

⟶ (* 4 7)

- вычисление 4 к 4
- вычисление 7 к 7

⟹ применение * к 4, 7

$\Longrightarrow$ 28

• 21

$\Longrightarrow$ применение equal к 28, 21

$\Longrightarrow$ NIL

Задание 1.4. (equal (* 2 3) (+ 7 2))

$\longrightarrow$ (equal (* 2 3) (+ 7 2))

$\longrightarrow$ (* 2 3):

• вычисление 2 к 2

• вычисление 3 к 3

$\Longrightarrow$ применение * к 2, 3

$\Longrightarrow$ 6

$\longrightarrow$ (+ 7 2)

• вычисление 7 к 7

• вычисление 2 к 2

$\Longrightarrow$ применение + к 7, 2

$\Longrightarrow$ 9

$\Longrightarrow$ применение equal к 6, 9

$\Longrightarrow$ NIL

Задание 1.5. (equal (- 7 3) (* 3 2))

$\longrightarrow$ (equal (- 7 3) (* 3 2))

$\longrightarrow$ (- 7 3)

• вычисление 7 к 7

• вычисление 3 к 3

$\Longrightarrow$ применение - к 7, 3

$\Longrightarrow$ 4

$\longrightarrow$ (* 3 2)

• вычисление 3 к 3

• вычисление 2 к 2

$\Longrightarrow$ применение * к 3, 2

$\Longrightarrow$ 6

$\Longrightarrow$ применение equal к 4, 6

$\Longrightarrow$ NIL

Задание 1.6. (equal (- 7 3) (* 3 2))

⟶ (equal (abs (- 2 4)) 3)

⟶ (abs (- 2 4))

⟶ (- 2 4)

- вычисление 2 к 2
- вычисление 4 к 4

⟹ применение - к 2, 4

⟹ -2

⟹ применение abs к -2

⟹ 2

- 3

⟹ применение equal к 2, 3

⟹ NIL

3.2 Задание 2

Функция, вычисляющая гипотенузу прямоугольного треугольника по заданным катетам:

Листинг 2 – Функция вычисления гипотенузы

```
1     (defun hypot (cath1 cath2) (sqrt (+ (* cath1 cath1) (* cath2 cath2))))
```

Диаграмма вычисления функции:

⟶ (hypot a b)

- вычисление a к a
- вычисление b к b

⟹ применение hypot к a, b

- создание переменной cath1 со значением a
- создание переменной cath2 со значением b

⟶ (sqrt (+ (* cath1 cath1) (* cath2 cath2)))

⟶ (+ (* cath1 cath1) (* cath2 cath2))

⟶ (* cath1 cath1)

- вычисление cath1 к a
- вычисление cath1 к a

⟹ применение * к a, a

$\Longrightarrow a^2$

$\longrightarrow$ (* cath2 cath2)

- вычисление cath2 к b
- вычисление cath2 к b

$\Longrightarrow$ применение * к b, b

$\Longrightarrow b^2$

$\Longrightarrow$ применение + к a^2, b^2

$\Longrightarrow a^2 + b^2$

$\Longrightarrow$ применение sqrt к $a^2 + b^2$

$\Longrightarrow \sqrt{a^2 + b^2}$

$\Longrightarrow \sqrt{a^2 + b^2}$

3.3 Задание 3

Функция, вычисляющая объём параллелепипеда по трём его сторонам:

Листинг 3 – Функция вычисления объёма параллелепипеда

```
1     (defun v (a1 a2 a3) (* a1 a2 a3))
```

Диаграмма вычисления функции:

$\longrightarrow$ (v a b c)

- вычисление a к b
- вычисление b к b
- вычисление c к c

$\Longrightarrow$ применение v к a, b, c

- создание переменной a1 со значением a
- создание переменной a2 со значением b
- создание переменной a3 со значением c

$\longrightarrow$ (* a1 a2 a3)

- вычисление a1 к a
- вычисление a2 к b
- вычисление a3 к c

$\Longrightarrow$ применение * к a, b, c

$\Longrightarrow a \cdot b \cdot c$

$\Longrightarrow a \cdot b \cdot c$

3.4 Задание 4

Результаты вычисления выражений представлены в таблице 1. Значком * в номере выражения обозначается исправленное выражение, которое возможно вычислить.

Таблица 1 – Выражения и результаты их вычислений задания 4

№	Выражение	Результат
1	(list 'a c)	UNBOUND-VARIABLE c
2	(cons 'a (b c))	UNBOUND-VARIABLE c
2*	(cons 'a '(b c))	(a b c)
3	(cons 'a '(b c))	(a b c)
4	(caddy (1 2 3 4 5))	Illegal function call 1, caddy is undefined
4*	(caddr '(1 2 3 4 5))	3
5	(cons 'a'b'c)	Invalid number of arguments
5*	(cons 'a'b)	(a . b)
6	(list 'a (b c))	UNBOUND-VARIABLE c
6*	(list 'a '(b c))	(a (b c))
7	(list a '(b c))	UNBOUND-VARIABLE a
7*	(list 'a '(b c))	(a (b c))
8	(list (+ 1 '(length '(1 2 3))))	TYPE-ERROR
8*	(list (+ 1 (length '(1 2 3))))	(4)

3.5 Задание 5

Функция longer_then от двух списков-аргументов, которая возвращает T, если первый аргумент имеет большую длину.

Листинг 4 – Функция longer_then с использованием length

```
1    (defun longer_than (list1 list2) (> (length list1) (length list2)))
```

Листинг 5 – Функция longer_then с использованием базисных функций

```
1    (defun longer_than_2 (list1 list2)
2        (cond ((null list1) nil)
3              ((null list2) T)
4              (T (longer_than_2 (cdr list1) (cdr list2)))))
```

3.6 Задание 6

Результаты вычисления выражений представлены в таблице 2.

Таблица 2 – Выражения и результаты их вычислений задания 6

№	Выражение	Результат
1	(cons 3 (list 5 6))	(3 5 6)
2	(cons 3 ’(list 5 6))	(3 list 5 6)
3	(list 3 ’from 9 ’lives (- 9 3))	(3 from 9 lives 6)
4	(+ (length for 2 too) (car ’(21 22 23)))	UNBOUND-VARIABLE for
4*	(+ (length ’(for 2 too)) (car ’(21 22 23)))	24
5	(cdr ’(cons is short for ans))	(is short for ans)
6	(car (list one two))	UNBOUND-VARIABLE one
7 (6*)	(car (list ’one ’two))	one

3.7 Задание 7

Функция mystery представлена в листинге 6. Результаты вычисления выражений представлены в таблице 3.

Листинг 6 – Функция mystery

```
1     (defun mystery (x) (list (second x) (first x)))
```

Таблица 3 – Выражения и результаты их вычислений задания 6

№	Выражение	Результат
1	(mystery (one two))	UNBOUND-VARIABLE two, one is undefined
2	(mystery one ’two))	UNBOUND-VARIABLE one
(1,2)*	(mystery ’(one two))	(two one)
3	(mystery (last one two))	UNBOUND-VARIABLE one, two
3*	(mystery (last ’(one two)))	(nil two)
4	(mystery free)	UNBOUND-VARIABLE free -> free is not list
4*	(mystery ’(free))	(nil free)