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

по лабораторной работе № 1

по курсу «Функциональное и логическое программирование»

на тему: «Списки в Lispe. Использование стандартных функций»

Студент ИУ7-61Б _____ Д. В. Шубенина
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Н. Б. Толпинская
(Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Ю. В. Строганов
(Подпись, дата) (И. О. Фамилия)

*2024 г.*

# СОДЕРЖАНИЕ

# 1 Теоретические вопросы

## 1.1 Элементы языка: определение, синтаксис, представление в памяти.

Вся информация (данные и программы) в Lisp представляется в виде символьных выражений — S-выражений. По определению

`S-выражение ::= <атом> | <точечная пара>.`

Атомы:

— символы (идентификаторы) — синтаксически — набор литер (букв и цифр), начинающихся с буквы;

— специальные символы — {T, Nil} (используются для обозначения логических констант);

— самоопределимые атомы — натуральные числа, дробные числа (например 2/3), вещественные числа, строки — последовательность символов, заключенных в двойные апострофы (например “abc”);

Более сложные данные — списки и точечные пары (структуры) строятся из унифицированных структур — блоков памяти — бинарных узлов. Определения:

`Точечные пары ::= (<атом>.<атом>) | (<атом>.<точечная пара>) | (<точечная пара>.<атом>) | (<точечная пара>.<точечная пара>;`

`Список ::= <пустой список> | <непустой список>,`
где `<пустой список> ::= () | Nil,`
`<непустой список> ::= (<первый элемент>.<хвост>),`
`<первый элемент> ::= <S-выражение>,`
`<хвост> ::= <список>.`

Синтаксически любая структура (точечная пара или список) заключается в круглые скобки; `( A.B )` — точечная пара, `( A )` — список из одного элемента; пустой список изображается как `Nil` или `()`.

`( A.( B.( C.( D () ))))`, допустимо изображение списка последовательностью атомов, разделенных пробелами — `( A B C D )`.

Элементы списка могут, в свою очередь, быть списками (любой список заключается в круглые скобки), например: `( A ( B C ) ( D (E) ))`. Таким образом, синтаксически наличие скобок является признаком структуры — списка или точечной пары.

Любая непустая структура Lisp в памяти представляется списковой ячейкой, хранящей два указателя: на голову (первый элемент) и хвост — все остальное.

## 1.2 Особенности языка Lisp. Структура программы. Символ апостроф.

Особенности языка Lisp:

— бестиповый язык;

— символьная обработка данных;

— любая программа может интерпретироваться как функция с одним или несколькими аргументами;

— автоматизированное динамическое распределение памяти, которая выделяется блоками;

— программа может быть представлена как данные, то есть программа может изменять саму себя.

Символ апостроф — сокращенное обозначение функции `quote`, блокирующей вычисление своего аргумента.

## 1.3 Базис языка Lisp. Ядро языка.

Базис языка — минимальный набор обозначений, в которые можно свести все правильные (то есть вычислимые) формулы системы.

Базис Lisp образуют:

— атомы;

— структуры;

— базовые функции;

— базовые функционалы.

Ядро языка — базис, дополненный наиболее употребимыми функциями языка.

# 2 Практические задания

## 2.1 Задание 1

Представить следующие списки в виде списочных ячеек:

