



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
(Группа)

(Подпись, дата)

Постнов С. А.
(Фамилия И. О.)

Преподаватель

(Подпись, дата)

Толпинская Н. Б.
(Фамилия И. О.)

Преподаватель

(Подпись, дата)

Строганов Ю. В.
(Фамилия И. О.)

2024 г.

СОДЕРЖАНИЕ

1	Теоретические вопросы	3
1.1	Элементы языка: определение, синтаксис, представление в памяти.	3
1.2	Особенности языка Lisp. Структура программы. Символ апостроф.	4
1.3	Базис языка Lisp. Ядро языка.	4
2	Практические задания	6
2.1	Задание 1	6
2.2	Задание 2	6
2.3	Задание 3	7
2.4	Задание 4	7
2.5	Задание 5	8

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

$$\text{S-выражение} ::= \langle \text{атом} \rangle \mid \langle \text{точечная пара} \rangle.$$

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:

$$\text{Точечные пары} ::= (\langle \text{атом} \rangle . \langle \text{атом} \rangle) \mid (\langle \text{атом} \rangle . \langle \text{точечная пара} \rangle) \mid (\langle \text{точечная пара} \rangle . \langle \text{атом} \rangle) \mid (\langle \text{точечная пара} \rangle . \langle \text{точечная пара} \rangle);$$
$$\text{Список} ::= \langle \text{пустой список} \rangle \mid \langle \text{непустой список} \rangle,$$

где $\langle \text{пустой список} \rangle ::= () \mid Nil,$

$$\langle \text{непустой список} \rangle ::= (\langle \text{первый элемент} \rangle . \langle \text{хвост} \rangle),$$
$$\langle \text{первый элемент} \rangle ::= \langle \text{S-выражение} \rangle,$$
$$\langle \text{хвост} \rangle ::= \langle \text{список} \rangle.$$

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

На рисунке 2.1 представлены списки в виде списочных ячеек:

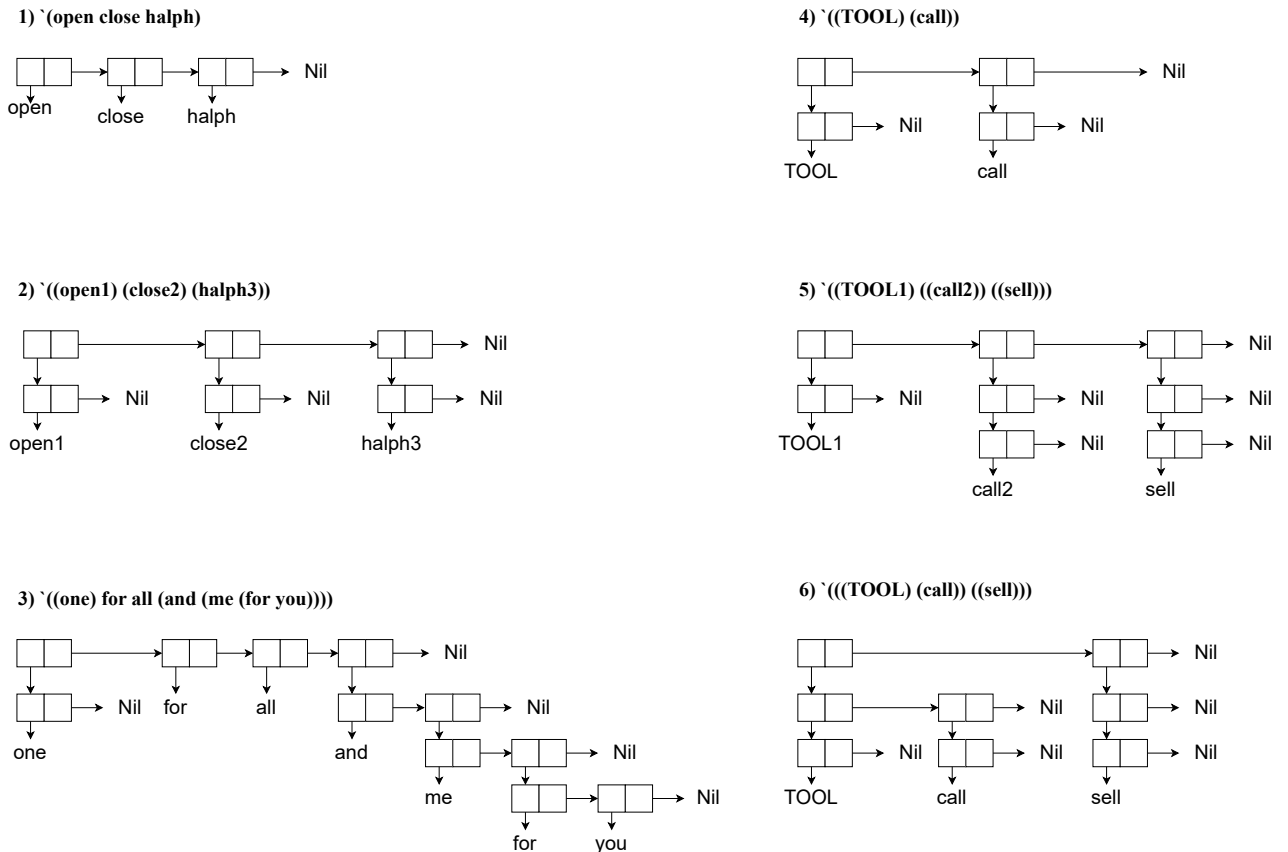


Рисунок 2.1 – Списки в виде списочных ячеек

2.2 Задание 2

В листинге 2.1 представлены выражения, написанные с использованием функций CAR и CDR и возвращающие 2, 3 и 4 элемента списка.

