|  |  |
| --- | --- |
| 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 №1, 2**

**по курсу “Функциональное и логическое программирование”**

**по теме “Списки в Lisp. Использование стандартных функций”**

|  |  |
| --- | --- |
| Студент: | Уласик Е.А. |
| Группа: | ИУ7-61 |
| Преподаватель: | Толпинская Н.Б. |

*2020 г.*

# Цели и задачи работы

Цель работы – приобрести навыки использования списков и стандартных функций Lisp.

Задачи работы – изучить способ использования списков для фиксации информации, внутреннее представление одноуровневых и структурированных списков, методы их обработки с использованием базовых функций Lisp.

3.Теоретические вопросы

* 1. Базовые элементы языка

Вся информация (программа и данные) в LISP’е представляются с помощью символьных выражений – s-выражений, атомов и точечных пар. Основные элементы языка: атомы, точечные пары, s-выражения, списки. Атомы могут восприниматься как константы и как идентификатор в зависимости от контекста. Атомы могут быть:

* Символы (идентификаторы (набор букв и цифр), которые начинаются с буквы)
* Специальные символы – T, Nil. Используются для обозначения логических констант.
* Самоопределимые атомы, натуральные, дробные и вещественные числа, строки – последовательность символов, заключённые в двойные апострофы.

Точечная пара ::= атом . атом | атом . точечная пара | точечная пара.точечная пара

* 1. Определение списка

Список — это динамическая структура, которая может быть пустой и не пустой. Если не пустой, то состоит из двух элементов: первый (голова) – любая структура, второй (хвост) – список.

* 1. Функция Quate и ‘

Функция QUOTE производит блокировку вычисления выражения. Использование апострофа ‘ просто сокращённое обозначение функции QUOTE.

* 1. Представление списков в памяти

Списки состоят из списочных ячеек. Списочная ячейка состоит из двух частей, полей CAR и CDR. Каждое из полей содержит указатель. Указатель может ссылаться на другую списочную ячейку или некоторый другой Lisp объект.

* 1. Базис Lisp

Базис – базовые структуры, атомы и функции;

* 1. Классификация функций
* Чистые математические функции (фиксированное число аргументов и один результат);
* Специальные формы (функции с произвольным числом аргументов или аргументы могут быть не вычислены);
  1. Как выполняются функции CAR и CDR, какие результаты они вернут в разных случаях?

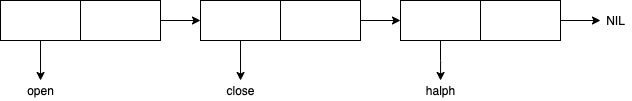
Базовыми функциями доступа к данным в Lisp являются функции CAR и CDR.

* CAR – в качестве аргумента принимает список (1 аргумент) и возвращает первый элемент списка, если список не пустой и Nil – иначе;
* CDR – в качестве аргумента принимает список (1 аргумент) и возвращает. список, состоящий из всех элементов, кроме первого, если список содержит хотя бы два элемента, иначе – Nil.

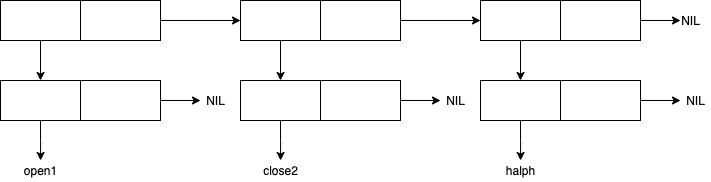
1. Практическая часть

Задание 1. Представить следующие списки в виде списочных ячеек.

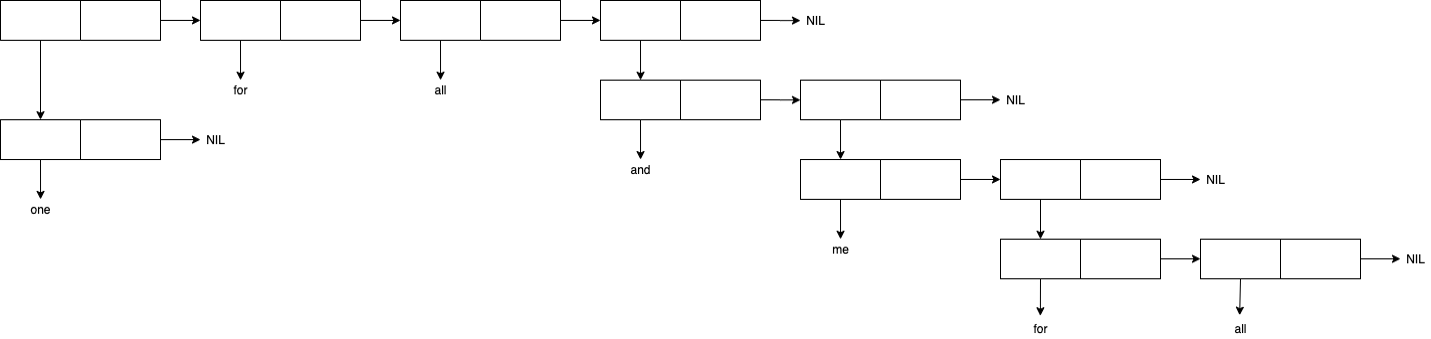
* 1. ‘(open close halph)



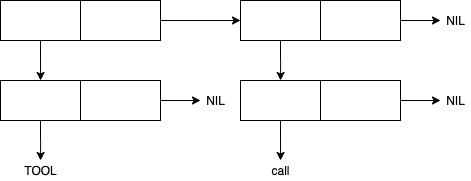
* 1. ‘((open1)(close2)(halph3))



