Искусственный интеллект

Язык ЛИСП

Юлия Станиславовна Корухова

слайды **http://al.cs.msu.su/system/files/AI_LISP.pdf**

01.10.2013

Язык Лисп

LISt Processing

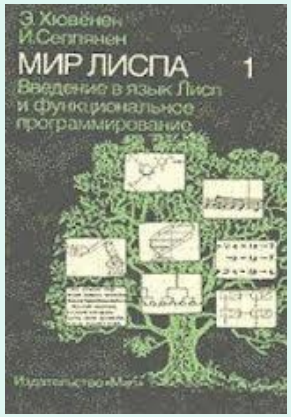
автор языка -
Джон Маккарти
(1927 — 2011)

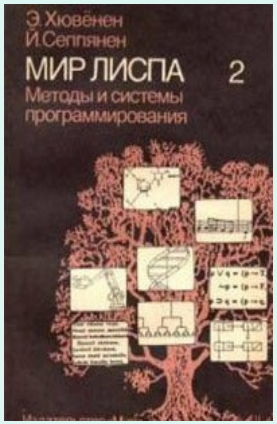
http://www-formal.stanford.edu/jmc/

***Recursive Functions Of Symbolic Expressions and Their Computation by Machine* (1960)**

Рекомендуемая литература

•**Э. Хювенен, Й. Сеппянен**
***Мир Лиспа*,**
М., Мир, 1990





• **Е.И. Большакова, Н.В. Груздева**
***Основы программирования на языке Лисп*,**
М., Издательский отдел факультета ВМК, 2010
http://www.recyclebin.ru/BMK/LISP/bolshakova.html

• **М.Ю. Семенов *Язык Лисп для персональных ЭВМ***
Издательство Московского университета, 1989
http://www.recyclebin.ru/BMK/LISP/semenov.html

План лекции

– Функциональное программирование на Лиспе

– Базовые понятия языка

– Основные функции

– Определение новых функций. Рекурсия

– Примеры программ

Функциональное программирование

3 основные конструкции:

– применение функции

– условное выражение

– рекурсия

Базовые понятия: атомы

Атомы = символьные атомы + числа

Символьные атомы представляют объекты, состоят из букв, цифр, специальных знаков:

+ - ; / " % ^ < > ~ пробел

могут быть включены и другие символы: | символ |

\ символ

Числа (не могут представлять лисповские объекты, кроме самих себя)

746 -3.14 3.055E8

Логические значения T и NIL

Базовые понятия: списки

Список — упорядоченная последовательность, элементами которой являются либо атомы, либо списки.

(a (b c) d)

(+ 2 3)

(((((первый) 2) третий) 4) 5)

(кот-37 (кличка Петя) (цвет ?) (хвост nil))

Пустой список: nil NIL ()

Атомы и списки называются **s-выражениями** (символьными выражениями)

Полезное свойство списков

Списки можно использовать для представления знаний.

```
(лекция-45
    (вид_занятия   лекция)
    (курс искусственный_интеллект)
    (тема  язык_Лисп)
    (аудитория П-13)
    (студенты_кафедр (АСВК АЯ СП КИ)
)
```

Представление списков в памяти

Списки - ссылочные структуры

Точечная пара состоит из указателей
на 1й и 2й элементы

(A . B)

(A B C D) это (A . (B . (C . (D . ()))))

Функции в Лиспе

"Чистый" Лисп — 8 функций

(car L)

(cdr L)

(cons a L)

(atom a)

(eq a b)

(quote a) или 'a

(eval a)

(cond (p1 e1) (p2 e2) … (pn en)), n>=1

Встроенные функции

28 функций caar, cadr, cdar, cddr, ….. , cddddr

list составляет список из значений своих аргументов

арифметические функции + - * /

предикаты сравнения = /= > >= < <=

логические функции not and or

(null L)

Определение новых функций

(defun имя_функции λ-список тело_функции)

Пример:

```
(defun list2 (x y) (cons (x (cons  y  () ) )))

(list2 'X 'Y) ===> (X Y)

(list2  (car (X Y Z)) 12) ===> (X 12)


(defun abs (x)
          (cond  ((< x 0) (- 0 x))
                 (T  x)))
```

Рекурсия

Вычисление длины списка

```
(defun length (L)
  (cond ((null L) 0)
        (T (+ 1 (length (cdr L))))))
```

Поиск элемента в списке

```
(defun member (x L)
   (cond ((null L) NIL)
         ((eq x (car L)) T)
         (T (member x (cdr L)))))
```

Рекурсия (продолжение)

Конкатенация списков

```
(defun append (L1 L2)
  (cond ((null L1)  L2)
        (T (cons (car L1) (append (cdr L1) L2)))))
```

Интерпретатор Лиспа на Лиспе

```
;запись в метанотации Лиспа.

evalquote[fn;x] = apply[fn;x;NIL]
apply[fn;x;a] =
     [atom[fn] -> [eq[fn;CAR] -> caar[x];
                   eq[fn;CDR] -> cdar[x];
                   eq[fn;CONS] -> cons[car[x];cadr[x]];
                   eq[fn;ATOM] -> atom[car[x]];
                   eq[fn;EQ] -> eq[car[x];cadr[x]];
                   T -> apply[eval[fn;a];x;a]];
      eq[car[fn];LAMBDA] -> eval[caddr[fn];parlis[cadr[fn];x;a]];
      eq[car[fn];LABEL] -> apply[caddr[fn];x;cons[cons[cadr[fn];
                                 caddr[fn]];a]]]


eval[e;a] = [atom[e] -> cdr[assoc[e;a]];
     atom[car[e]] ->
      [eq[car[e],QUOTE] -> cadr[e];
      eq[car[e];COND] -> evcon[cdr[e];a];
      T -> apply[car[e];evlis[cdr[e];a];a]];
      T -> apply[car[e];evlis[cdr[e];a];a]]


evcon[c;a] = [eval[caar[c];a] -> eval[cadar[c];a];
      T -> evcon[cdr[c];a]]


evlis[m;a] = [null[m] -> NIL;
      T -> cons[eval[car[m];a];evlis[cdr[m];a]]]
```