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, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 1
по дисциплине «Функциональное и логическое программирование»

Тема Списки в Lispe. Использование стандартных функций.

Студент Царев А.А.

Группа ИУ7-63Б

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

Москва, 2023

ВВЕДЕНИЕ

Целью работы является приобретение навыков использования списков и стандартных функций Lisp.

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

1 Теоретические вопросы

1.1 Элементы языка: определение, синтаксис, представление в памяти

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

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

В языке Lisp допустимы синтаксические конструкции, в которых используются элементарные или более сложные конструкции.

Атомы — элементарные конструкции, которые являются:

— символами (идентификаторами) — синтаксически, это набор литер (букв и цифр), начинающихся с буквы;

— специальными символами T, Nil (используются для обозначения логических констант);

— самоопределимыми атомами — натуральные, дробные и вещественные числа, строки (последовательности символов, заключенных в двойные апострофы);

Более сложные данные — списки и точечные пары, которые строятся из унифицированных структур — блоков памяти. Синтаксис списков и точечных пар выглядит следующим образом:

Точечные пары ::= (<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>.

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

1.2 Особенности языка Lisp. Структура программы. Символ апостроф

Lisp является языком символьной обработки. Из этого следует, что система не различает программы и данные.

Основная конструкция, которая используется в Lisp, — это списки. Список — это структура, которая может быть пустой или непустой; если непустой, то он имеет хотя бы один элемент, называемый головой, а остальные — хвостом.

Основой языка Lisp является лямбда-исчисление, суть которого заключается в том, что любые вычисления могут быть преобразованы в композицию функций.

Безымянная функция:

```
(lambda (x1 x2 ... xn) f)
```

Функция с именем:

```
(defun <name> (x1 x2 ... xn) f)
```

Вызов:

```
(<name> a1 a2 ... an)
((lambda (x1 x2 ... xn) f) a1 a2 ... an)
```

По умолчанию система воспринимает конструкции как программы, при этом голова списка интерпретируется как имя функции, а остальные элементы — как ее фактические параметры. Для обозначения данных используется блокировка вычислений с помощью quote или символа апострофа:

```
(quote (A B C))
‘(A B C)
```

1.3 Базис языка Lisp. Ядро языка

Базис — минимальный набор конструкций языка, на основе которого могут быть построены вычисления.

Базис языка составляют:

— атомы;

— структуры;

— базовые функции;

— базовые функционалы.

Базовые функции языка:

— quote (блокировка вычисления);

— eval (принудительное вычисление);

— car (разыменование указателя на голову);

— cdr (разыменование указателя на хвост);

— cons (создание бинарного узла);

— atom (проверяет, является ли аргумент атомом или нет);

— cond (условная функция);
— eq (проверка на равенство, применима только к символьным атомам).

2 Практические задания

2.1 Задание 1

На рисунках 2.1 – 2.6 представлены списки в виде бинарных узлов.

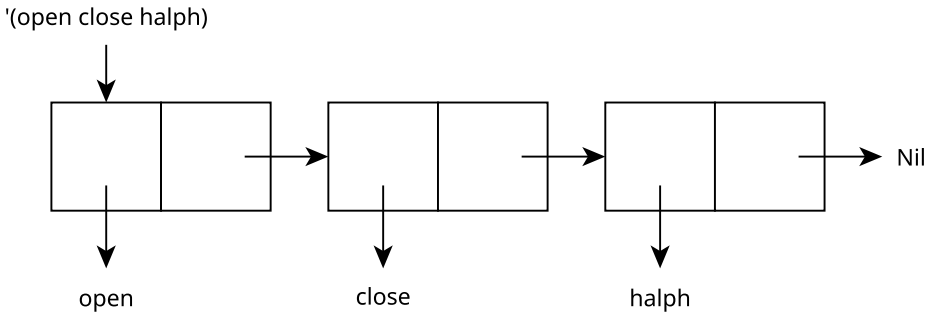

Рисунок 2.1 — Список ’(open close halph)

