Список - частный случай S-выражения, кот. может быть пустым или не пустым (имеет голову и хвост, кот. также пустой или не пустой).

Лисп. Атомы и структуры - исп для хранения структур и данных, в том числе ячейку, кот хранит 2 указателя

Функция -

Базис – необходимый набор средств

Способ представления программ и данных

Базис языка:

* атомы(eq, e)
* структуры
* Базисные функции:
* Atom
* Eq
* Car
* Cdr
* Cons
* quote ‘
* cond
* lambda
* label
* eval
* Базисные функционалы:
* apply
* funcall

Над базисом строятся простые формулы в виде списков, где первый элем- имя функции, остальное – её аргументы. Остальные механизмы вычисления могут сводиться к этому базису

Классификация функций:

* «Чистые» - строго математические функции – имеет фиксированное кол-во аргументов и один результат.
* Специальные функции – формы – обрабатывают аргументы нестандартным способом или имеют неопределенное кол-во аргументов.(+, …)
* Псевдо-функции – создают эффект на внешних устройствах.
* Функции, допускающие вариантные значения – позволяют реализовывать логическое программирование
* Допускаются вычисления - ленивые вычисления, сведение вызова функций к представлению правила вычисления функций в определенном контексте
* Функции высших порядков – функционалы

Классификация по назначению:

* Функции-конструкторы
  + cons - создает одну списковую ячейку и раздаёт указатели
  + list – всегда возвращает список, имеет любое кол-во аргументов, расставляет указатели для связи, элементов столько, сколько изначально дано, работает дольше
* Селекторы
  + car- переход по внешнему указателю
  + cdr- переход по внутреннему указателю
* Предикаты
  + atom- возвращает True, если это атом
  + listp- список или нет
  + consp- проверяет представлено ли в виде списковых ячеек
  + numberp –
  + symbol
* Сравнение
  + eq – сравнивает указатели на символьные атомы, не сравнивает числа
  + eql – сравнивает числа

(eql 3 3) -> T

(eql 3 3.0) -> Nil

(= 3 3.0) -> T

* + equal ~ eql + списки
  + equalp – сравнивает аргументы разной структуры
  + oddp – проверка на четность
  + evenp – проверка на нечетность

Определение функций:

* Определение функции с помощью спец функции defun

(defun name (arg1..argN)) (тело функции)

* Лямбда определение функции- определение функции без имени

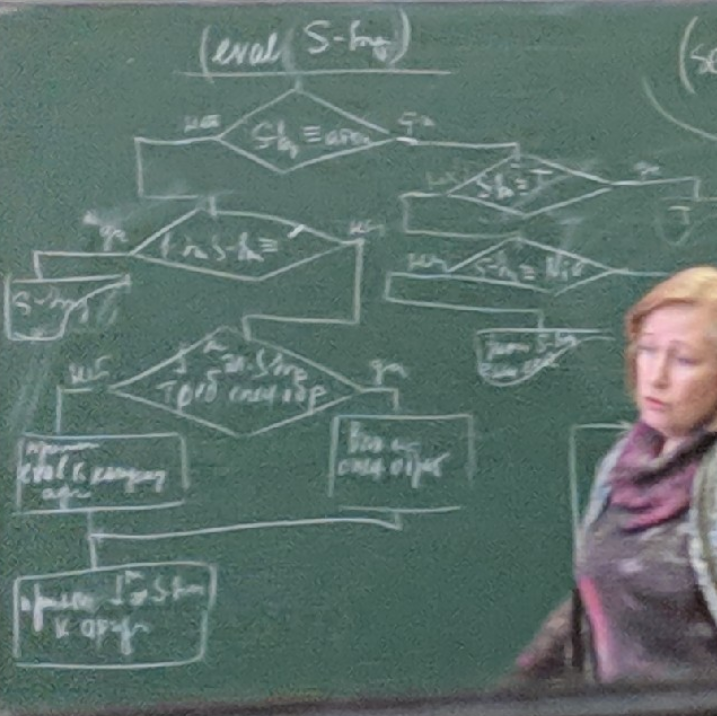
Лямбда- функция, кот входит в базис

(lambda (X Y) (/(+ X Y) 2.0))

* (apply #’(lambda…) 3 5)

#’ – функциональная блокировка, кот организовывает замыкание контекста функций

* (setf name value)



Eval:

1. S-выражение атом?
   1. Это Т?
      1. Да – печать
      2. Нет- Это Nil?
         1. Да – печать
         2. Нет – значение, если есть
   2. Нет- есть апостроф?
      1. Да- печать
      2. Нет – 1й элемент S- выраж требует спец обработки аргументов?
         1. Да – спец обработка аргументов, функция применяется к аргументам
         2. Нет – функция применяется к аргументам