

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
по дисциплине «Функциональное и логическое
программирование»

Тема Списки в Lisre. Использование стандартных функций.

Студент Калашников С.Д.

Группа ИУ7-63Б

Преподаватель Толпинская Н.Б., Строганов Ю.В.

Москва, 2023

СОДЕРЖАНИЕ

1	Ответы на вопросы.	3
1.1	Базис лиспа.	6
1.2	Функции car и cdr	7
1.3	Отличие list и cons	7
2	Практическая часть	9
2.1	Задание 1	9
2.2	Задание 2	11
2.3	Задание 3	11
2.4	Задание 4	11
2.5	Задание 5	13

1 Ответы на вопросы.

1. Элементы языка.

Элементами языка являются атомы и точечные пары.

Атомы представляю из себя:

1. Символы - синтаксически представляется как набор букв и цифр, начинающийся с буквы.
2. Специальные символы - {T, Nil}.
3. Самоопределимые атомы - натуральные, дробные и вещественные числа, а также строки, заключенные в двойные апострофы.

Атомы обычно выглядит как последовательность букв или цифр.

Примеры атомов:

```
1 В
2 САТЯАтом
3 123ВотЭтоТожеАтом
4
5 123
6 Т
7 Nil
8 2/3
9 "abc"
```

Точечная пара - (A . B). Строится с помощью бинарных узлов.

```
1 Точечнаяпара
2 ::= атоматом(<>.<>) | атомточечная
3 (<>.< пара>) | точечная
4 (< параатом>.<>) | точечная
5 (< параточечная>.< пара>)
```

Пример точечной пары:

$(A . (B . (C . (D . Nil))))$

Облегченная форма записи:

$(A B C D)$

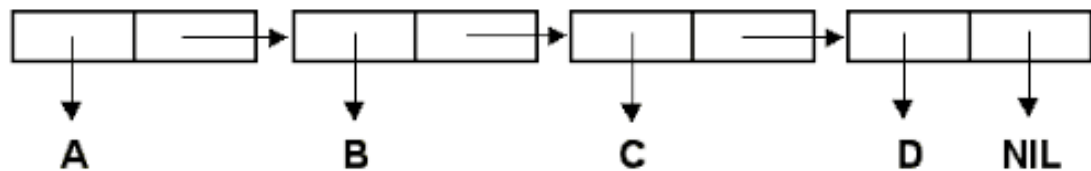


Рис. 1.1 – Представление в памяти $(A B C D)$.

Представление в памяти:

1. $(A . B)$

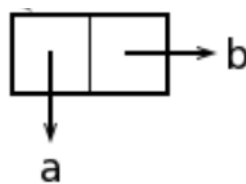


Рис. 1.2 – Представление в памяти $(A . B)$.

2. $(A B)$ - экономия памяти, но проблема при рекурсивной обработке (т.к. не сможем идентифицировать конец, как Nil)

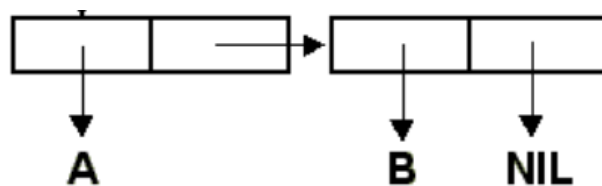


Рис. 1.3 – Представление в памяти $(A B)$.

Свыражение ::= атом<> | точная< пара>

Список является частым случаем S-выражения.

S-выражение представлено

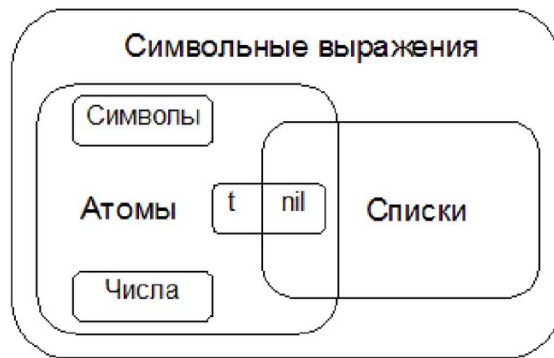


Рис. 1.4 – S-выражение

Список - динамическая структура данных, которая может быть пустой или непустой. Если она не пустая, то состоит из двух элементов:

1. Головы - любая структура.
2. Хвоста - список.