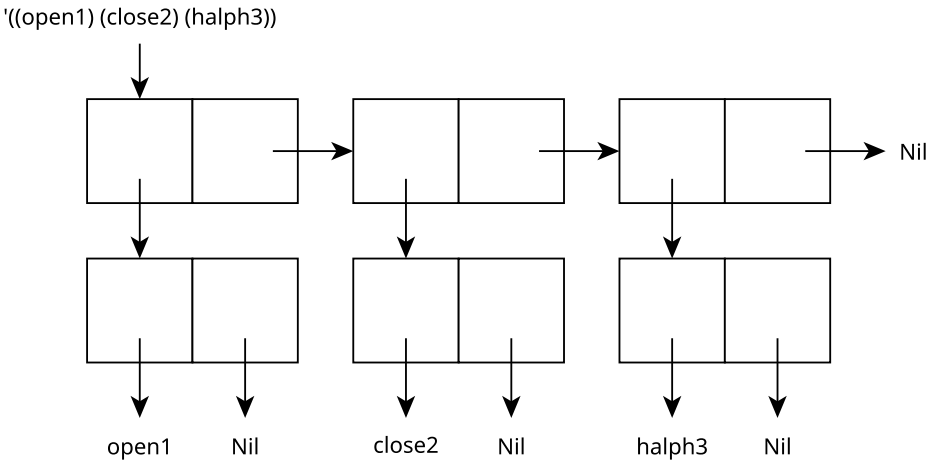

Рисунок 2.2 — Список ’((open1) (close2) (halph3))

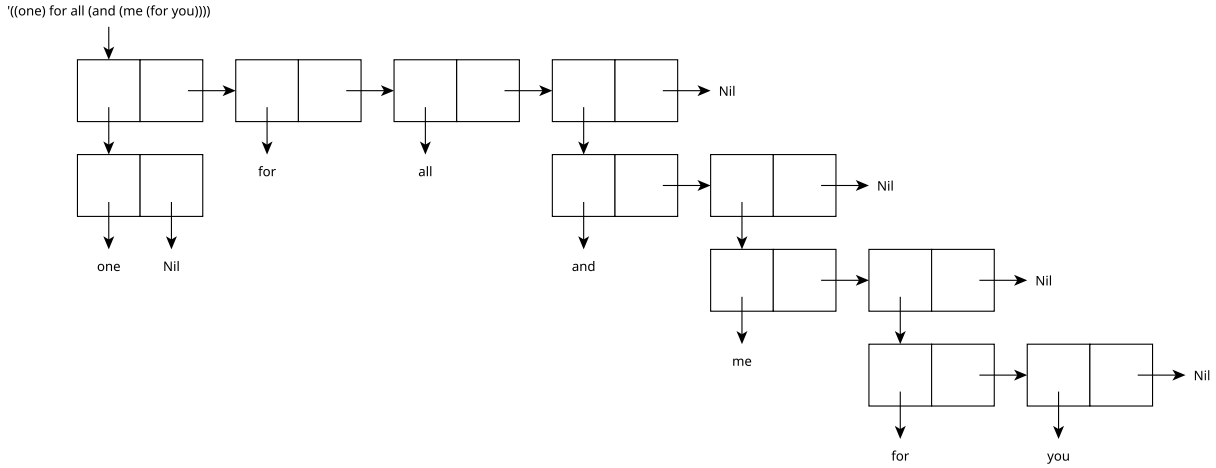

Рисунок 2.3 — Список ’((one) for all (and (me (for you))))

