Programmation fonctionnelle
Le langage LISP

D. Le langage LISP

- . Objets
- . Structure d'un programme
- . Primitives de manipulation des listes
- . Primitives booléennes
- . Définition de nouvelles fonctions
- . Fonctionnelle (Fonction de fonctions)
- . Exemples

E. Fonctionnement de l ’interpréteur Lisp

- . Forme interne
- . Algorithme d'interprétation.

Programmation fonctionnelle
Le langage LISP

Objets

- **LISP : LISt Processor (1960)**

- **Les objets peuvent être**

— **atomes (symbole, nombre)**

— **liste '(' < suite d'objets >')'**

Programmation fonctionnelle
Le langage LISP

Structure d'un programme :

- **Toute donnée et tout programme Lisp est une S-expression(liste) engendrée par la grammaire suivante :**

  <Liste> --> <atome> ! (<Liste> , <Liste>)
- **Les programmes LISP sont des listes particulières où le premier terme représente une fonction. les termes suivants sont les paramètres d'appel de cette fonction.**

  Exemple: (ceci est une fonction) représente l'application de la fonction ceci aux paramètres est, une et fonction)
- **(Sin x) c'est sin(x)**
- **Lisp utilise donc la notation préfixée : (+ 3 2) c'est 3 + 2.**

Programmation fonctionnelle
Le langage LISP

Quelques fonctions utiles

- **Conditionnelle : (IF cond liste1 liste2 ..listen)**
- **Évalue suivant la valeur de Cond soit liste1 soit en "séquentielle" liste2, liste3, ..., listen.**
- **La valeur retournée est celle de la dernière forme évaluée.**
- **On se limite a l'utilisation de (If Cond liste1 liste2).**
- **<u>Valeurs de vérité :</u>**

  T pour vrai

  Nil et 0 pour faux
- **T et Nil sont des atomes spéciaux.**

Programmation fonctionnelle
Le langage LISP

Quelques fonctions utiles

- ***QUOTE*** **permet de distinguer les atomes spéciaux des constantes caractères dans les S-expressions**
- ***(Quote objet)*** **c'est retourner l'objet sans évaluation**

- **Exemple :**

(Quote (+ 1 2)) c'est équivalent à (+ 1 2)

Primitives de manipulation des listes :

Programmation fonctionnelle
Le langage LISP

Quelques fonctions utiles

- ***(Car Listenonvide)*** **: fournit la première composante de la liste.**

- **Exemple :**
  (Car '(1 2 3)) retourne 1.
  (Car '((a b) 2 (3 c))) retourne (a b).

Programmation fonctionnelle
Le langage LISP

Quelques fonctions utiles

- ***(Cdr Listenonvide)*** **: fournit listenonvide privée de son premier élément.**

- **Exemple s:**

(Cdr '(1 2 3)) retourne (2 3).
(Cdr '((a b) 2 (3 c))) retourne (2 (3 c))
(Cdr '(1)) retourne 0.

Programmation fonctionnelle
Le langage LISP

Quelques fonctions utiles

- ***(Cons objet1 Liste)*** **: retourne une liste dont le premier terme est son premier argument (objet1) et les termes suivants sont ceux du second argument.**
- **Exemples :**

  (Const '1 '(2 3 4)) retourne (1 2 3 4)

  (Const '(1 2) '(3 4 5)) ((1 2) 3 4 5)

- **Propriétés :**

  (Car (Cons x y)) = x

  (Cdr (Cons x y)) = y

Programmation fonctionnelle
Le langage LISP

Primitives booléennes :

- ***(Atom objet)*: retourne T si objet est un atome, Nil autrement.**
- ***(NumberP obj) :* retourne T ssi obj est un nombre, Nil autrement.**
- ***(SymbolP Obj) :* retourne T ssi obj est un symbole, Nil autrement.**
- ***(Consp Obj) :* retourne T ssi Obj est une liste non vide, Nil autrement.**
- ***(eq Symb1 Symb2)* : teste l'égalité de 2 atomes.**
- ***(Equal Obj1 Obj2)* : égalité de 2 objets.**
- ***(Null Obj) :* retourne T ssi Obj est une liste vide, Nil autrement.**