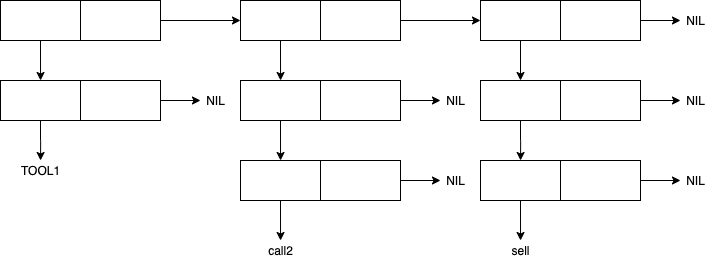
* 1. ‘((one) for all (and(me(for you))))



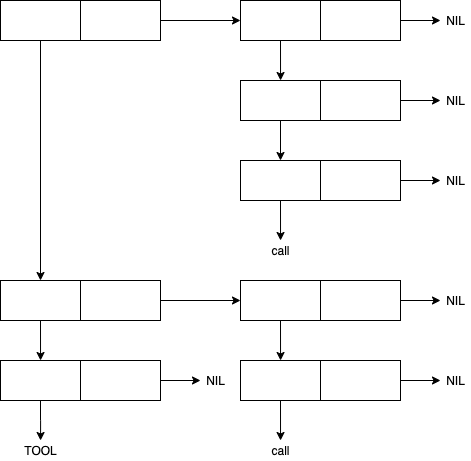
* 1. ‘((TOOL)(call))



* 1. ‘((TOOL1)((call2))((sell)))



* 1. ‘(((TOOL)(call))((sell)))



Задание 2. Используя только функции CAR и CDR, написать выражения, возвращающие: пусть задан список (a b c d e f), тогда

* 1. Второй элемент заданного списка

(CAR(CDR(a b c d e f))

* 1. Третий элемент заданного списка

(CAR(CDR(CDR(a b c d e f))))

* 1. Четвёртый элемент заданного списка

(CAR(CDR(CDR(CDR(a b c d e f)))))

Задание 3. Что будет в результате вычисления выражения?

3.1 (CAADR ‘((blue cube) (red pyramid)))

Результат: red

3.2 (CDAR ‘((abc)(def)(ghi)))

Результат: Nil

3.3 (CADR ‘((abc)(def)(ghi)))

Результат: def

3.4 (CADDR ‘((abc)(def)(ghi)))

Результат: (ghi)

Задание 4. Напишите результат вычисления выражений:

4.1 (list ‘Fred ‘and Wilma)

Результат: The variable WILMA is unbound.

[Condition of type UNBOUND-VARIABLE]

4.2 (list ‘Fred ‘(and Wilma))

Результат: (Fred (and Wilma))

4.3 (cons Nil Nil)

Результат: (Nil)

4.4 (cons T Nil)

Результат: (T)

4.5 (cons Nil T)

Результат: (Nil . T)

4.6 (list Nil)

Результат: (Nil)

4.7 (cons (T) Nil)

Результат: The function COMMON-LISP:T is undefined.

4.8 (list ‘(one two) ‘(free temp))

Результат: ((one two) (free temp))

4.9 (cons ‘Fred ‘(and Wilma))

Результат: (Fred and Wilma)

4.10 (cons ‘Fred ‘(Wilma))

Результат: (Fred Wilma)

4.11 (list Nil Nil)

Результат: (Nil Nil)

4.12 (list T Nil)

Результат: (T Nil)

4.13 (list Nil T)

Результат: (Nil T)

4.14 (cons T (list Nil))

Результат: (T Nil)

4.15 (list (T) Nil)

Результат: The function COMMON-LISP:T is undefined.

4.16 (cons ‘(one two) ‘(free temp))

Результат: ((one two) free temp)

Задание 5.

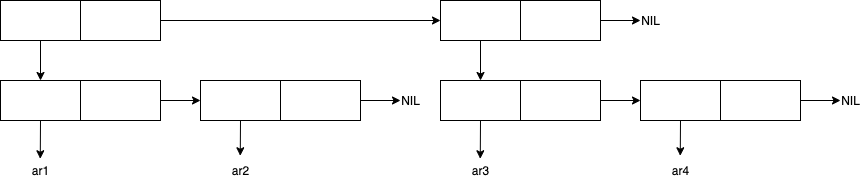
5.1 Написать функцию (f ar1 ar2 ar3 ar4), возвращающую список:

((ar1 ar2)(ar3 ar4))

Ответ: (defun f (ar1 ar2 ar3 ar4)(list (list ar1 ar2)(list ar3 ar4)))

или (defun f5(ar1 ar2 ar3 ar4) (cons (cons ar1 (cons ar2 nil))(cons (cons ar3 (cons ar4 nil)) nil)))

Результат:

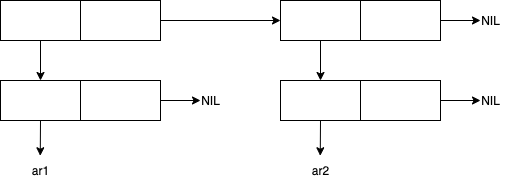


5.2 Написать функцию (f ar1 ar2), возвращающую ((ar1)(ar2))

Ответ: (defun f(ar1 ar2)(list (list ar1)(list ar2)))

или (defun f4(ar1 ar2)(cons (cons ar1 nil) (cons (cons ar2 nil) nil)))

Результат:



5.3 Написать функцию (f ar1), возвращающую (((ar1)))

Ответ: (defun f6(ar1) (list (list (list ar1))))

или (defun f3(ar1)(cons(cons(cons ar1 nil) nil) nil))

Результат:

