**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»

**Тема** Списки в Lisp. Использование стандартных функций»

**Студент** Виноградов А. О.

**Группа** ИУ7-66Б

**Оценка (баллы)**

**Преподаватели** Толпинская Н. Б., Строганов Ю. В.

Москва — 2023 г.

# Оглавление

# Введение

Цель работы: приобрести навыки использования стандартных функций Lisp.

Задачи работы:

- изучить списки LISP как метод фиксации информации;
- изучить методы обработки списков с использованием базовых функций Lisp.

# Глава 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:

$$S\text{-выражение} ::= <\text{атом}>|<\text{точечная пара}>. \tag{1.1}$$

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 $\frac{2}{3}$), вещественные числа, строки – последовательность символов, заключенных в двойные апострофы (например “abc”).

Точечная пара:

<точечная пара> ::= (<атом>.<атом>)|(<атом>.<точечная пара>)|
(<точечная пара>.<атом>)|
(<точечная пара>.<точечная пара>);
<список> ::= <пустой список>|<непустой список>
<пустой список> ::= ()|$Nil$
<непустой список> ::= (<первый элемент>.<хвост>)
<первый элемент> ::= <S-выражение>
<хвост> ::= <список>

Точечная пара вида (A.B) представлена в памяти в виде двух идущих друг за другом адресов: первый указывает на A, второй — на B.

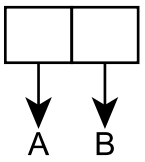


Рисунок 1.1 – Представление в памяти точечной пары (A.B)

Список представлен в памяти в виде пар ячеек, называемых списочными ячейками(пример на рисунке 1.2). В каждой такой паре первое значение является адресом соответствующего элемента списка, второе значение — адресом следующей пары списочной ячейки. Вторым значением списочной ячейки, соответствующей последнему элементу списка, является Nil.

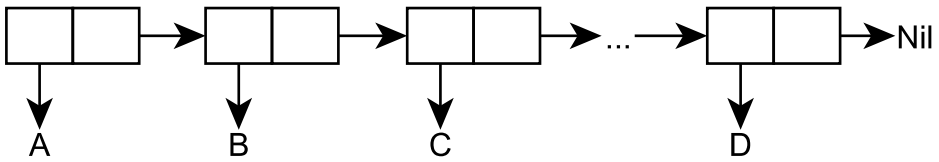


Рисунок 1.2 – Представление в памяти списка вида (A B C ... D)

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

Lisp — бестиповый язык программирования. Все структуры данных, которые в нём поддерживаются, являются списками. По умолчанию, любой список расценивается как вызов функции. При чем первый элемент списка интерпретируется как сама функция, а все последующие элементы — как ее аргументы.

Программа на языке Lisp состоит из единственного S-выражения, результат вычисления которого является результатом работы программы. Вызов функции quote отключает исчисление S-выражений. Символ апостроф — сокращение вызова функции quote, сокращающее количество символов в программе.

## 1.3 Базис языка Lisp. Ядро языка.

Базисные функции Lisp:

1) *quote* — блокировка вычислений;

2) *eval* — интерпретатор;

3) *car* — возвращает голову списка;

4) *cdr* — возвращает хвост списка;

5) *cons* — бинарная функция объединения двух параметров в точечную пару;

6) *eq* — бинарная функция; будет возвращено T, если оба параметра являются одним и тем же атомом, Nil иначе;

7) *atom* — унарная функция, возвращающая T, если S-выражение является атомом, Nil иначе.

8) *cond* — функция произвольного числа аргументов, возвращающая результат первого выполненного условия из множества пар. Если ни одно условие не вернуло T, будет возвращено Nil.

# Глава 2

# Ход выполнения практического задания

## 2.1 Представление списков в виде списочных ячеек

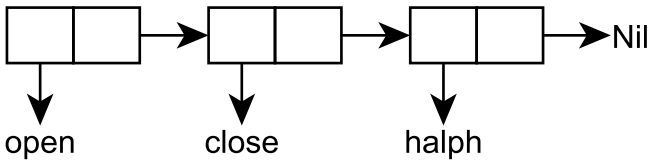


Рисунок 2.1 – '(open close halph)

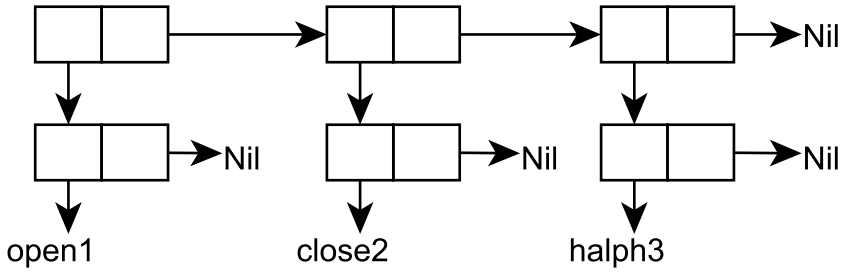


Рисунок 2.2 – '((open) (close) (halph))