Programmation fonctionnelle
Le langage LISP

Définition de nouvelles fonctions :

- **(DE Nomdefonction (Var1 Var2 ..Varn) Liste1 Liste2 ... Listen)**
- **Retourne Nomdefonction comme valeur.**

Programmation fonctionnelle
Le langage LISP

Fonctionnelle (Fonction de fonctions)

- **Valeur nominale et fonctionnelle d'un symbole :**
- **Quand on définit une fonction par :**
- **(DE f(x) (...))))**
- **on dit que l'on définit sa valeur fonctionnelle**
- **Quand on définit une fonction de fonction par :**
- **(DE g(f) (...)))**
- **on dit que f a deux valeurs :**
- **- une valeur nominale (provenant de l'appel)**
- **- une valeur fonctionnelle (celle de la fonction f(x)**

Programmation fonctionnelle
Le langage LISP

Fonctionnelle (Fonction de fonctions)

- **On peut définir des fonctions ayant d'autres fonctions comme argument. Le principe est d'utiliser la valeur nominale d'un symbole comme nom de fonction à l'aide de la fonction Funcall définie comme suit :**

- **(Funcall Symbol Arg1 Arg2 ...Argn)**
- **La valeur nominale de Symbol doit être le nom d'une fonction**

Programmation fonctionnelle
Le langage LISP

Fonctionnelle (Fonction de fonctions)

- **Construire une fonction H qui étant donnée une fonction f et une liste X=(x1 x2 xn), retourne une liste Y=(y1 y2 yn) tel que yi=f(xi)**

Programmation fonctionnelle
Le langage LISP

Fonctionnelle (Fonction de fonctions)

```
(De H (f x)
(if (consp x)
  (cons (funcall f (car x))
        (H f (cdr x))
)))
```


- **Exemple:**

si on définit g par (de g(x) (* 3 x)) et si on fait appel à H par (H 'g '(1 2 3)) on obtient (3 6 9). Par contre, si on supprime Funcall c'est f qui est considérée mais pas g.

Programmation fonctionnelle
Le langage LISP

Exemple 1 : Longueur d'une liste.

```
(DE Long(L)
        ( IF(Null L) 0
          (+ (Long ( Cdr L)) 1 )
        )
)
```

Programmation fonctionnelle
Le langage LISP

Exemple 2 : Somme des éléments d'une liste.

```
(DE Som1(L)
        (IF (Consp L)
      (+ (Car L)  (Som1  Cdr L))
        0
        )
)
```

Programmation fonctionnelle
Le langage LISP

Exemple 2 : Somme des éléments d'une liste.

- **Ou bien :**

```
(DE Som2(L)
        (IF (Consp L)
        (+ (Som2 (Car L))
           (Som2 (Cdr L)))
        (IF (NumberP L) L
                    0
        )
        )
)
```

Programmation fonctionnelle
Le langage LISP

Exemple 3 : Appartenance d'un atome à une liste.

```
(DE Exist (a L)
        (IF (Consp L )
        (Or (Exist a (Car L))
            (Exist a (Cdr L)))
        (Eq a L)
        )
)
```

Programmation fonctionnelle
Le langage LISP

Exemple 4 : Inverser une liste d'atomes.

```
(De Inverse(L)
(Cons
(Dernier L)
(Inverse(Supprimer(Dernier L) L)
)
)
```

- **C'est le dernier concaténé à l'inverse de la liste sans le dernier.**

Programmation fonctionnelle
Le langage LISP

Exemple 4 : Inverser une liste d'atomes.

```
(De Dernier(L)
(IF (Consp L)
   (IF (Null Cdr(L) )
         (Car L)
         (Dernier (Cdr L)))))
```

Programmation fonctionnelle
Le langage LISP

Exemple 4 : Inverser une liste d'atomes.

```
(DE Supprimer(a L)
(IF (Consp L)
        (IF (Eq a Car(L))
        Cdr(L)
        Cons( Car(L)
          Supprimer(a (Cdr(L)))
        )
)
)
```

- **Si c'est le premier élément, le résultat est le reste de la liste, sinon c'est ce premier élément concaténé au reste de la liste sans a.**

