|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 предмету: «Функциональное и логическое программирование»*

**Тема:** Списки в LISP. Использование стандартных функций.

Преподаватель: Толпинская Н.Б.,

Студент: Мирзоян С.А.,

Группа: ИУ7-65Б

Москва, 2020 г.

Программы в лиспе выглядят как списки. Рассмотрим базовые функции.

* Селекторы:
  1. car – указатель на голову списка
  2. cdr – указатель на хвост списка
* Конструкторы:
  1. cons - Создает списковую ячейку, расставляя указатели на первый и второй аргумент. Не всегда даёт список, но всегда даёт списковую ячейку.
  2. list - всегда создает список (столько списковых ячеек, сколько аргументов).
* Предикаты (возвращающие логическое значение):
  1. atom - возвращает Т, если аргумент является атомом.
  2. listp - является ли списком
  3. consp - является ли точечной парой.
  4. null - пуст ли список
  5. symbolp - является ли символом
  6. numberp - является ли числом
  7. zerop - является ли нулём
  8. oddp, evenp - проверка на четность
  9. < , >
  10. eq - сравнивает два символа по указателям, установлены ли указатели на один объект. Позволяет корректно сравнивать логические константы Т и Nil
  11. eql - используется в стандартных функциях, сравнивает числа. Корректно сравниваются только "одного типа" - 3 и 3, но не 3 и 3.0. Не сравнивает списки.
  12. = - может быть использована для сравнениях разных типов (например 3 и 3.0)
  13. equal - сравнивает списки, но не сравнивает разнотипные числа.
  14. equalp - сравнивает все подряд.

Некоторые функции взаимозаменяемы, но скорость их работы может сильно отличаться. Функция носит частичный характер - на одних аргументах работает корректно, на других нет (например только с числами: < , >)

Первоначально запускается базовая функция eval . Значение символа может быть не только числом, но и другим символом или даже сложной конструкцией.

Также есть функция блокирующая вычисления - quote , также ' .

'(A B C) эквивалентно (quote (A B C)).

Цикл работы LISP-машины:

1. read S-выражение
2. eval S-выражение
3. print
4. GOTO п.1

В любых императивных системах система печатает только по запросу. Лисп работает по указателям, в процессе работу может преобразовывать значения, и на экран всегда выводится последнее вычисленное значение. Не всегда оно может соответствовать тем преобразованиям, которые прошли в памяти.

Схема работы функции eval.

(eval s-выражение)

1. S-выражение атом? –yes GOTO->2 –no GOTO->7
2. S-выражение - Т? –yes GOTO->3 –no GOTO->4
3. печать T \*
4. S-выражение =nil? ->yes GOTO->5 –no GOTO->6
5. печать nil \*
6. значение S-выражение если есть
7. является первым элементом quote? –yes GOTO->8 –no GOTO->9
8. S-выражение без символа квоты \*
9. требует особой обработки? –да GOTO->12 –нет GOTO->10
10. применить eval ко всем элементам S-выражение кроме первого
11. применить первый элемент к остальным \*
12. провести обработку в зависимости от функции
13. применить первый элемент к остальным \*

**Задание 3: Что будет в результате вычисления выражений**

1. (caadr '((blue cube) (red pyramid))) -> RED;
2. (cdar '((abc) (def) (ghi))) -> ();
3. (cadr '((abc) (def) (ghi))) -> (DEF);
4. (caddr '((abc) (def) (ghi))) -> (GHI);

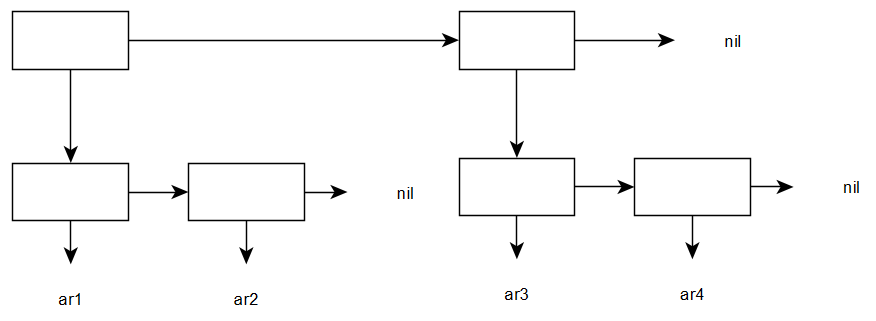
**Задание 4: Напишите результат вычисления выражений**

1. (list 'Fred 'and Wilma) -> The variable WILMA is unbound.
2. (list 'Fred '(and Wilma)) -> (FRED (AND WILMA))
3. (cons Nil Nil) -> (NIL)
4. (cons T Nil) -> (T)
5. (cons Nil T) -> (NIL . T)
6. (list Nil) -> (NIL)
7. (cons (T) Nil)) -> The function T is undefined.
8. (list '(one two) '(free temp)) -> ((ONE TWO) (FREE TEMP))
9. (cons 'Fred '(and Wilma))-> (FRED AND WILMA)
10. (cons 'Fred '(Wilma)) -> (FRED WILMA)
11. (list Nil Nil) -> (NIL NIL)
12. (list T Nil) -> (T Nil)
13. (list Nil T) -> (Nil T)
14. (cons T (list Nil)) -> (T NIL)
15. (list (T) Nil) -> The function T is undefined.
16. (cons '(one two) '(free temp)) -> ((ONE TWO) FREE TEMP)

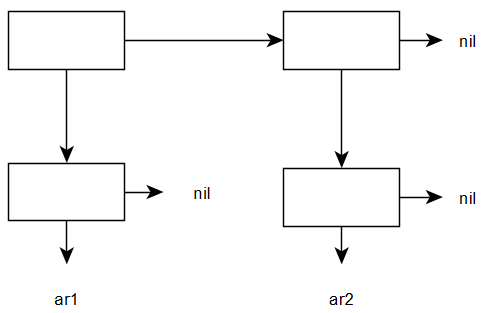
**Задание 5:**

1. Напиcать функцию (f ar1 ar2 ar3 ar4), возвращающую список: ((ar1 ar2)(ar3 ar4)):
   1. (defun f (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3 ar4)))
   2. (defun f (a b c d) (cons `a (cons `b Nil)) (cons `c (cons `d Nil)))
   3. (defun f (a b c d) (cons `(list a b)(list c d))
2. Напиcать функцию (f ar1 ar2), возвращающую: ((ar1) (ar2)):
   1. (defun f (ar1 ar2) (list (list ar1) (list ar2))
   2. (defun f (a b) (cons (cons `a Nil) (cons (cons `b Nil) Nil)))
   3. (defun f (a b c d) ((list (cons `a Nil)(cons `b Nil)))
3. Напиcать функцию (f ar1), возвращающую: (((ar1))):
   1. (defun f (ar1) (list (list (list ar1))))
   2. (defun f (a) (list (list (cons `a Nil))))
   3. (defun f (a) (list (cons (cons a Nil) Nil)))

1.



2.



3.