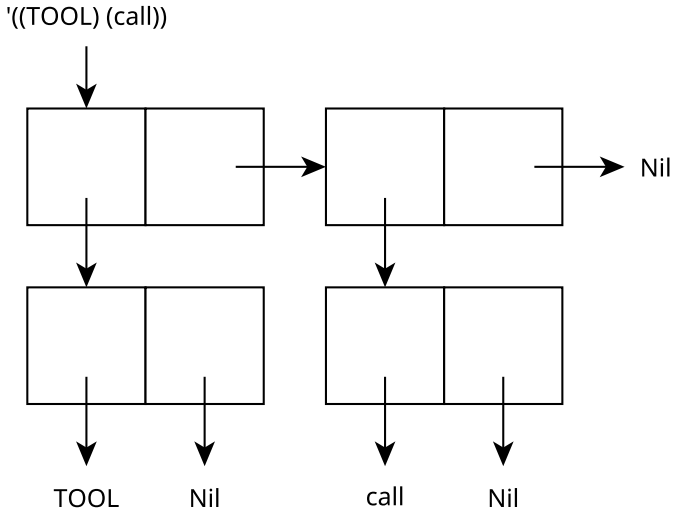

Рисунок 2.4 — Список ’((TOOL) (call))

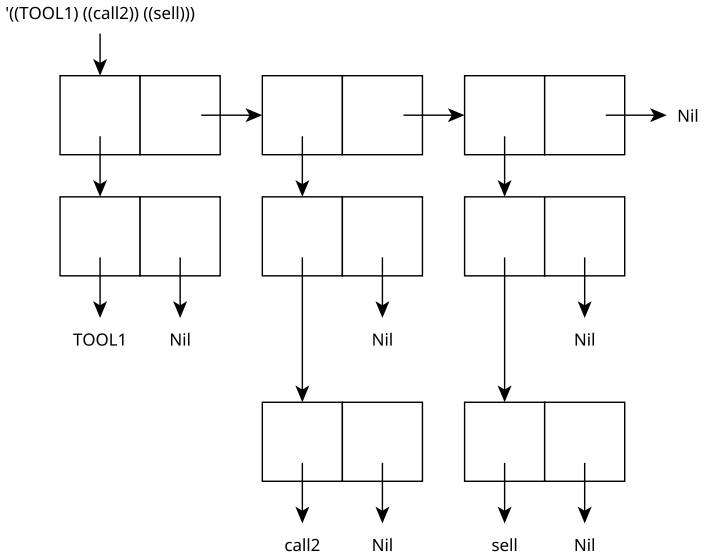

Рисунок 2.5 — Список ’((TOOL1) ((call2)) ((sell)))

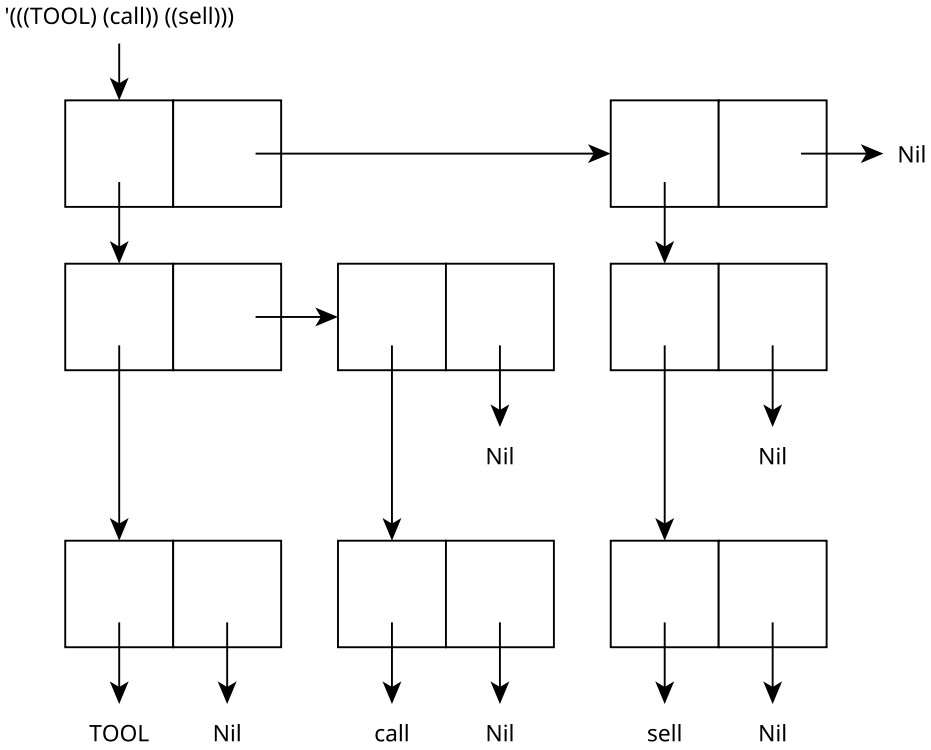

Рисунок 2.6 — Список ’(((TOOL) (call)) ((sell)))

