Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»
Кафедра «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №3

по курсу:

«Функциональное и Логическое программирование»

Студент группы ИУ7-63Б: М. В. Заколесник
(Фамилия И.О.)
Преподователь: Н. Б. Толпинская
(Фамилия И.О.)

Москва, 2019

Оглавление

Задание

Лабораторная работа №3

- Используя только функции car и cdr, написать выражения, возвращающие:

  1) (list 'Fred 'and Wilma)) = **Error** ; т.к. Wilma ни к чему не вычисляется.
  2) (list 'Fred '(and Wilma)) = (Fred (and Wilma))
  3) (cons Nil Nil) = (Nil.Nil) =(Nil)
  4) (cons T Nil) = (T.Nil) = (T)
  5) (cons Nil T) = (Nil.T)
  6) (list Nil) = (Nil)
  7) (cons (T) Nil) = **Error** ; т.к. первый элемент списка является функцией, а T- атом(не является функцией)
  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) = **Error** ; т.к. первый элемент списка является функцией, а T- атом(не является функцией)
  16) (cons '(one two) '(free temp)) = ((one two) free temp)

Ответы на вопросы

1) Как представляются списки в ОП? Выписать и дать определение основным элементам языка Lisp.

   Списочная ячейка состоит из двух частей, полей first и rest. Каждое из полей содержит указатель. Указатель может ссылаться на другую списочную ячейку или на некоторый другой Lisp объект, как, например, атом. Указатели между ячейками образуют как бы цепочку, по которой можно из предыдущей ячейки попасть в следующую и так, наконец, до атомарных объектов. Каждый известный системе атом записан в определённом месте памяти лишь один раз.

   Основные элементы языка:

   - Атомы:
     - Символ
     - Символьная константа
     - Число(целые, действительные, рациональные)
     - Логические(T, Nil)
   - Точечные пары:
     - (<атом> . <атом>)
     - (<точ. пара> . <атом>)
     - (<атом> . <точ. пара>)
     - (<точ. пара> . <точ. пара>)
   - S-выражения:
     - <атом> | <точ.пара>
   - Списки:
     - (<s-выражение> . <список>)
     - (<пустой список>) ≡ Nil

2) Как выполняются CAR и CDR?

   - car - переходит по car указателю и возвращает голову
   - cdr - переходит по cdr указателю и возвращает хвост(остаток)

3) Классификация функций:

3.1. Чистые(математические)- функции имеют фиксированное количество аргументов;

3.2. Специальные- специальным образом обрабатывают свои аргументы;

3.3. Псевдофункции- реализация аппаратно-зивисимых действий;

3.4. Функции, допускающие вариантные значения- в процессе работы могут быть выбран один из результатов;

3.5. Функции, позволяющие организовать отложенные "ленивые"вычисления- ленивые вычисления могут быть вообще не выполнены;

3.6. Функции высших порядков- в процессе работы формируют другие функции.

4) Классификация базисных функций:

4.1. Конструкторы(cons, list)

- **cons**- двухъаргументная функция, создаёт одну списковую ячейку и расставляет указатели;
  (cons 'a 'b) -> (a.b)
  (cons 'a '(b)) -> (a b)
- **list**- создаёт столько списковых ячеек, сколько аргументов
  (list 'a 'b) -> (a b)
  cons работает быстрее, чем list, но может быть организован не список

4.2. Селекторы(car, cdr)

- **car**- переходит по car указателю и возвращает голову
- **cdr**- переходит по cdr указателю и возвращает хвост(остаток)

4.3. Предикаты(позволяющие определить структуру элементов)

- **atom**- возвращает T, если аргумент- атом, иначе Nil;
- **listp**- возвращает T, если аргумент- список, иначе Nil;
- **consp**- представлена ли пара в виде списковых ячеек;
- **numberp**- возвращает T, если аргумент- число, иначе Nil;
- **symbolp**- возвращает T, если аргумент- не число, иначе Nil;

4.4. Сравнение(позволяющие сравнивать элементы):

- **базовая eq**- сравнивает два символьных атома(указатели на них) на их тождественность возвращает Nil или T, применима только для символов.

  (eq T T) -> T
- **eql**- сравнивает два числа(по форме их представления)

  (eql 3 3) -> T

  (eql 3 3.0) -> Nil
- =- сравнивает два числа

  (= 3 3) -> T

  (= 3 3.0) -> T
- **equal**- сравнивает как и eql, но также может сравнивать списки
- **equalp**- сравнивает все варианты представления, но долго работает

Для чисел:

- **oddp**- проверка на нечетность
- **evenp**- проверка на четность

5) В чем отличие выполнения функций list и cons?

- **cons**- является базисом языка, она на вход принимает ровно два аргумента и создает одну списковую ячейку(расставляет указатели)
- **list**- написана на базе функции **cons**, принимает любое количество аргументов и создает список

Литература

1. Толпинская Н.Б. - Курс лекций по "Функциональному и Логическому программированию"[Текст], Москва 2019 год.

2. Городняя Л.В. - Основы функционального программирования. Курс лекций : учеб. пособие для вузов / Городняя Л. В. - М. : Интернет-Университет Информационных Технологий, 2004. - 272 с. - (Основы информационных технологий). - Библиогр.: с. 269-272 и в конце кн. - ISBN 5-9556-0008-6.