



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

к лабораторной работе №3

*По курсу: «Функциональное и логическое
программирование»*

Тема: «Работа интерпретатора Lisp».

Студент: Якуба Д.В.

Группа: ИУ7-63Б

Преподаватели: Толпинская Н. Б.,

Строганов Ю. В.

Москва, 2021 г.

Практическая часть

Задание 1. Составить диаграмму вычисления следующих выражений:

1. (equal 3 (abs -3));

2. (equal (+ 1 2) 3);

3. (equal (* 4 7) 21);

4. (equal (* 2 3) (+ 7 2));

5. (equal (- 7 3) (* 3 2));

6. (equal (abs (-2 4)) 3);

Решение:

1.

```
(equal 3 (abs -3))
  3 вычисляется к 3
  (abs -3)
    -3 вычисляется к -3
    применяется abs к -3
    3
  применяется equal к 3 и 3
  Т
```

2.

```
(equal (+ 1 2) 3)
  (+ 1 2)
    1 вычисляется к 1
    2 вычисляется к 2
    применяется + к 1 и 2
    3
  3 вычисляется к 3
  применяется equal к 3 и 3
  Т
```

3.

```
(equal (* 4 7) 21)
  (* 4 7)
    4 вычисляется к 4
    7 вычисляется к 7
    применяется * к 4 и 7
    28
  21 вычисляется к 21
  применяется equal к 28 и 21
  Nil
```

4.

```
(equal (* 2 3) (+ 7 2))  
  (* 2 3)  
    2 вычисляется к 2  
    3 вычисляется к 3  
    применяется * к 2 и 3  
    6  
  (+ 7 2)  
    7 вычисляется к 7  
    2 вычисляется к 2  
    применяется + к 7 и 2  
    9  
  применяется equal к 6 и 9  
  Nil
```

5.

```
(equal (- 7 3) (* 3 2))  
  (- 7 3)  
    7 вычисляется к 7  
    3 вычисляется к 3  
    применяется - к 7 и 3  
    4  
  (* 3 2)  
    3 вычисляется к 3  
    2 вычисляется к 2  
    применяется * к 3 и 2  
    6  
  применяется equal к 4 и 6  
  Nil
```

6.

```
(equal (abs (- 2 4)) 3)  
  (abs (- 2 4))  
    (- 2 4)  
      2 вычисляется к 2  
      4 вычисляется к 4  
      применяется - к 2 и 4  
      -2  
    применяется abs к -2  
    2  
  3 вычисляется к 3  
  применяется equal к 2 и 3  
  Nil
```

Задание 2. Написать функцию, вычисляющую гипотенузу прямоугольного треугольника по заданным катетам и составить диаграмму её вычисления.

Решение:

```
(defun findHypo (x y) (sqrt (+ (* x x) (* y y))))
```

Задание 3. Написать функцию, вычисляющую объём параллелепипеда по 3-м его сторонам, и составить диаграмму её вычисления.

Решение:

```
(defun volumeOfParPed (x y z) (* x y z))
```

Задание 4. Каковы результаты вычисления следующих выражений?

- | | |
|-------------------------|-------------------------------------|
| 1. (list `a `b c); | 5. (cons `a `b `c); |
| 2. (cons `a (b c)); | 6. (list `a (b c)); |
| 3. (cons `a `(b c)); | 7. (list a `(b c)); |
| 4. (caddr (1 2 3 4 5)); | 8. (list (+ 1 `(length `(1 2 3)))); |

Решение:

1. (list `a `b c) -> переменная C не связана со значением. Для разрешения ошибки потребуется либо задать для переменной C некоторое значение, либо ввести запрет на вычисление: (list `a `b `c) -> (A B C);

2. (cons `a (b c)) -> функция В не задана, переменная C не связана со значением. Для разрешения ошибки потребуется либо создать функцию В, которая будет принимать единственный аргумент (или их неограниченное количество), и задать для переменной C некоторое значение, либо ввести запрет на вычисление, как это выполнено в следующем примере;

3. (cons `a `(b c)) -> (A B C);

4. (caddr (1 2 3 4 5)) -> неправильный вызов функции, так как название функции не может состоять только из чисел и начинаться с них. Для разрешения ошибки потребуется ввести запрет на вычисление: (caddr `(1 2 3 4 5)) -> 3;

5. (cons `a `b `c) -> неверное количество аргументов, так как функция cons принимает на вход лишь два аргумента. Ошибка может быть разрешена путём удаления одного из аргументов в вызове, например: (cons `a `b) -> A . B;

