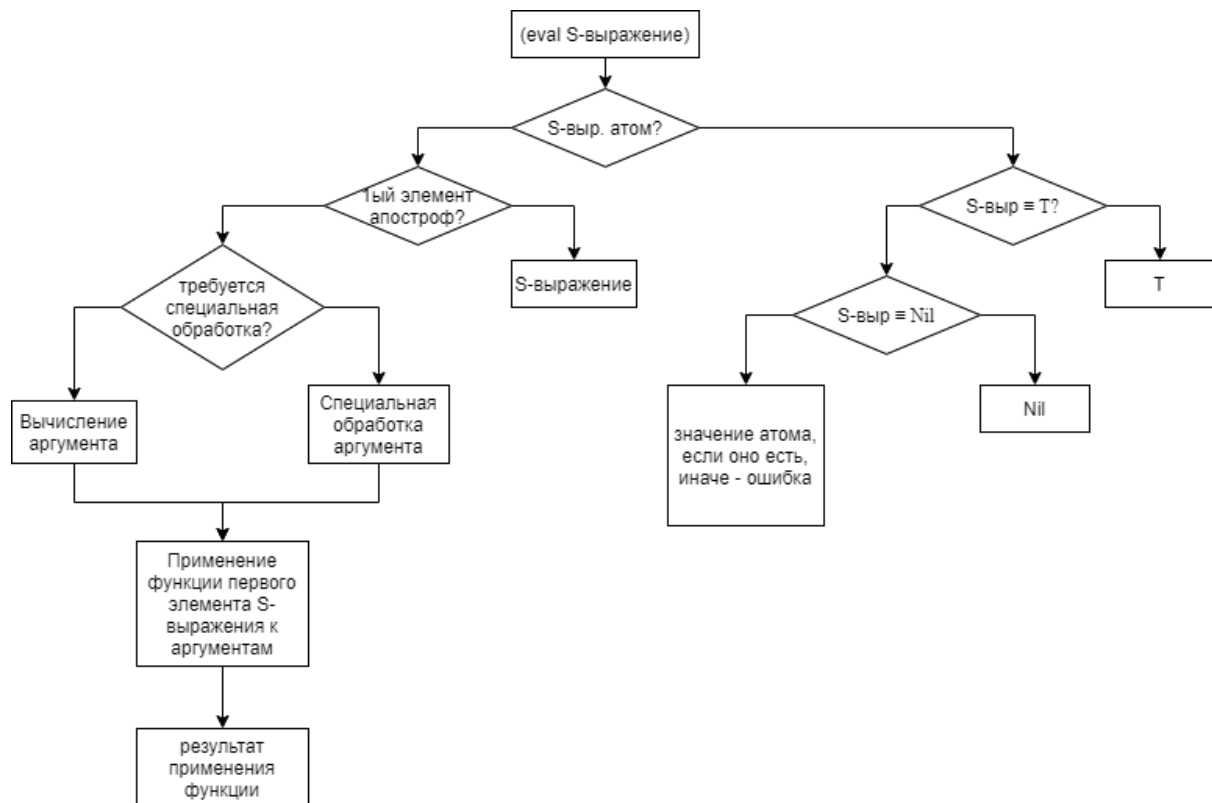


2.1) Диаграммы вычислений выражений



```

(equal 3 (abs -3)) -> (equal 3 3) -> T
(equal (+ 1 2) 3) -> (equal 3 3) -> T
(equal (* 4 7) 21) -> (equal 28 21) -> Nil
(equal (* 2 3) (+ 7 2)) -> (equal 6 (+ 7 2)) -> (equal 6 9) -> Nil
(equal (- 7 3) (* 3 2)) -> (equal 4 (* 3 2)) -> (equal 4 6) -> Nil
(equal (abs (- 2 4)) 3) -> (equal (abs -2) 3) -> (equal 2 3) -> Nil
  
```

2.2) Функция вычисления гипотенузы прямоугольного треугольника по заданным катетам, диаграмма ее вычисления

```

(defun f (ar1 ar2)
  (sqrt (+ (* ar1 ar1) (* ar2 ar2))))
  
```

```

(f 2 3) -> (sqrt (+ (* 2 2) (* 3 3))) -> (sqrt (+ 4 (* 3 3))) ->
(sqrt (+ 4 9)) -> (sqrt (+ 4 9)) -> (sqrt 13) -> 3.6055512
  
```

Вопросы

1. Базис языка

- a. Атом: символ, число (самоопределимый атом).
Символы могут обозначать числа, строки, сложные структуры, функции. Т (true, истина) и NIL (false, ложь) - зарезервированные константы, встроенные функции.

