Introduction à LISP

1 L'histoire de LISP

- Langage de programmation conçu à la fin des années 60 (John Mc Carthy)
- Langage fonctionnel: toute expression rend un résultat comme une fonction mathématique. un programme est une collection de fonctions
- Langage interprété: chaque expression est lue, évaluée et retourne un résultat
- Traite des listes (LISt Processor)

2 Les objets manipulés

* **ATOMES**
  Un atome peut être:
  - un nombre: entier ou réel
  - une chaîne de caractères.
    Ex: "toto ", "il fait beau aujourd'hui"
  - un symbole
    Ex: X13, toto
  - *NIL*
  - *T* (vrai)
* LISTES
  une liste est :
  - soit la liste vide notée *()* ou *NIL*
  - soit un doublet composé d'un premier élément appelé le **car** de la liste et d'un second élément appelé le **cdr** de la liste qui est une liste.
  Une liste contenant n éléments se note par la suite de ces n éléments encadrés par des**( )**.
  Ex: (1 2 3)
  (a b c)
  ((1 a) (2 b) (3 c))
  (1 ("bonjour" "vous ") 3)

3 Evaluation d'une expression

Toute expression soumise à l'interpréteur Lisp est évaluée et retourne une valeur (ou message d'erreur).

- Un atome, qui n'est pas un symbole, est évalué à lui-même.
  Ex:
  \> 2
  2
  \> ()
  NIL
  \> ”bonjour”
  ”bonjour”

- Une expression évaluable qui n'est pas un atome est de la forme:

  (nom_de fonction expression_évaluable$_1$ … *expression_évaluable*$_n$*)*

  Cette expression s'évalue en appliquant la fonction *nom_fonction* aux n arguments résultant de l'évaluation de *expression_évaluables*$_1$ … *expression_évaluable*$_n$. La fonction doit donc avoir pour arité n!
  Ex:
  \> (+ 1 1)
  2
  \> (* (+ 9 8) 3)
  51

- La fonction **quote** interdit l'évaluation de son argument.
  Ex:
  \>(quote (+1 1))
  (+ 1 1)
  \>(1 2 3)
  ** - EVAL: 1 n'est pas un nom de fonction.
  \> (quote (1 2 3))
  (1 2 3)

Remarque : l'utilisation de la fonction quote peut être abrégée avec **'** : '(1 2 3) ≡ (quote (1 2 3))

4 Quelques fonctions...

- **Fonctions numériques**
  Les fonctions +, -, *, /, ... sont utilisables sur les entiers et les réels.

- **Fonctions booléennes**
  La valeur vrai se note T et faux se note *NIL*.
  Les fonctions booléennes sont *and, or, not*
  La fonction *(atom a)* renvoie T si a est un atome
  La fonction *(null l)* renvoie T si l est une liste vide

- **Comparaisons**
  Les fonctions $=, <, > =<, >=$ sont utilisables sur les entiers, les réels.
  La fonction *equal* teste l'égalité de deux objets (listes ou atomes.

La fonction *eq* teste l'égalité de deux atomes.(renvoie NIl s'il ne s'agit pas d'atomes)
Ex:
> (equal '(1 2) '(1 2))
T
> (eq '(a b) '(a b))
NIL

- **Gestion de listes**
  car renvoie le premier élément d'une liste
  cdr renvoie la liste privée de son premier élément.
  car et *cdr* ne s'appliquent qu'à des listes.

  Ex:
  > (equal (car '((b c))) (cdr '(a b c)))
  T
  La fonction *cons* construit une liste dont le premier élément est e et dont le "cdr" est l.
  Ex:
  Pour construire la liste (1 2 3) on peut écrire :
  > (cons 1 (cons 2 (cons 3 ())))
  (1 2 3)
  ou encore
  > (cons 1 '(2 3))
  (1 2 3)
  (question à poser aux étudiants: Pourquoi met-on une quote ici?)
  > (cons 'a '(b c))
  (A B C)
  > (cons '(1) (2 3))
  ((1) 2 3)
  > (car(cons 1 '(2 3)))
  1
  > (cdr(cons 1 '(2 3)))
  (2 3)
  > (cons (car '(a b c)) (cdr '(a b c)))
  (A B C)
  > (cons '(a b c) nil)
  ((A B C))