1) `‘(open close halph)`

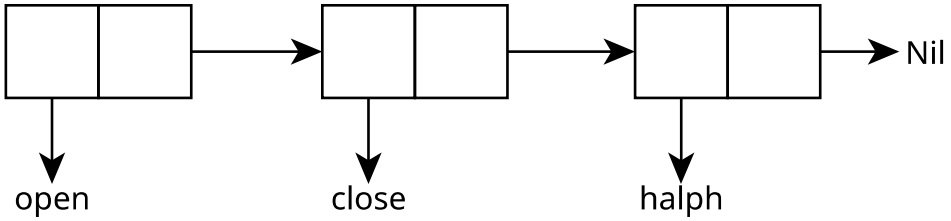


Рисунок 2.1

2) `‘((open1) (close2) (halph3))`

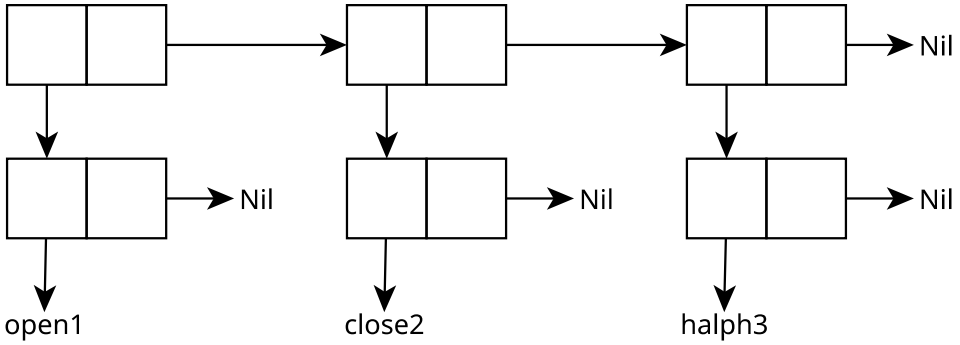


Рисунок 2.2

3) `‘((one) for all (and (me (for you))))`

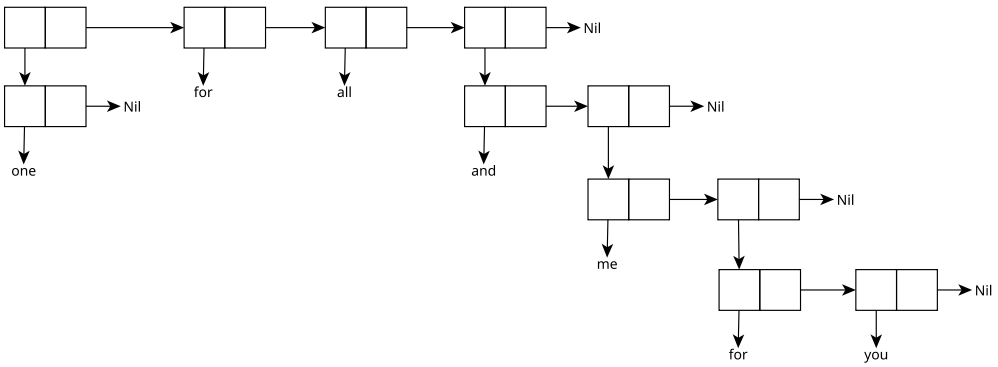


Рисунок 2.3

4) `‘((TOOL) (call))`

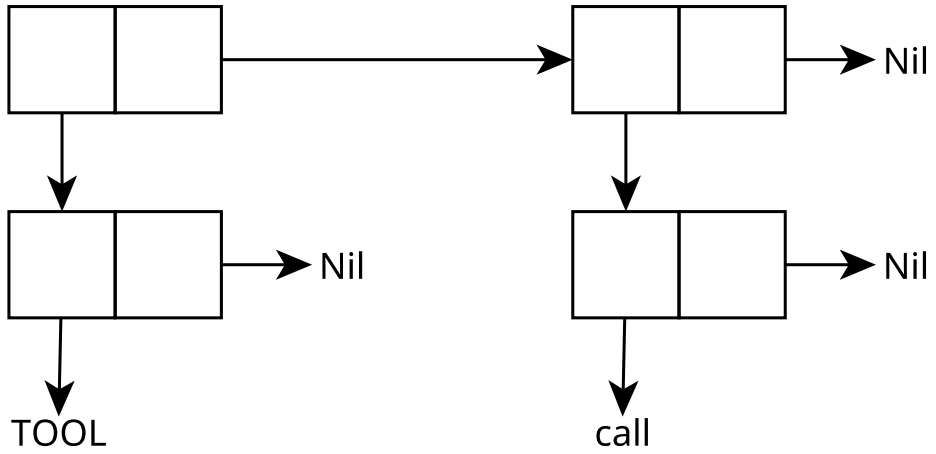


Рисунок 2.4

5) `‘((TOOL1) ((call2)) ((sell)))`

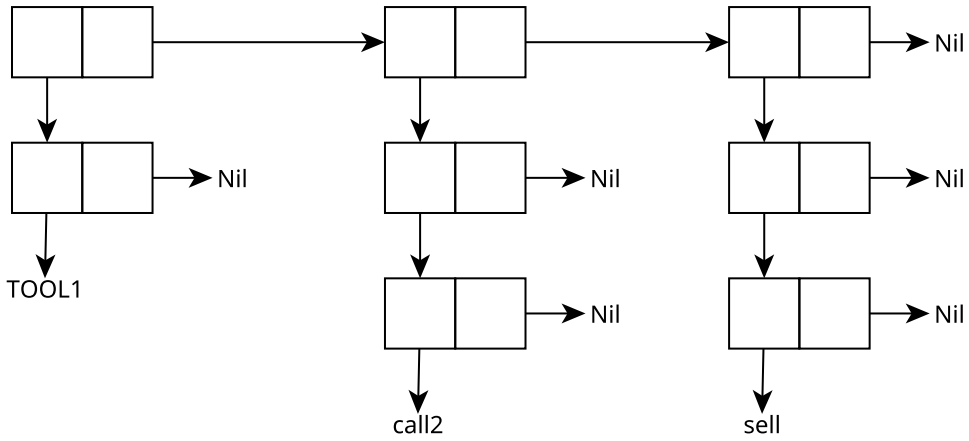


Рисунок 2.5

6) `‘(((TOOL) (call)) ((sell)))`

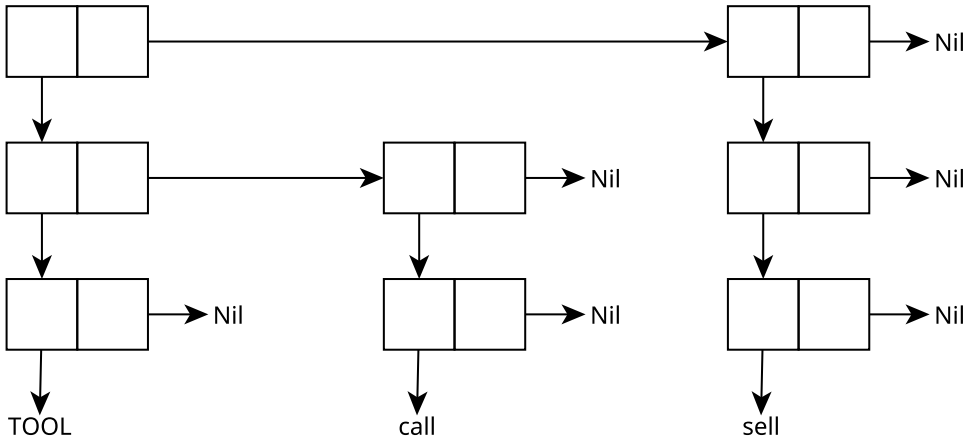


Рисунок 2.6

## 2.2 Задание 2

Используя только функции CAR и CDR, написать выражения, возвращающие элементы заданного списка:

1) второй;

```
(car (cdr '(A B C D)))
(cadr '(A B C D))
```

2) третий;

```
(car (cdr (cdr '(A B C D))))
(caddr '(A B C D))
```

3) четвертый.

```
(car (cdr (cdr (cdr '(A B C D)))))
(cadddr '(A B C D))
```

## 2.3 Задание 3

Что будет в результате вычисления выражений?

1) `(CAADR '((blue cube) (red pyramid)))`