2.2 Задание 2

```
(car (cdr '(first second third fourth)))
(car (cdr (cdr '(first second third fourth))))
(car (cdr (cdr (cdr '(first second third fourth)))))
```

2.3 Задание 3

В таблице 2.1 представлены результаты вычисления выражений.

Таблица 2.1 — Результаты вычисления выражений

Выражение	Результат
(CAADR '((blue cube) (red pyramid)))	red
(CDAR '((abc) (def) (ghi)))	Nil
(CADR '((abc) (def) (ghi)))	(def)
(CADDR '((abc) (def) (ghi)))	(ghi)

2.4 Задание 4

В таблице 2.2 представлены результаты вычисления выражений.

Таблица 2.2 — Результаты вычисления выражений

Выражение	Результат	Объяснение
(list 'Fred 'and 'Wilma)	(Fred and Wilma)	Создается список, состоящий из атомов Fred, and и Wilma, переданных функции list в качестве фактических параметров

(cons 'Fred '(and Wilma))	(Fred and Wilma)	Создается бинарный узел, первый указатель которого ссылается на атом Fred, а второй — на список (and Wilma). Так как второй указатель узла ссылается на список, то в результате получается список (Fred and Wilma)
(list 'Fred '(and Wilma))	(Fred (and Wilma))	Создается список, состоящий из атома Fred и списка (and Wilma), переданных функции list в качестве фактических параметров
(cons 'Fred '(Wilma))	(Fred Wilma)	Создается бинарный узел, первый указатель которого ссылается на атом Fred, а второй — на список (Wilma). Так как второй указатель узла ссылается на список, то в результате получается список (Fred Wilma)
(cons Nil Nil)	(Nil)	Создается бинарный узел, первый и второй указатели которого ссылаются на пустые списки. Так как второй указатель узла ссылается на пустой список, то в результате получается список (Nil)
(list Nil Nil)	(Nil Nil)	Создается список, состоящий из двух пустых списков, переданных функции list в качестве фактических параметров

(cons T Nil)	(T)	Создается бинарный узел, первый указатель которого ссылается на атом T, а второй — на пустой список. Так как второй указатель узла ссылается на пустой список, то в результате получается список (T)
(list T Nil)	(T Nil)	Создается список, состоящий из атома T и пустого списка, переданных функции cons в качестве фактических параметров
(cons Nil T)	(Nil . T)	Создается бинарный узел, первый указатель которого ссылается на пустой список, а второй — на атом T
(list Nil T)	(Nil T)	Создается список, состоящий из пустого списка и атома T, переданных функции list в качестве фактических параметров
(list Nil)	(Nil)	Создается список, содержащий в себе пустой список, переданный функции list в качестве фактического параметра
(cons T (list Nil))	(T Nil)	Создается бинарный узел, первый указатель которого ссылается на атом T, а второй — на результат выполнения функции list. Так как результат выполнения (list Nil) является списком, содержащим в себе пустой список, то по итогу получается список (T Nil)

(cons '(T) Nil)	((T))	Создается бинарный узел, первый указатель которого ссылается на список (T), а второй — на пустой список. Так как второй указатель узла ссылается на пустой список, то в результате получается список ((T))
(list '(T) Nil)	((T) Nil)	Создается список, состоящий из списка (T) и пустого списка, переданных функции list в качестве фактических параметров
(list '(one two) '(free temp))	((one two) (free temp))	Создается список, состоящий из списков (one two) и (free temp) переданных функции list в качестве фактических параметров
(cons '(one two) '(free temp))	((one two) free temp)	Создается бинарный узел, первый указатель которого ссылается на список (one two), а второй — на список (free temp). Так как второй указатель узла ссылается на список, то в результате получается список ((one two) free temp)

2.5 Задание 5

```
; 1
(defun f1 (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3 ar4)))
(lambda (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3 ar4)))

; 2
(defun f2 (ar1  ar2) (list (list ar1) (list ar2)))
(lambda (ar1  ar2) (list (list ar1) (list ar2)))

; 3
(defun f3 (ar1) (list (list (list ar1))))
(lambda (ar1) (list (list (list ar1))))
```

На рисунках 2.7 – 2.9 представлены результаты функций в виде списочных ячеек.