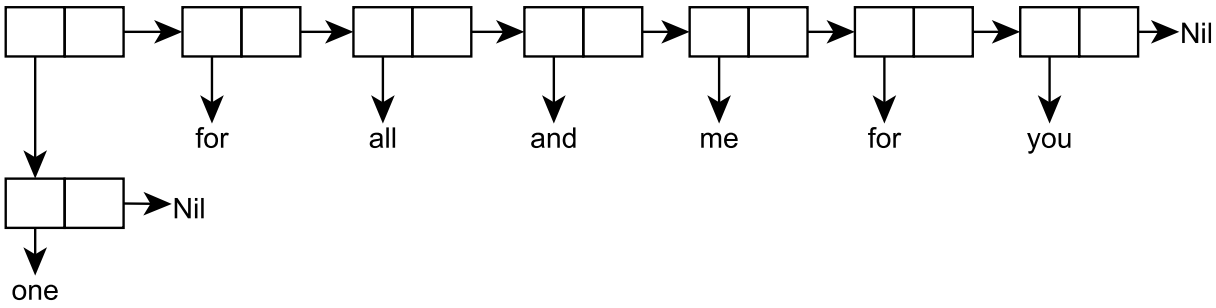


Рисунок 2.3 – ’((one) for all (and (me (for you))))

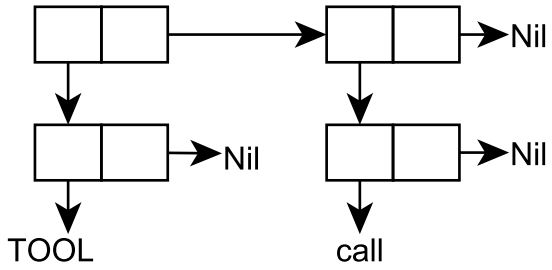


Рисунок 2.4 – ’((TOOL) (call))

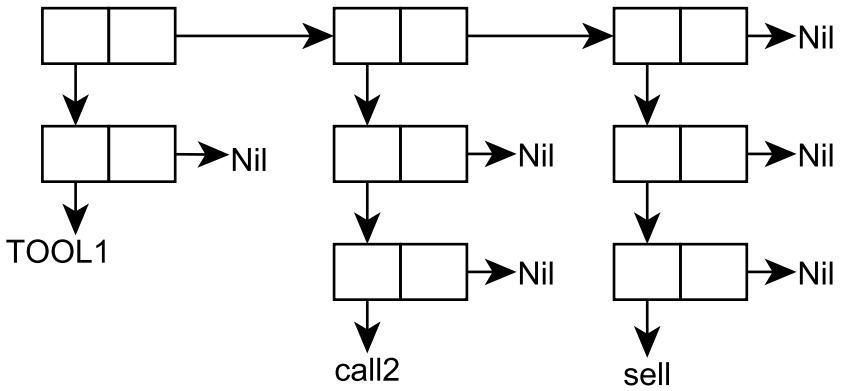


Рисунок 2.5 – ’((TOOL1) ((call2)) ((sell)))

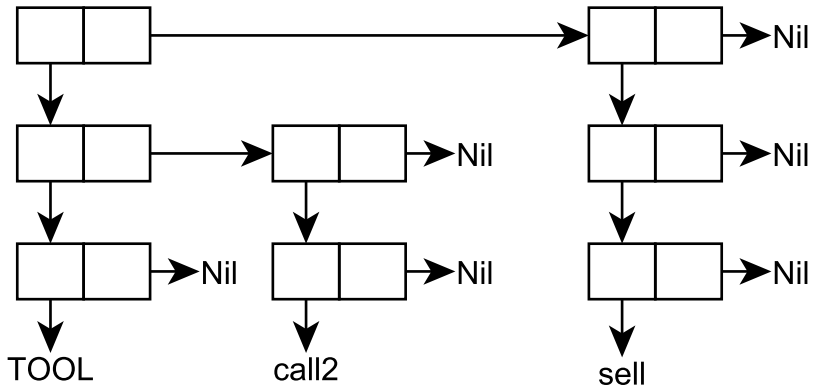


Рисунок 2.6 – ’ (((TOOL) (call)) ((sell)))

## 2.2 Используя только функции CAR и CDR, написать выражения

Рассмотрим выражения, возвращающие 2-й, 3-й и 4-й элементы соответственно, на примере списка: '(a b c d).

1) второй элемент списка — (car (cdr '(a b c d))) = (cadr '(a b c d)) →

2) третий элемент списка — (car (cdr (cdr '(a b c d)))) = (caddr '(a b c d)) →

3) четвертый элемент списка — (car (cdr (cdr (cdr '(a b c d))))) = (cadddr '(a b c d)) →

## 2.3 Определение результата вычисления выражений

1) (CAADR '((blue cube) (red pyramid))) — результат: red;

2) (CDAR '((abc) (def) (ghi))) — результат: Nil;

3) (CADR '((abc) (def) (ghi))) — результат: (def);

4) (CADDR '((abc) (def) (ghi))) — результат: (ghi);

## 2.4 Результат и объяснение вычисления выражений

1) (list 'Fred 'and 'Wilma) — результат: (Fred and Wilma). list — функция произвольного числа аргументов, формирующая список. В результате будет получен список из трех элементов. Апострофы обозначают работу с данными;

2) (list 'Fred '(and Wilma)) — результат: (Fred (and Wilma)). Аналогичный пример; второй аргумент является списком (and Wilma).