Список представляет из себя заключенную в скобки последовательность из атомов, разделенных пробелами, или списков. Любой список является программой - его нужно вычислять.

Примеры списков:

1	(A B C)
2	(1 2 3)
3	((A B) (C D))

2. Синтаксис.

Lisp является регистронезависимым языком.

Универсальным разделителем, между атомами, является пробел. В начальных версиях была предложена запятая, но она не прижилась.

Наличие скобок является признаком структуры - списка или точечной пары.

Специальные символы:

1. **T** - Константа. обозначает логическое значение "истина". Истинным значением является все, что отличное от Nil.
2. **Nil** - "ложь". Также обозначает пустой список. Записи nil и () эквивалентны. Являются синтаксисом пустого списка

Любая структура заключается в круглые скобки.

(A . B) - точечная пара.

(A) - список из одного элемента.

() или Nil - пустой список.

Одноуровневый список:

```
1 (A B C D)
```

Структурированный список:

```
1 (A (B C) (D E))
```

3. Как воспринимается символ апостроф

Символ апостроф - синоним quote.

quote - блокирует вычисление своего аргумента. В качестве своего значения выдаёт сам аргумент, не вычисляя его. Перед константами - числами и атомами T, Nil можно не ставить апостроф.

Пример использования quote:

```
1 (quote (car (A B C))) => (car (A B C))
```

Вычисление начинается с внешней функции quote, которая возвращает аргумент в неизмененном виде.

1.1 Базис лиспа.

Базис - минимальный набор средств для решения любой задачи.

Базис:

1) атомы и бинарные узлы;

2) atom, eq, cons, car, cdr, cond, quote, eval.

atom проверяет, является ли объект, переданный в качестве аргумента, атомом.

```
1 (atom 'a) ;; t
2 (atom '(a b c)) ;; nil
```

eq проверяет идентичность двух символов.

```
1 (eq 'a 'b) ;; nil
2 (eq 'a 'a) ;; t
```

cond - сокращение от англ. *condition* – условие. Не имеет фиксированного количества аргументов. Каждый аргумент - это список, голова которого рассматривается как условие, и если оно истинно, то результатом будет хвост рассматриваемого списка.

```
1 (cond ((eq 'A 'B) 'are_equal)
2 (T 'not_equal)) ;; NOT_EQUAL
```

eval - выполняет двойное вычисление своего аргумента.

```
1 (eval (cons (quote car) (quote ('(A B))))) => A
2 |-----(car '(A B))-----|
```

1.2 Функции **car** и **cdr**

car и **cdr** - базовые функции доступа к данным.

car - принимает точечную пару или список и возвращает голову (первый элемент).

cdr - принимает точечную пару или список и возвращает хвост (все элементы, кроме первого).

1.3 Отличие **list** и **cons**

cons - имеет фиксированное количество аргументов (два). В случае, когда аргументами являются атомы создает точечную пару. В случае, когда первый аргумент атом а второй список, атом становится головой, а второй аргумент (список) становится хвостом.

```
1 (cons 'a 'b) ;; (A . B)
2 (cons 'a '(a b c)) ;; (A A B C)
3 (cons '(a c) '(b d)) ;; ((A C) B D)
4 (cons 'a 'v 'd) ;; Error (invalid number of arguments: 3)
```

list - не имеет фиксированное количество аргументов. Создает список, у которого голова - это первый аргумент, хвост - все остальные аргументы.

```
1 (list 'a 'b) ;; (A B)
2 (list 'a 'b 'v '(c d) 'd) ;; (A B V (C D) D)
```

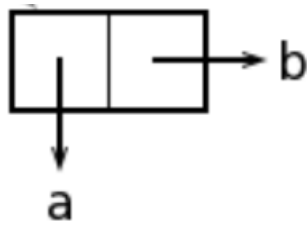


Рис. 1.5 – Результат cons.

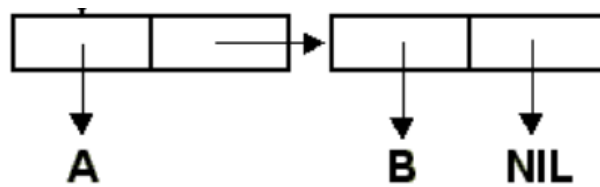


Рис. 1.6 – Результат list.

cons - имеет фиксированное число аргументов и более экономный по памяти.