Programmation fonctionnelle
Le langage LISP

Exemple 5 : fonctionnelle

Construire une fonction g qui retourne la valeur d'une fonction f à deux arguments, appliquée aux élément d'une liste X = (x1 x2 xn) de la manière suivante :

g(f, x) = f(x1, f(x2, f(x3, ...,f(xn-1, xn))))

Programmation fonctionnelle
Le langage LISP

Exemple 5 : fonctionnelle

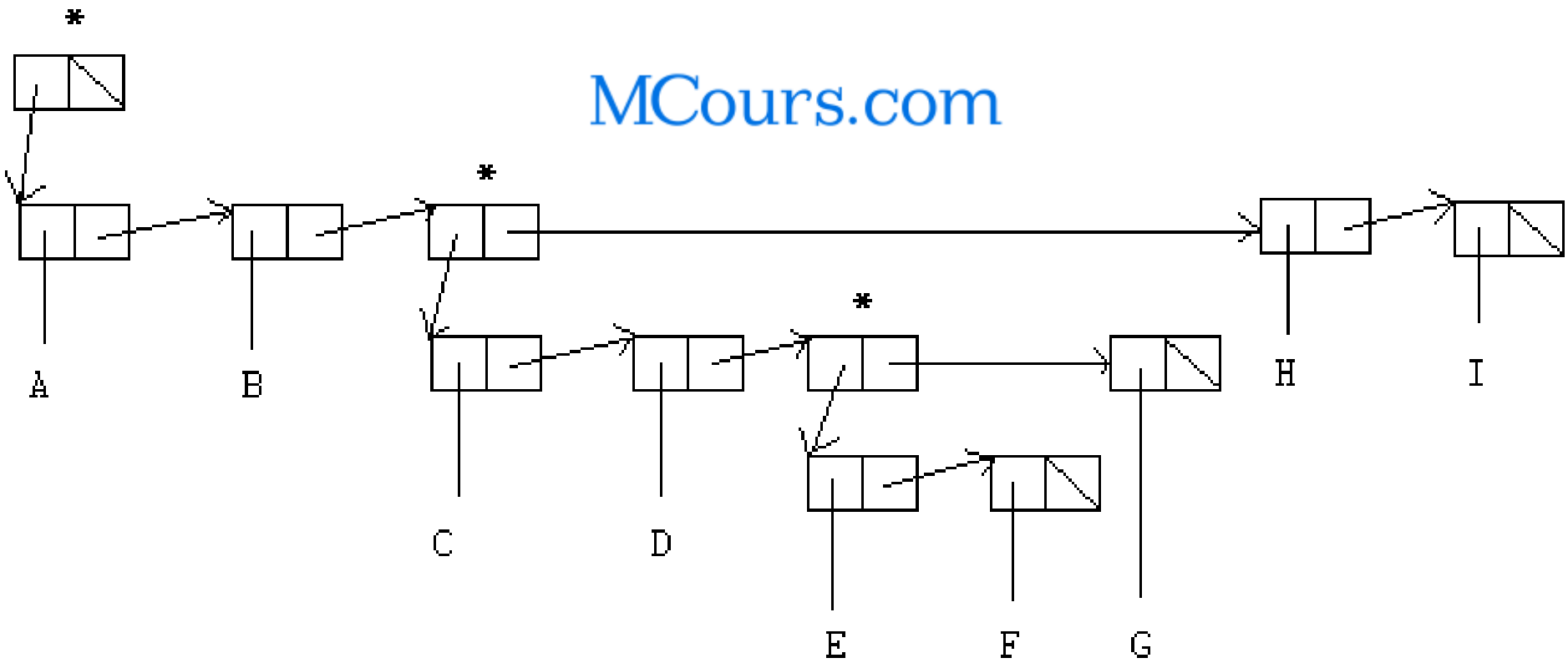
```
(DE g ( f x)
( if    (consp x)
(if (null cdr (cdr x)))
(Funcall f (car x)(car( cdr x)))
(funcall f (car x)(g f (cdr x)))
)))
```

Programmation fonctionnelle

Fonctionnement de l'interpréteur LISP

Forme interne

Le programme Lisp (A B (C D (E F) G) H I) est représenté en mémoire sous la la forme d'une liste comme suit :