   red

2) `(CDAR '((abc) (def) (ghi)))`

   Nil

3) `(CADR '((abc) (def) (ghi)))`

(def)

4) `(CADDR '((abc) (def) (ghi)))`

(ghi)

## 2.4 Задание 4

Напишите результат вычисления выражений и объясните, как он получен:

| Выражение | Результат вычисления |
|---|---|
| `(list 'Fred 'and 'Wilma)` | (FRED AND WILMA) |
| `(list 'Fred '(and Wilma))` | (FRED (AND WILMA)) |
| `(cons Nil Nil)` | (NIL) |
| `(cons T Nil)` | (T) |
| `(cons Nil T)` | (NIL . T) |
| `(list Nil)` | (NIL) |
| `(cons '(T) Nil)` | ((T)) |
| `(list '(one two) '(free temp))` | ((ONE TWO) (FREE TEMP)) |
| `(cons 'Fred '(and Wilma))` | (FRED AND WILMA) |
| `(cons 'Fred '(Wilma))` | (FRED WILMA) |
| `(list Nil Nil)` | (NIL NIL) |
| `(list T Nil)` | (T NIL) |
| `(list Nil T)` | (NIL T) |
| `(cons T (list Nil))` | (T NIL) |
| `(list '(T) Nil)` | ((T) NIL) |
| `(list '(one two) '(free temp))` | ((ONE TWO) FREE TEMP) |

## 2.5 Задание 5

Написать лямбда-выражение и соответствующую функцию:

1) функция (f ar1 ar2 ar3 ar4), возвращающая ((ar1 ar2) (ar3 ar4));

```
1 ((lambda (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3
    ar4))) 'A 'B 'C 'D)
```

```
(defun f (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3
   ar4)))
```

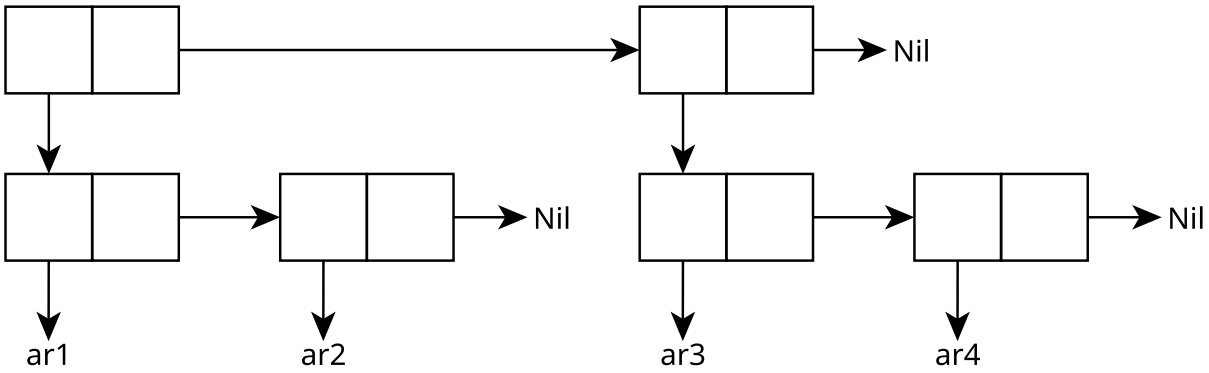


Рисунок 2.7

2) функция (f ar1 ar2), возвращающая ((ar1)(ar2));

```
((lambda (ar1 ar2) (list (list ar1) (list ar2))) 1 2)
(defun f (ar1 ar2) (list (list ar1) (list ar2)))
```

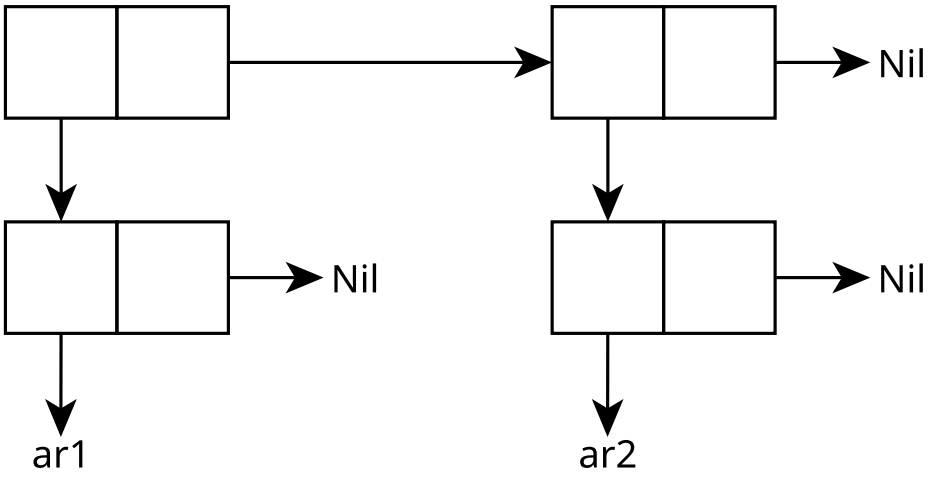


Рисунок 2.8

3) функция (f ar1), возвращающая (((ar1))).

```
((lambda (ar1) (list (list (list ar1)))) "abc")
(defun f (ar1) (list (list (list ar1))))
```

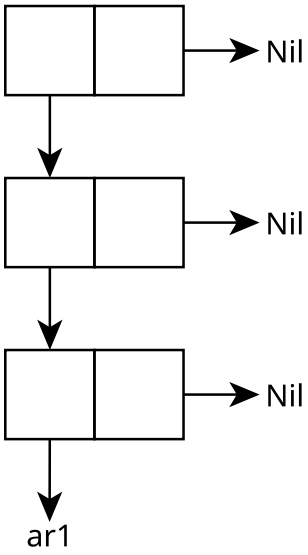


Рисунок 2.9