- b. Точечная пара.

`<точечная пара> ::= (<атом | точечная пара>.<атом | точечная пара>)`

Атомы и точечные пары объединяют под общим названием S-выражения.

Особым видом S-выражения является список

`<список> ::= NIL | (<S-выражение>.<список>)`

2. Классификация функций

- Чисто математические (фиксированное количество аргументов, результат только один)
- Формы (либо переменное число аргументов, либо они по-разному обрабатываются, либо обрабатываются не все)
- Функционал (принимают в качестве аргумента функцию)

1) Конструкторы(cons, list)

- a) cons - двухаргументная функция, создает одну списковую ячейку и расставляет указатели; (cons 'a 'b) -> (a . b); (cons 'a '(b)) -> (a b)
- b) list- создает список. Создает и связывает столько списковых ячеек, сколько аргументов (list 'a 'b 'c 'd) -> (a b c d)

2) Селекторы(car, cdr)

- a) car- переходит по car указателю и возвращает голову списка
- b) cdr- переходит по cdr указателю и возвращает хвост списка

3) Предикаты (позволяют определить природу элементов)

- a) atom - возвращает Т, если аргумент - атом, иначе Nil;
- b) listp - возвращает Т, если аргумент - список, иначе Nil;
- c) consp - возвращает Т, если аргумент представлен списковой ячейкой, иначе Nil;
- d) numberp - возвращает Т, если аргумент- число, иначе Nil;
- e) symbolp - возвращает Т, если аргумент- не число, иначе Nil;

4) Функции сравнения:

- a) eq - принимает два аргумента - символьные атомы. Возвращает Т, если аргументы указывают на одно и то же значение (сравнивает указатели), иначе Nil.
- b) eql - сравнивает два числа (по форме их представления)
(eql 3 3) -> Т (eql 3 3.0) -> Nil

- c) = - сравнивает два числа (= 3 3) -> T (= 3 3.0) -> T
- d) equal - сравнивает как и eql, но также может сравнивать списки
- e) equalp - сравнивает все варианты представления, но долго работает
- 5) Арифметические функции (- +, -, *, /)
- 6) Определяющие функции
 - a) (defun func_name (список аргументов) (тело функции))
 - b) (lambda (список аргументов) (тело функции))
- 7) Определяющее значение

связывают символ со значением, предварительно вычисляя значения аргументов

 - a) (set 'a 'b)

В качестве значения функция SET возвращает значение второго аргумента. Если перед первым аргументом нет апострофа, то значение будет присвоено значению этого аргумента.

 - b) (setq a 'b) = (set 'a 'b); q - обозначает quote
 - c) (setf place 'value) - в качестве первого аргумента, в отличие от предыдущих функция, может принимать не только символьный атом

(setq x '(1 2))	x = (1 2)
(setf (car x) 5))	x = (5 2)
- 8) Функционалы (принимают функцию как аргумент)
 - a) (apply #'(lambda-выражение | имя функции) arg1 arg2 arg3 ...)
 - b) (fncall #'(lambda-выражение | имя функции) (arg1 arg2 arg3 ...))

3. Представление списка в памяти

Пустой список представлен одним атомом - Nil. Не пустой список представлен одной или несколькими списковыми ячейками. Списковая ячейка состоит из двух полей - двух указателей: car и cdr. Car - указатель на голову, cdr - на хвост.

4. Car и cdr

Car - предназначена для получения первого элемента точечной пары или же головы списка. Cdr - предназначена для получения второго элемента точечной пары или же хвоста списка. Данные функции в качестве аргумента принимают точечную пару или список. Их использование возможно лишь в списочном контексте, использование для атома приведет к ошибке. Головой и хвостом пустого списка для удобства считается Nil.

5. Отличие cons и list

cons- является базисом языка, она на вход принимает ровно два аргумента и создает одну списковую ячейку
 list- написана на базе функции cons, принимает любое количество аргументов и создает список
 cons работает быстрее