Листинг 2.1 – Выражения, возвращающие 2, 3 и 4 элемента списка

```
1 | (car (cdr '(A B C D)))           ; Второй элемент списка
2 | (cadr '(A B C D))
3 |
4 | (car (cdr (cdr '(A B C D))))    ; Третий элемент списка
5 | (caddr '(A B C D))
6 |
7 | (car (cdr (cdr (cdr '(A B C D))))) ; Четвертый элемент списка
8 | (caddr '(A B C D))
```

2.3 Задание 3

Результаты 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

В листинге 2.2 представлен результат вычисления выражений:

Листинг 2.2 – Результат вычисления выражений

```
1 (list 'Fred 'and 'Wilma) -> (FRED AND WILMA)
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) -> ((T))
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) -> ((T) NIL)
16 (cons '(one two) '(free temp)) -> ((ONE TWO) FREE TEMP)
```

2.5 Задание 5

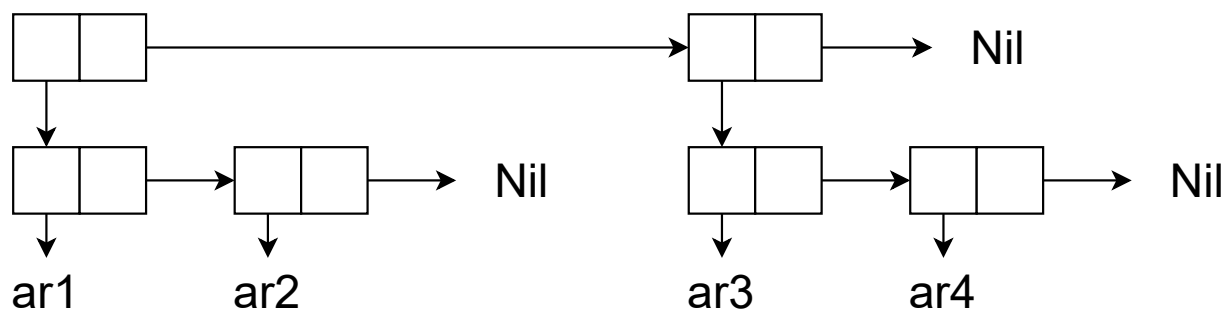
В листинге 2.3 представлены лямбда-выражения и соответствующие функции:

Листинг 2.3 – Лямбда-выражения и соответствующие функции

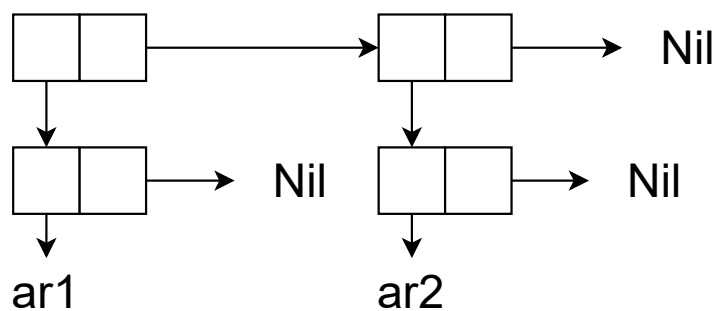
```
1 ; (f ar1 ar2 ar3 ar4) -> ((ar1 ar2) (ar3 ar4))
2 (defun print1 (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3
   ar4)))
3 (defun print1 (ar1 ar2 ar3 ar4) (cons (cons ar1 (cons ar2 Nil))
   (cons (cons ar3 (cons ar4 Nil)) Nil)))
4 ((lambda (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3 ar4)))
   3 2 4 5)
5 ((lambda (ar1 ar2 ar3 ar4) (cons (cons ar1 (cons ar2 Nil)) (cons
   (cons ar3 (cons ar4 Nil)) Nil))) 3 2 4 5)
6
7 ; (f ar1 ar2) -> ((ar1) (ar2))
8 (defun print2 (ar1 ar2) (list (list ar1) (list ar2)))
9 (defun print2 (ar1 ar2) (cons (cons ar1 Nil) (cons (cons ar2
   Nil) Nil)))
10 ((lambda (ar1 ar2) (list (list ar1) (list ar2))) 3 2)
11 ((lambda (ar1 ar2) (cons (cons (cons ar1 Nil) (cons ar2 Nil))
   Nil) 3 2))
12
13 ; (f ar1) -> (((ar1)))
14 (defun print3 (ar1) (list (list (list ar1))))
15 (defun print3 (ar1) (cons (cons (cons ar1 Nil) Nil) Nil))
16 ((lambda (ar1) (list (list (list ar1)))) 2)
17 ((lambda (ar1) (cons (cons (cons ar1 Nil) Nil) Nil)) 2)
```


На рисунке 2.2 представлены результаты в виде списочных ячеек:

1) (f ar1 ar2 ar3 ar4) -> ((ar1 ar2) (ar3 ar4))



2) (f ar1 ar2) -> ((ar1) (ar2))



3) (f ar1) -> (((ar1)))

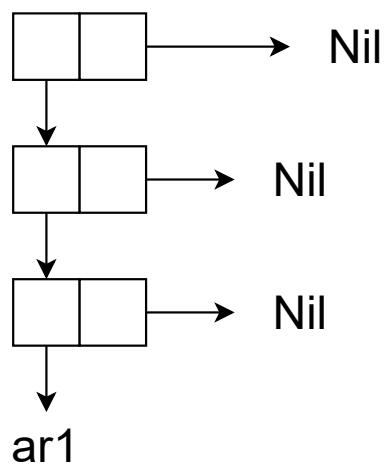


Рисунок 2.2 – Результаты в виде списочных ячеек