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»

Отчёт по лабораторной работе №1 по дисциплине «Функциональное и логическое программирование»

Тема: Списки в Lisp. Использование стандартных функций
Студент: Княжев А. В.
Группа: ИУ7-62Б
Оценка (баллы): _____
Преподаватели: Толпинская Н. Б., Строганов Ю. В.

Москва — 2023 г.

Оглавление

1. Теоретическая часть **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. Практическая часть **5**

2.1. Задание 1 5

2.1.1. Задание 5

2.1.2. Решение 5

2.2. Задание 2 5

2.2.1. Задание 5

2.2.2. Решение 6

2.3. Задание 3 6

2.3.1. Задание 6

2.3.2. Решение 6

2.4. Задание 4 6

2.4.1. Задание 6

2.4.2. Решение 7

2.5. Задание 5 8

2.5.1. Задание 8

2.5.2. Решение 8

1. Теоретическая часть

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>.

Элементарные значения структур данных:

— 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”);

— Точечные пары

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

— Списки

Список ::= <пустой список> | <непустой список>,

<пустой список> ::= ( ) | Nil,

<непустой список>::= (<первый элемент> . <хвост>),

<первый элемент> ::= <S-выражение>, <хвост> ::= <список>.

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

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 `()`; непустой список по определению может быть изображен: `(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

Важной особенностью языка Lisp является то, что программы, написанные на Lisp, представляются в виде его же структур данных. Это позволяет выдать программы за данные и изменять их.

Программа на языке Lisp представляет собой последовательность вычислимых выражений, которые являются атомами или списками.

Символ `'` (апостроф) эквивалентен функции `quote` — блокирует вычисление выражения, и оно воспринимается интерпретатором как данные.

1.3. Базис языка Lisp. Ядро языка

Базис — это минимальный набор действий в языке программирования. Базис языка Lisp представлен атомами, структурами и функциями. Некоторые базисные функции: `car`, `cdr`, `cons`, `list`, `lambda`, `quote`, `eq`.

Ядром языка называется базис языка и функции стандартной библиотеки (часто используемые функции, созданные на основе базиса).

2. Практическая часть

2.1. Задание 1

2.1.1. Задание

Представить следующие списки в виде списочных ячеек:

— `'(open close halph)`

— `'((open1) (close2) (halph3))`

— `'((one) for all (and (me (for you))));`

— `'((TOOL) (call));`

— `'((TOOL1) ((call2)) ((sell)))`

— `'(((TOOL) (call)) ((sell))).`

2.1.2. Решение

Приложено на отдельном листе.

2.2. Задание 2

2.2.1. Задание

Используя только функции `CAR` и `CDR`, написать выражения, возвращающие

1. второй
2. третий
3. четвертый

элементы заданного списка.

2.2.2. Решение

1. `(car (cdr ‘(1 2 3 4 5 6)))`
2. `(car (cdr (cdr ‘(1 2 3 4 5 6))))`
3. `(car (cdr (cdr(cdr ‘(1 2 3 4 5 6)))))`

2.3. Задание 3

2.3.1. Задание

Что будет в результате вычисления выражений?

1. `(CAADR ’ ((blue cube) (red pyramid)))`
2. `(CDAR ’((abc) (def) (ghi)))`
3. `(CADR ’ ((abc) (def) (ghi)))`
4. `(CADDR ’ ((abc) (def) (ghi)))`

2.3.2. Решение

1. `RED`
2. `NIL`
3. `(DEF)`
4. `(GHI)`

2.4. Задание 4

2.4.1. Задание

Напишите результат вычисления выражений и объясните как он получен:

1. `(list ’Fred ’and ’Wilma)`
2. `(list ’Fred ’(and Wilma))`
3. `(cons Nil Nil)`

4. `(cons T Nil)`
5. `(cons Nil T)`
6. `(list Nil)`
7. `(cons ' (T) Nil)`
8. `(list ' (one two) ' (free temp))`
9. `(cons 'Fred '(and Wilma))`
10. `(cons 'Fred '(Wilma))`
11. `(list Nil Nil)`
12. `(list T Nil)`
13. `(list Nil T)`
14. `(cons T (list Nil))`
15. `(list '(T) Nil)`
16. `(cons '(one two) '(free temp))`

2.4.2. Решение

Функция `cons` создает списочную ячейку. Принимает два аргумента. Если второй аргумент — список, то результатом вызова функции будет список, если атом — результатом будет точечная пара. Если второй аргумент — `Nil`, то результатом будет список из одного элемента.

Функция `list` создает столько списочных ячеек, сколько ей было передано аргументов. Она не является частью базиса.

1. `(FRED AND WILMA)`
2. `(FRED (AND WILMA))`
3. `(NIL)`
4. `(T)`
5. `(NIL . T)`

6. `(NIL)`
7. `((T))`
8. `((ONE TWO) (FREE TEMP))`
9. `(FRED AND WILMA)`
10. `(FRED WILMA)`
11. `(NIL NIL)`
12. `(T NIL)`
13. `(NIL T)`
14. `(T NIL)`
15. `((T) NIL)`
16. `((ONE TWO) FREE TEMP)`

2.5. Задание 5

2.5.1. Задание

Написать лямбда-выражение и соответствующую функцию:

1. Написать функцию `(f arl ar2 ar3 ar4)`, возвращающую список: `((arl ar2) (ar3 ar4))`.
2. Написать функцию `(f arl ar2)`, возвращающую `((arl) (ar2))`.
3. Написать функцию `(f arl)`, возвращающую `(((arl)))`.
4. Представить результаты в виде списочных ячеек.

2.5.2. Решение

1. ```
   ((lambda (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3 ar4))) 1 2 3 4)
   (defun f1 (ar1 ar2 ar3 ar4) (list (list ar1 ar2) (list ar3 ar4)))
   ```
2. ```
   ((lambda (ar1 ar2) (list (list ar1) (list ar2))) 1 2)
   (defun f2 (ar1 ar2) (list (list ar1) (list ar2)))
   ```

3. ```
   ((lambda (ar1) (list (list (list ar1)))) 1)

   (defun f3 (ar1) (list (list (list ar1))))
   ```

4. Приложено на отдельном листе.
   ```