Министерство образования Российской Федерации
Московский Государственный Технический Университет
им. Н.Э. Баумана

Отчет по лабораторной работе №4
По курсу «Функциональное и логическое программирование»

Студент	Медведев А.В.
Группа	ИУ7-62
Преподаватель	Толпинская Н.Б.

Москва, 2019

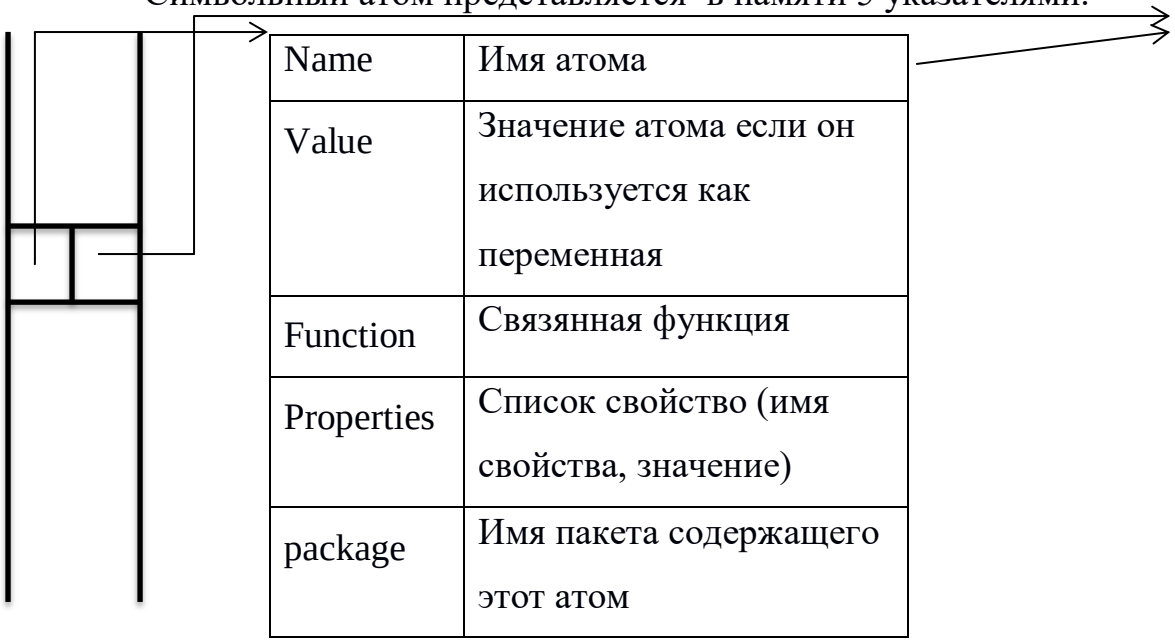
Теоретическая часть

Синтаксическое представление программы на Lisp, хранение программы в памяти

Программы на Lisp представлены в виде S-выражения. Интерпретатор Lisp читает входящие команды, имеющие вид S-выражений, вычисляет значение каждого из введённых выражений, и возвращает результат. Lisp при работе программы может изменять программу за счет использования списков. Чаще всего программы в Lisp строится из рекурсивных функций над S-выражениями. Определения и вывозы этих функций так же имеют вид S-выражений, то есть формально они могут быть обработаны как обычное данные, полученные в процессе вычислений, и преобразованы как значения. Переменные, константы, выражения ветвления, вызовы функций представляются в виде S-выражениями.

Самая простая форма выражения - переменная. Она может быть представлена как атом. Lisp-интерпретатор в ходе своей работы поддерживает специальную таблицу символьных атомов (таблица символов) в которой хранится информация обо всех атомах, встретившихся в тексте интерпретируемой программы или в качестве обрабатываемых данных. При обработке очередного символьного атома интерпретатор проверяет, занесен ли он в таблицу. Атом может иметь несколько независимых друг от друга значений. Символьный атом может использоваться как имя функции или как имя функционального параметра. Нужная интерпретация определяется из контекста его применения.

Символьный атом представляется в памяти 5 указателями.



