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 №2 по дисциплине «Функциональное и логическое программирование»

Тема Определение функций пользователя

Студент Золотухин А. В.

Группа ИУ7-64Б

Оценка (баллы) _____

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

Москва — 2023 г.

Теоретические вопросы

1. Базис Lisp

Базис – это минимальный набор инструментов языка и стркутур данных, который позволяет решить любые задачи.

Базис Lisp :

- атомы (представляются в памяти пятью указателями – name, value, function, property, package) и структуры (представляющиеся бинарными узлами);
- базовые (несколько) функций, функционалов и форм: встроенные — примитивные функции (atom, eq, cons, car, cdr); формы (quote, cond, lambda, eval); функционалы (apply, funcall).

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, вещественные числа, строки – последовательность символов, заключенных в двойные апострофы (например, “abc”);

Более сложные данные – списки и точечные пары (структуры), которые строятся с помощью унифицированных структур – блоков памяти – бинарных узлов.

Определения:

Точечная пара ::= (<атом> . <атом>) | (<атом> . <точечная пара>) | (<точечная пара> . <атом>) | (<точечная пара> . <точечная пара>);

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

<пустой список> ::= () | Nil,

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

<первый элемент> ::= <S-выражение>,

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

<хвост> ::= <список>.

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

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

Форма – функция, которая особым образом обрабатывает свои аргументы или имеет переменное количество параметров.

2.Классификация функций

1. Чистые математические функции (имеют фиксированное количество аргументов, сначала выяисляются все аргументы, а только потом к ним применяется функция);
2. Рекурсивные функции (основной способ выполнения повторных вычислений);
3. Специальные функции, или формы (могут принимать произвольное количество аргументов, или аргументы могут обрабатываться по-разному);
4. Псевдофункции (создают «эффект», например, вывод на экран);
5. Функции с вариантами значений, из которых выбирается одно;
6. Функции высших порядков, или функционалы – функции, аргументом или результатом которых является другая функция (используются для построения синтаксически управляемых программ);

Классификаци базисных функций и функций ядра.

1. Селекторы: car и cdr (будут подробнее расссмотрены ниже).
2. Конструкторы: cons и list (будут подробнее расссмотрены ниже).
3. Предикаты – «логические» функции, позволяющие определить структуру элемента:
   - atom возвращает T, если значением её единственного аргумента является атом, иначе – NIL;
   - null возвращает T, если значение его аргумента – NIL (пустой список), иначе – NIL;
   - listp возвращает T, если значением её аргумента является список, иначе – NIL;

- consp возвращает T, если значением её аргумента является структура, представленная в виде списковой ячейки, иначе – NIL.

4. Функции сравнения (принимают два аргумента, перечислены по мере роста «тщательности» проверки):

   - eq корректно сравнивает два символьных атома. Так как атомы не дублирутюся для данного сеанса работы, то фактически сравниваются соответсвующие указатели. Возвращает T, когда: 1) значением одного из аргументов является атом, и одновременно 2) значения аргументов равны (идентичны). В ином случае значением функции eq является NIL. (eq 'ab 'Ab) => T, но (eq 1 2) => NIL.
   - eql корректно сравнивает атомы и числа одинакового типа (синтетической формы записи). Например, (eql 1 1) вернет T, а (eql 1 1.0) – Nil, так как целое значение 1 и значение с плавающей точкой 1.0 являются представителями различных классов;
   - = корректно сравнивает только числа, причем числа могут быть разных типов. Например, и (= 1 1), и (= 1 1.0) вернет T;
   - equal работает идентично eql, но в дополнение умеет корректно сравнивать списки (считая списки эквивалентными, если они рекурсивно, согласно тому же equal, имеют одинаковую структуру и содержимое; считая строки эквивалентными, если они содержат одинаковые знаки);
   - equalp корректно сравнивает любые S-выражения.

3. Способы создания функций

Определение функций пользователя в Lisp-е возможно двумя способами.