3) (cons Nil Nil) результат: (Nil). cons — бинарная функция объединения двух параметров в точечную пару. Так как вторым аргументом является список (Nil), точечная пара интерпретируется как список из одного элемента — Nil.

4) (cons T Nil) — результат: (T). Аналогично предыдущему примеру, будет получен список из одного элемента T.

5) (cons Nil T) — результат: (Nil.T). Вторым аргументом функции cons является не список, поэтому точечная пара не приводится к списку.

6) (list Nil) — результат: (Nil). Создан список из единственного элемента — Nil.

7) (cons '(T) Nil) — результат: ((T)). Аналогично рассмотренным выше случаям, второй аргумент — список (Nil), следовательно результатом является список из одного элемента — первого аргумента функции. Первым аргументом функции является список из одного элемента T.

8) (list '(one two) '(free temp)) — результат: ((one two) (free temp)). Будет получен список, элементами которого являются указанные списки.

9) (cons 'Fred '(and Wilma)) — результат: (Fred and Wilma). Точечная пара, вторым элементом которой является список, является списком.

10) (cons 'Fred '(Wilma)) — результат: (Fred Wilma). Аналогично предыдущему номеру.

11) (list Nil Nil) — результат: (Nil Nil). Формируем список из двух элементов.

12) (list T Nil ) — результат: (T Nil). Формируем список из двух элементов.

13) (list Nil T) — результат: (Nil T). Формируем список из двух элементов.

14) (cons T (list Nil)) — результат: (T Nil). Аналогично примерам выше; точечная пара, вторым элементом которой является список (Nil).

15) (list '(T) Nil) — результат: ((T) Nil). Формируем список из двух элементов, первый из которых является списком.

16) (cons '(one two) '(free temp)) — результат: ((one two) free temp). Точечная пара из двух списков становится списком, элементы которого

— аргументы функции cons (списки (one two) и (free temp) в данном примере).

## 2.5 Лямбда-выражения и соответствующие им функции

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

- (lambda (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3 ar4)))
- (defun f (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3 ar4)))

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

- (lambda (ar1 ar2) (list (list ar1) (list ar2)))
- (defun f (ar1 ar2) (list (list ar1) (list ar2)))

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

- (lambda (ar1) (list (list (list ar1))))
- (defun f (ar1) (list (list (list ar1))))

Представить результаты в виде списочных ячеек.

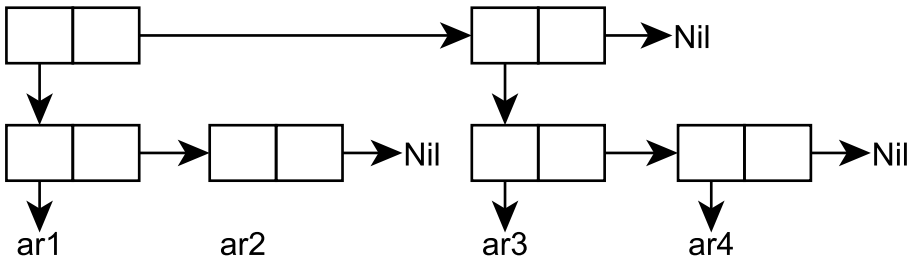


Рисунок 2.7 – ((ar1 ar2) (ar3 ar4))

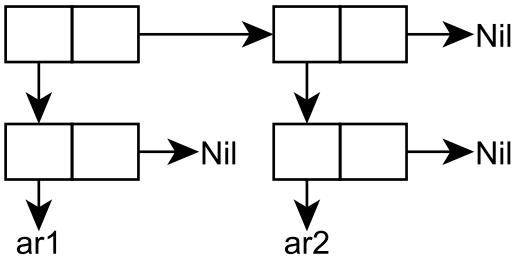


Рисунок 2.8 – ((ar1) (ar2))

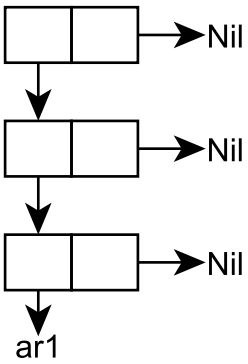


Рисунок 2.9 – (((ar1)))

# Вывод

Цель, которая была поставлена в начале лабораторной работы, была достигнута: были получены навыки использования списков и стандартных функций языка Lisp.

Кроме того, были выполнены все поставленные задачи:

- были изучены списки LISP как метод фиксации информации;
- были изучены методы обработки списков с использованием базовых функций Lisp.