Ядро - основные действия, которые наиболее часто используются. Ядро шире, чем базис.

2 Практическая часть

2.1 Задание 1

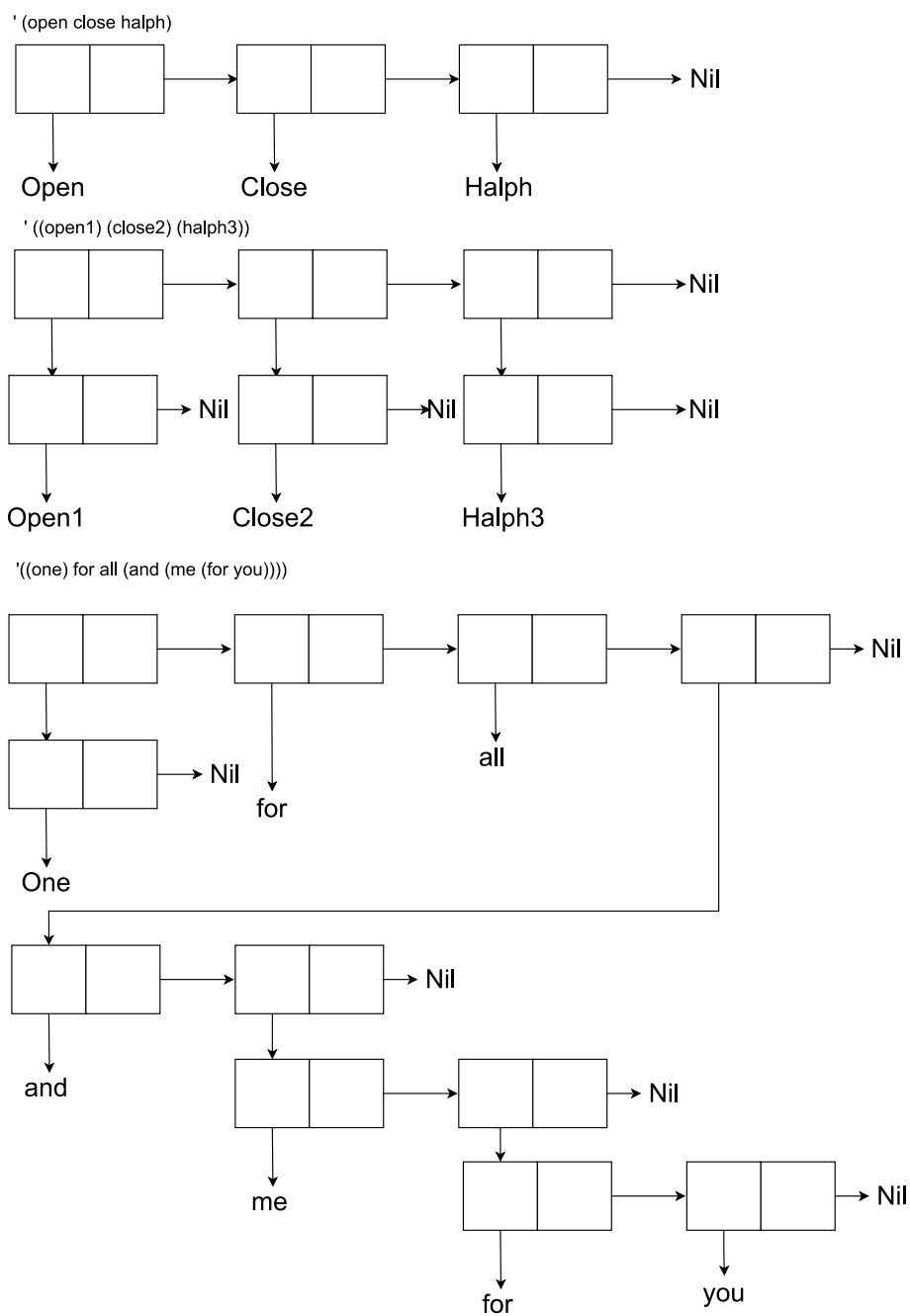
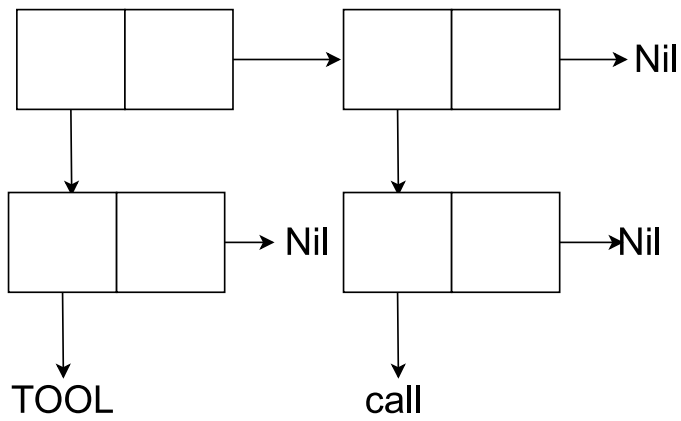
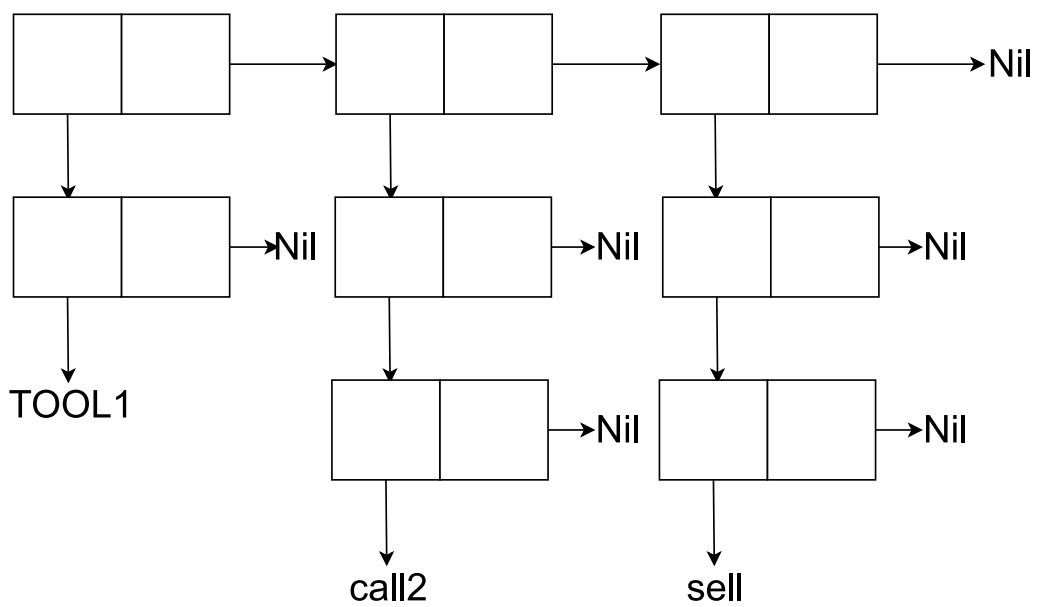