Programmation fonctionnelle
Fonctionnement de l'interpréteur LISP

Forme interne

- **Un maillon de la liste est une paire de pointeurs(doublet).**
- **Les listes sont créées a chaque rencontre de parenthèse ouvrante.**
- **On associe aux atomes des listes spéciales(P-listes) qui contiennent les propriétés de l'atome.**

Programmation fonctionnelle
Fonctionnement de l'interpréteur LISP

Forme interne

- **Afin de ne pas dupliquer des atomes identiques, il est nécessaire de garder tous les atomes dans une table (nom de l'atome, adresse de la p-liste correspondante)**
- **On distingue :**

— **les cellules de données où le premier champ pointe vers la P-liste associé à l'atome et le deuxième champs vers la cellule suivante (de données ou d'accrochage)**

— **les cellules d'accrochage où le premier champ pointe vers une nouvelle liste et le deuxième champs vers la cellule suivante (de données ou d'accrochage)**

- **Dans le but de savoir si le premier champs pointe une p-liste ou un autre doublet, on convient de rajouter un autre champ "PL".**

Programmation fonctionnelle
Fonctionnement de l'interpréteur LISP

Forme interne: (Structure d'une P-liste)

- **Contient 4 champs :**
  - **Indicateur,**
  - **Nom-variable,**
  - **Valeur-variable,**
  - **Pointeur**

Programmation fonctionnelle
Fonctionnement de l'interpréteur LISP

Forme interne : Informations à la phase d'analyse syntaxique.

- **Si atome numérique**

  Indicateur = 0 **Nom-variable = ***

  Valeur-nombre = valeur

  Pointeur = Nil

- **Si atome mot-clé**

  Indicateur = 2 **Nom-variable = &**

  Valeur-nombre &

  Pointeur = Nil

- **Si atome littéral**

  Indicateur = 1 **Nom-variable = nom de la variable***

  Valeur-nombre est à blanc

  Pointeur = Nil

Programmation fonctionnelle
Fonctionnement de l'interpréteur LISP

Forme interne : Informations à l ’exécution

- **Si atome littéral et désigne une variable simple, l'indicateur sera forcé à 3 et le champs Valeur-variable sera rempli par la valeur de la variable.**

- **Si atome littéral et désigne une liste, l'indicateur sera forcé à 4, le champs Valeur-nombre reste à blanc. Le champ pointeur recevra le pointeur du premier doublet de la liste nouvellement créée (liste d'atomes numériques)**

Programmation fonctionnelle
Fonctionnement de l'interpréteur LISP

Algorithme d'interprétation :

- **Parcourir la liste**
- **Pour chaque cellule de données rencontrée, on teste le code-indicateur de p-liste.**
- **Si ce dernier est positionné à "PL", alors le pointeur pointe effectivement vers une p-liste.**
- **A ce moment là, un autre indicateur(premier champ de la p-liste) sera testé.**
- **Il nous permet de distinguer un mot-clé, d'une constante ou d'une variable(simple ou liste)**
- **Selon le cas empiler la valeur de l'élément courant soit dans une *pile de fonction* soit dans une *pile d'atomes***

Programmation fonctionnelle
Fonctionnement de l'interpréteur LISP

Algorithme d'interprétation :

```
Si code-indicateur = "PL" :
    accéder l'indicateur du premier champ de la p-liste
   Si indicateur=2 :
       si valeur-motcle= 'setq"
            Fct := "setq"
            Num := 2
            Empiler(Fct, Num)
            retour
       Fsi
       Si ...
              ...
                ..
```

Programmation fonctionnelle

Fonctionnement de l'interpréteur LISP

Algorithme d'interprétation :

```
Sinon
Si indicateur = 0
        empiler(constante)
Sinon
        empiler (nom de variable)
Fsi
fsi
```

Programmation fonctionnelle

Fonctionnement de l'interpréteur LISP

Algorithme d'interprétation :


```
sinon
        dépiler (Fct, num)
        Si num = 1
                dépiler (atome)  ;   Appliquer
        sinon
       si num = 2
                depiler(v1)  ;  depiler(v2) ;  Appliquer
          sinon
                depiler (v1); depiler(v2);  depiler(v3);  appliquer
fsi
```