Name	Имя атома
Value	Значение атома если он используется как переменная
Function	Связянная функция
Properties	Список свойство (имя свойства, значение)
package	Имя пакета содержащего этот атом

Пакет - это объект lisp, сопоставляющий именам символы (набор символьных атомов о которых знает интерпретатор)

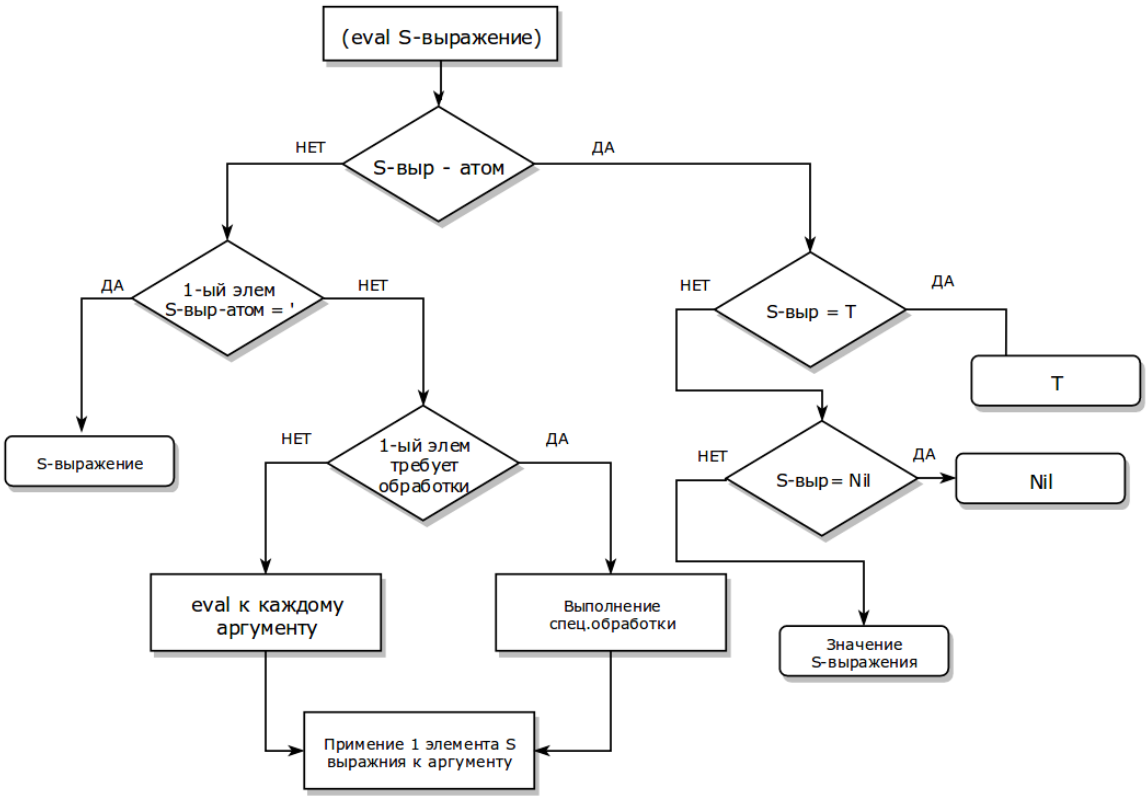
Для получения этих данных можно использовать функцию symbol_name, symbol_value, symbol_function, symbol_plist, symbol_package

Как трактуются элементы списка

Список - это особый вид S-выражения, который может быть пустым/не пустым, если он не пустой, то он имеет первый элемент (голову) и хвост (является списком). S-выражения представлены в виде точечных пар, которые состоят из унифицированных структур - блоков памяти - бинарных узлов. Каждый бинарный узел имеет небольшой объем, достаточный для хранения двух типизированных указателей (CAR и CDR, левый и правый, голова и хвост). Пара из первого элемента списка ("голова") и остальных элементов списка ("хвост") представляют собой пару указателей на точечную пару и список.