Рис. 2.1 – Часть 1

' ((TOOL) (call))



' ((TOOL1) ((call2)) ((sell)))



' (((TOOL) (call)) ((sell)))

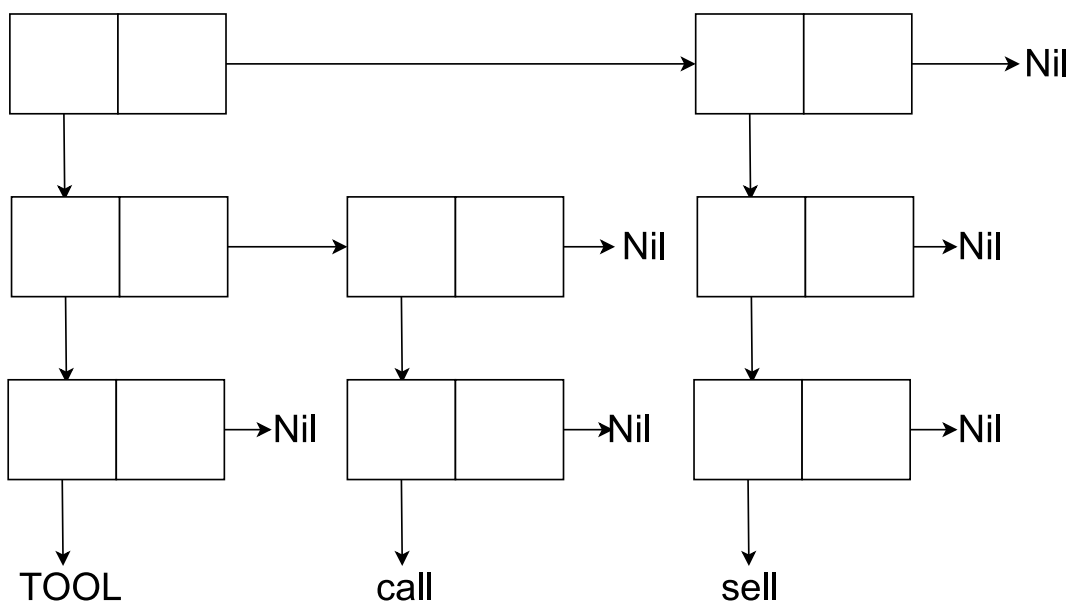


Рис. 2.2 – Часть 2

2.2 Задание 2

Листинг 2.1 — Выражение 1

```
1 (CAR (CDR '(1 2 3 4 5 6)))
```

Листинг 2.2 — Выражение 2

```
1 (CAR (CDR (CDR '(1 2 3 4 5 6))))
```

Листинг 2.3 — Выражение 3

```
1 (CAR (CDR (CDR (CDR '(1 2 3 4 5 6)))))
```

2.3 Задание 3

Листинг 2.4 — Выражение 1

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

Листинг 2.5 — Выражение 2

```
1 (CDAR ' ((abc) (def) (ghi))) ;= nil
```

Листинг 2.6 — Выражение 3

```
1 (CADR ' ((abc) (def) (ghi))) ;= def
```

Листинг 2.7 — Выражение 4

```
1 (CADDR ' ((abc) (def) (ghi))) ;= ghi
```

2.4 Задание 4

Листинг 2.8 — Выражение 1

```
1 (list 'Fred 'and 'Wilma) ;(FRED AND WILMA)
```

Листинг 2.9 — Выражение 2

```
1 (list 'Fred '(and Wilma)) ;(FRED (AND WILMA))
```

Листинг 2.10 — Выражение 3

```
1 (cons Nil Nil) ;(NIL)
```

Листинг 2.11 — Выражение 4

```
1 (cons T Nil) ;(T)
```

Листинг 2.12 — Выражение 5

```
1 (cons Nil T) ;(NIL . T)
```

Листинг 2.13 — Выражение 6

```
1 (list Nil) ;(NIL)
```

Листинг 2.14 — Выражение 7

```
1 (cons ' (T) Nil) ;((T))
```

Листинг 2.15 — Выражение 8

```
1 (list ' (one two) ' (free temp)) ;((ONE TWO) (FREE TEMP))
```

Листинг 2.16 — Выражение 9

```
1 (cons 'Fred '(and Wilma)) ;(FRED AND WILMA)
```

Листинг 2.17 — Выражение 10

```
1 (cons 'Fred '(Wilma)) ;(FRED WILMA)
```

Листинг 2.18 — Выражение 11

```
1 (list Nil Nil) ;(NIL NIL)
```

Листинг 2.19 — Выражение 12

```
1 (list T Nil) ;(T NIL)
```

Листинг 2.20 — Выражение 13

```
1 (list Nil T) ;(NIL T)
```

Листинг 2.21 — Выражение 14

```
1 (cons T (list Nil)) ;(T NIL)
```

Листинг 2.22 — Выражение 15

```
1 (cons '(one two) '(free temp)) ;((ONE TWO) FREE TEMP)
```

Листинг 2.23 — Выражение 16

```
1 (list '(T) Nil) ;((T) NIL)
```

2.5 Задание 5

Листинг 2.24 — Выражение 1

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

Листинг 2.25 — Выражение 2

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

Листинг 2.26 — Выражение 3

```
1 (defun f (ar1) (list (list (list ar1))))
```