6. (list `a (b c)) -> функция В не определена, переменная C не связана со значением. Для разрешения ошибки потребуется либо создать функцию В, которая будет принимать единственный аргумент (или их неограниченное количество), и задать для переменной C некоторое значение, либо ввести запрет на вычисление: (list `a `(b c)) -> (A (B C));

7. (list a `(b c)) -> переменная A не связана со значением. Для разрешения ошибки потребуется либо задать некоторое значение для переменной A, либо ввести запрет на вычисление: (list `a `(b c)) -> (A (B C));

8. (list (+ 1 `(length `(1 2 3)))) -> значение (length `(1 2 3)) не является числом. Для разрешения ошибки потребуется убрать запрет на вычисление, так как оно не позволяет выполниться функции length, которая должна вернуть требуемое число: (list (+ 1 (length `(1 2 3)))) -> (4).

Задание 5. Написать функцию `longer_then` от двух списков-аргументов, которая возвращает Т, если первый аргумент имеет большую длину.

Решение:

```
(defun longer_then (firstList secondList)
  (> (length firstList) (length secondList)))
```

Задание 6. Каковы результаты вычисления следующих выражений?

1. `(cons 3 (list 5 6))`;
2. `(cons 3 `(list 5 6))`;
3. `(list 3 `from 8 `gives (-9 3))`;
4. `(+ (length `(1 foo 2 too)) (car `(21 22 23)))`;
5. `(cdr `(cons is short for ans))`;
6. `(car (list one two))`;
7. `(car (list `one `two))`;

Решение:

1. `(cons 3 (list 5 6)) -> (3 5 6)`;
2. `(cons 3 `(list 5 6)) -> (3 LIST 5 6)`;
3. `(list 3 `from 8 `gives (- 9 3)) -> (3 FROM 8 GIVES 6)`;
4. `(+ (length `(1 foo 2 too)) (car `(21 22 23))) -> 25`;
5. `(cdr `(cons is short for ans)) -> (IS SHORT FOR ANS)`, так как введён запрет на вычисление, `cons` будет воспринят как элемент списка, а не команда, при этом он будет являться «головой» списка, а, следовательно, команда `cdr` вернёт «хвост» списка, в который не входит `cons`;
6. `(car (list one two)) ->` переменная `ONE` не связана со значением. Для разрешения ошибки потребуется либо инициализировать значения переменных `one` и `two`, либо указать запрет на их вычисление: `(car (list `one `two)) -> ONE`;
7. `(car (list `one `two)) -> ONE`;

Теоретическая часть

1. Базис Lisp.

В базис языка входят:

- 1) атомы и бинарные узлы;

- 2) функции: car, cdr (селекторы), atom, eq, cons, cond, quote, lambda, eval, label.

2. Классификация функций.

- 1) чистые математические функции (имеют фиксированное количество аргументов и в качестве возврата – единственное значение);
- 2) рекурсивные функции;
- 3) специальные функции – формы (имеют произвольное количество аргументов, либо эти аргументы обрабатываются не все одинаково);
- 4) псевдофункции – функции, эффект которых виден на внешних устройствах;
- 5) функции с вариантными значениями, из которых выбирается одно;
- 6) функции высших порядков – функционалы (используются для построения синтаксически-управляемых программ, в качестве одного из аргументов принимают описание функции).

Классификация базисных функций:

- 1) селекторы: car и cdr;
- 2) конструкторы: cons, list;
- 3) предикаты: atom, null, consp, listp;
- 4) сравнения: eq, eql, equal, equalp.

3. Функции car и cdr.

Функция car предоставляет доступ к «голове» списка. Функция cdr предоставляет доступ к «хвосту списка».

4. Назначение и отличие в работе cons и list.

Функция cons принимает два аргумента и создаёт бинарный узел, первая ячейка которого указывает на первый переданный аргумент, а вторая ячейка – на второй.

Функция list не имеет ограничений по количеству передаваемых ей аргументов. Данная функция создаёт список, элементами которого являются все переданные функции аргументы.

Особенности работы функций:

1) cons не всегда создаёт список, например вызов (cons A B) создаст точечную пару, которая будет представлена бинарным узлом, что может привести к проблемам при рекурсивной обработке, так как будет отсутствовать NIL. Данная особенность работы состоит в том, что cons создаёт список из двух принимаемых аргументов: «голова» и «хвост». Таким образом, если второй передаваемый аргумент не является списком, получается точечная пара.

2) cons работает эффективнее list;

3) list описан с использованием cons;