- Базисный способ определения функции - использование λ-выражения (λ-нотации). Так создаются функции без имени.

  λ-выражение: (lambda λ-список форма), где λ-список – это формальные параметры функции (список аргументов), а форма – это тело функции.

  Вызов такой функции осуществляется следующим способом: (λ-выражение последовательность_форм), где последовательность_форм – это фактические параметры.

  Вычисление функций без имени может быть также выполнено с использованием функционала apply: (apply λ-выражение последовательность_форм),

где последовательность_форм – это список фактических параметров; или с использованием функционала funcall: (funcall λ-выражение последовательность_форм), где последовательность_форм – это фактические параметры.

Функционал apply является обычной функцией с двумя вычисляемыми аргументами, обращение к ней имеет вид: (apply F L), где F – функциональный аргумент и L – список, рассматриваемый как список фактических параметров для F. Значение функционала – результат применения F к этим фактическим параметрам.

Функционал funcall – особая функция с вычисляемыми аргументами, обращение к ней: (funcall F e1 ... en), n $\geqslant$ 0. Её действие аналогично apply, отличие состоит в том, что аргументы применяемой функции F задаются не списком, а по отдельности.

funcall используется тогда, когда во время написания кода количество аргументов известно, apply – когда неизвестно.

- Другой способ определения функции – использование макро-определения defun:

  (defun имя_функции λ-выражение),

  или в облегченной форме:

  (defun имя_функции $(x_1, x_2, ..., x_k)$ форма), где $(x_1, x_2, ..., x_k)$ – это список аргументов.

  В качестве имени функции выступает символьный атом. Вызов именованной функции осуществляется следующим образом: (имя_функции последовательность_форм), где последовательность_форм – это фактические параметры. Также для ее вызова можно воспользоваться рассмотренными выше функционалами funcall (например, (foo 1 2 3) === (funcall #'foo 1 2 3)) и apply (например, (apply #'plot plot-data), где plot-data - список, хранящий аргументы).

λ-определение более эффективно, особенно при повторных вычислениях.

Параметры функции, переданные при вызове, будут связаны с переменными в списке параметров из объявления функции. Еще один способ связывания формальных параметров с фактическими – использование функции let:

(let ((x1 p1) (x2 p2) ... (xk pk)) e),

где xi – формальные параметры, pi – фактические параметры (могут быть формами), e – форма (что делать).

4. Функции Car и Cdr

Функции car и cdr переходят по соответсвующему указателю аргумента (бинарного узла).

Функция car от одного аргумента возвращает первый элемент списка, являющегося значением её аргумента.

Функция cdr возвращает хвост списка, являющегося значением её единственного аргумента (хвостом, или остатком списка является список без своего первого элемента).

Современные диалекты Лиспа обычно допускают для функций car и cdr мнемоничные синонимичные названия: first и rest соответственно.

5. Назначение и отличие в работе Cons и List

cons принимает 2 указателя на любые S-выражения и возвращает новую cons-ячейку (списковую ячейку), содержащую 2 значения. Если второе значение не NIL и не другая cons-ячейка, то ячейка печатается как два значения в скобках, разделённые точкой (так называемая точечная пара). Иначе, по сути, эта функция включает значение первого аргумента в начало списка, являющегося значением второго аргумента.

Функция list, составляющая список из значений своих аргументов (у которого голова – это первый аргумент, хвост – все остальные аргументы), создает столько списковых ячеек, сколько аргументов ей было передано. Эта функция относится к особым, поскольку у неё может быть произвольное число аргументов, но при этом все аргументы вычисляются

Итак, отличия: cons является базисной, list – нет; cons имеет фиксированное количество аргументов (два), list – произвольное; cons создает точечную пару или список (в зависимости от второго аргумента), list – список; в отличие от функции cons, результат функции list симметричен относительно аргументов: (list '(A) '(B)) => ((A)(B)), но (cons '(A) '(B)) => ((A) B).

cons эффективнее list, list определяется с помощью cons.

Практические задания