Порядок реализации программы

Реализация программы определяется базовой функцией eval, которая запускается автоматически



Практическая часть

1. **(equal 3 (abs -3))**

обработка функции equal
вычисление первого аргумента: 3
вычисление второго аргумента:
обработка функции abs
вычисление аргумента -3: -3
возврат 3
применение equal к 3 и 3
возврат T

2. **(equal (+ 1 2) 3)**

обработка функции equal
    вычисление первого аргумента:
        обработка функции +
            вычисление первого аргумента 1: 1
            вычисление второго аргумента 2: 2
            применение + к 1 и 2
            возврат 3
    вычисление второго аргумента 3: 3
    применение equal к 3 и 3
    возврат T

3. **(equal (* 4 7) 21)**

обработка функции equal
    вычисление первого аргумента:
        обработка функции *
            вычисление первого аргумента 4: 4
            вычисление второго аргумента 7: 7
            применение * к 4 и 7
            возврат 28
    вычисление второго аргумента 21: 21
    применение equal к 28 и 21
    возврат Nil

4. **(equal (* 2 3) (+7 2))**

обработка функции equal
    вычисление первого аргумента:
        обработка функции *:
            вычисление первого аргумента 2: 3
            вычисление второго аргумента 2: 3
            применение * к 2 и 3
            возврат 6
    вычисление второго аргумента:
        обработка функции +:
            вычисление первого аргумента 7: 7
            вычисление второго аргумента 2: 2
            возврат 9
    применение equal к 6 и 9
    возврат Nil

5. **(equal (- 7 3) (* 3 2))**

обработка функции equal
    вычисление первого аргумента:
        обработка функции -:
            вычисление первого аргумента 7: 7
            вычисление второго аргумента 3: 3
            применение - к 7 и 3
            возврат 4
    вычисление второго аргумента:
        обработка функции *:
            вычисление первого аргумента 3: 3
            вычисление второго аргумента 2: 2
            возврат 6
    применение equal к 4 и 6
    возврат Nil

6. **(equal (abs (- 2 4)) 3)**

обработка функции equal
    вычисление первого аргумента:
        обработка функции abs:
            вычисление первого аргумента:
                обработка функции -:
                    вычисление первого аргумента 2: 2
                    вычисление второго аргумента 4: 4
                    применение - к 2 и 4
                    возврат -2
            применение abs к -2
            возврат 2
    вычисление второго аргумента 3:3
    применение equal к 2 и 3
    возврат Nil

Написать функцию, вычисляющую гипотенузу прямоугольного треугольника по заданным катетам и составить диаграмму её вычисления.

```
(defun hyp (a b) (sqrt (+ (* a a) (* b b))))

((lambda (a b) (sqrt (+ (* a a) (* b b)))) 3 4)
```

```
(hyp 3 4)
обработка функции hyp
        вычисление первого аргумента. :3
        вычисление второго аргумента.: 4
        вызов hyp с аргументами 3 и 4
                создание переменной a со значением 3
                создание переменной b со значением 4
                обработка функции sqrt
                        обработка функции +
                                вычисление л.ч.:
                                        обработка функции *
                                                вычисление л.ч.: 3
                                        вычисление п.ч.: 3
                                        применение * к 3 и 3
                                        возврат 9
                                вычисление п.ч.:
                                        обработка функции *
                                                вычисление л.ч.: 4
                                                вычисление п.ч.: 4
                                                применение * к 4 и 4
                                                возврат 16
                                применение + к 9 и 16
                                возврат 25
                        применение sqrt к 25
                возврат 5.0
        возврат 5.0
```

Написать функцию, вычисляющую площадь трапеции по её основаниям и высоте, составить диаграмму её вычисления.

```
(defun S (a b h) (* h (/ (+ a b) 2)))
```

(S 1 3 2)
Обработка функции S
    Вычисление первого аргумента: 1
    вычисление второго аргумента: 3
    вычисление третьего аргумента: 2
    применение S к 1 3 и 2:
        создание переменной a со значением 1
        создание переменной b со значением 3
        создание переменной h со значением 2
        обработка функции *
            вычисление л.ч.: 2
            вычисление п.ч.:
                обработка функции /
                    вычисление л.ч.:
                        обработка функции +
                            вычисление л.ч.: 1
                            вычисление п.ч.:3
                            применение + к 1 и 3
                            возврат 4
                    вычисление п.ч.: 2
                    применение / к 4 и 2
                    возврат 2
            применение * к 2 и 2
            возврат 4
        возврат 4