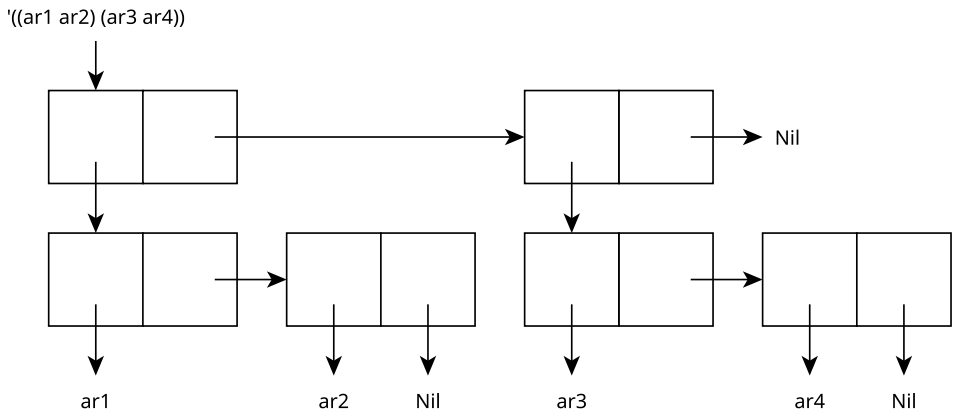

Рисунок 2.7 — Список ’((ar1 ar2) (ar3 ar4))

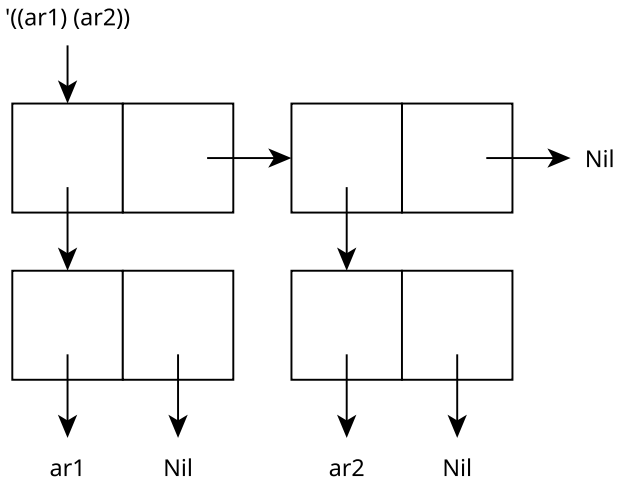

Рисунок 2.8 — Список ’((ar1) (ar2))

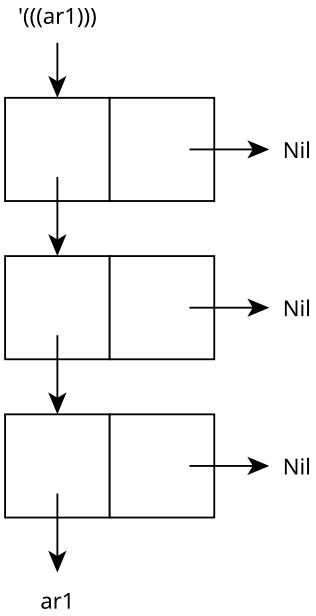

Рисунок 2.9 — Список ’(((ar1)))

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были приобретены навыки использования списков и стандартных функций Lisp.

Была выполнена задача работы: изучены способ использования списков для фиксации информации, внутреннее представление одноуровневых и структурированных списков, методы их обработки с использованием базовых функций Lisp.