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 Lispe. Использование стандандартных функций.

Студент Козлова И.В.

Группа ИУ7-62Б

Оценка (баллы)

Преподаватель Толпинская Н.Б.

Преподаватель Строганов Ю.В.

Москва — 2022 г.

Практические вопросы

Решение оформлено на отдельном листе бумаге, прилагающемуся к отчету.

Теоретические 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  B
2  CAT
3  123
4  2/3
5  "abc"
```

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.

Более сложные данные - списки и точечные пары. строятся из унифицированных структур – блоков памяти – бинарных узлов.

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

```
1  Точечная пара ::= (<атом>.<атом>) |
2                    (<атом>.<точечная пара>) |
3                    (<точечная пара>.<атом>) |
4                    (<точечная пара>.<точечная пара>)
```

Пример точечной пары:

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

Облегченная форма записи:

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

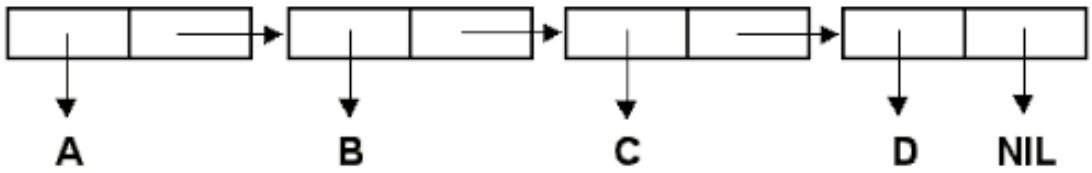

Рисунок 1 – Представление в памяти (A B C D).

Представление в памяти:

1. **(A . B)**

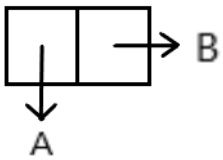

Рисунок 2 – Представление в памяти (A . B).

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

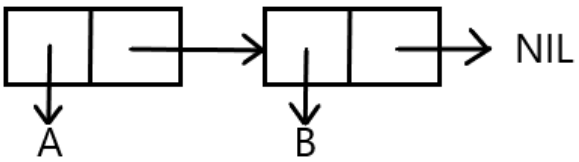

Рисунок 3 – Представление в памяти (A B).

Вся информация (данные и программы) в Lisp представляется в виде символьных выражений – S-выражений.

```
1    S–выражение ::= <атом> | <точечная пара>
```

Список является частым случаем S-выражения.

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

1. Головы – S-выражение.

2. Хвоста – список.

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

Структуры – тотечная пара и список

2. Синтаксис.

Lisp является регистронезависимым языком. Lisp использует префиксную нотацию.

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.

Любая структура заключается в круглые скобки, в памяти представляется как списковая ячейка, хранящая два указателя: на голову (первый элемент) и на хвост (все остальное).

(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. Особенности языка Lisp. Символ апостроф

Особенности языка Lisp: только символьная обработка данных.

От других языков программирования Lisp отличается следующими свойствами:

1. Используется символьная обработка данных.
2. Нет типизации (без типовый язык).
3. Программы, написанные на Lisp, представляются в виде списков.
4. Программа может быть представлена в виде данных, поэтому программа может модифицировать сама себя.
5. Память выделяется блоками. Lisp сам распределяет память.

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.

4. Базис Lisp

Базис Lisp

- атомы и структуры (представляющиеся бинарными узлами);
- базовые (несколько) функций и функционалов: встроенные — примитивные функции (atom, eq, cons, car, cdr); специальные функции и функционалы (quote, cond, lambda, eval, apply, funcall).

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

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

- eq проверяет идентичность двух символов

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

- cond (от англ. condition – условие). Нет фиксированного количества аргументов.

  Каждый аргумент это список, голова которого рассматривается как условие, в случае если оно истинно, то хвост рассматривается как результат.

```
(cond ((eq 'A 'B) 'yes)
      (T 'not)) ;; NOT
```

- eval выполняет двойное вычисление своего аргумента.

```
(eval (cons (quote car) (quote ('(A B)))))
```

- apply функция, аргументом которой является функция. Данная функция требует функцию и список ее аргументов и возвращает результат этой функции с заданными аргументами.

```
(apply #'+ '(1 2 3))
(apply #'+ 1 2 '(1 2 3))
```

- funcall аналогично apply, но не требует, чтобы аргументы были упакованы в список.

```
(funcall #'+ 1 2 3)
(funcall #'+ 1 2 1 2 3)
```

Ядро – основные действия, которые наиболее часто используются. «Ядро» шире, чем «базис».

5. Функция

Функцией называется правило, по которому каждому значению одного или нескольких аргументов ставится в соответствие конкретное значение результата.

Функция в Lispe есть однозначное отображение множества исходных данных на множество её значений. У функции может быть произвольно много аргументов, от нуля до любого конечного числа, но обязательно должно быть хотя бы одно значение

Функционалом, или функцией высшего порядка называется функция, аргументом или результатом которой является другая функция.

7. Классификация функций

Классификация базисных функций:

- селекторы (car cdr)
- конструкторы (cons list)
- предикаты (atom, consp, listp, null, numberp, oddp)

8. CAR и CDR

Список в языке Lisp представлен одним бинарным узлом, который хранит два указателя (на голову и хвост).

CAR и CDR являются базовыми функциями доступа к данным.

CAR принимает точечную пару или пустой список в качестве аргумента и возвращает первый элемент или nil, соответственно.

CDR принимает точечную пару или пустой список и возвращает список состоящий из всех элементов, кроме первого. Если в списке меньше двух элементов, то возвращается Nil.

9. LIST и CONS

LIST и CONS являются функциями создания списков (cons – базовая, list – нет). Функция cons (принимает только два аргумента) создает списочную ячейку и устанавливает два указателя на аргументы.

Функция list принимает переменное число аргументов и возвращает список, элементы которого – переданные в функцию аргументы.

Например список '(open close halph) из задания 1 можно представить как: (cons 'open (cons 'close (cons 'halph nil))) или (list 'open 'close 'halph).