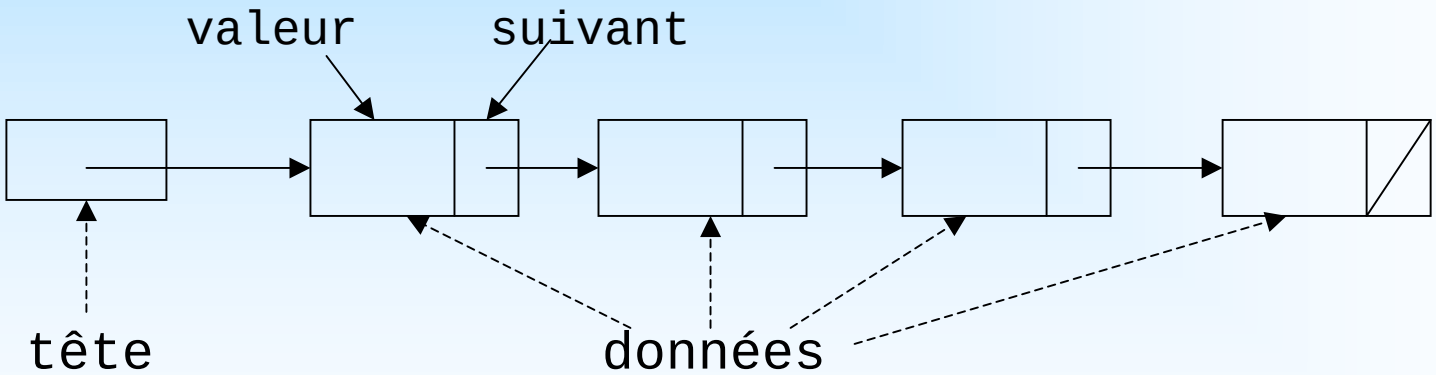
Types récursifs : les listes

Chapitre 11

Les listes

Une liste est une structure de données qui relie des objets de type identique (et quelconque)



Le type liste d'entiers

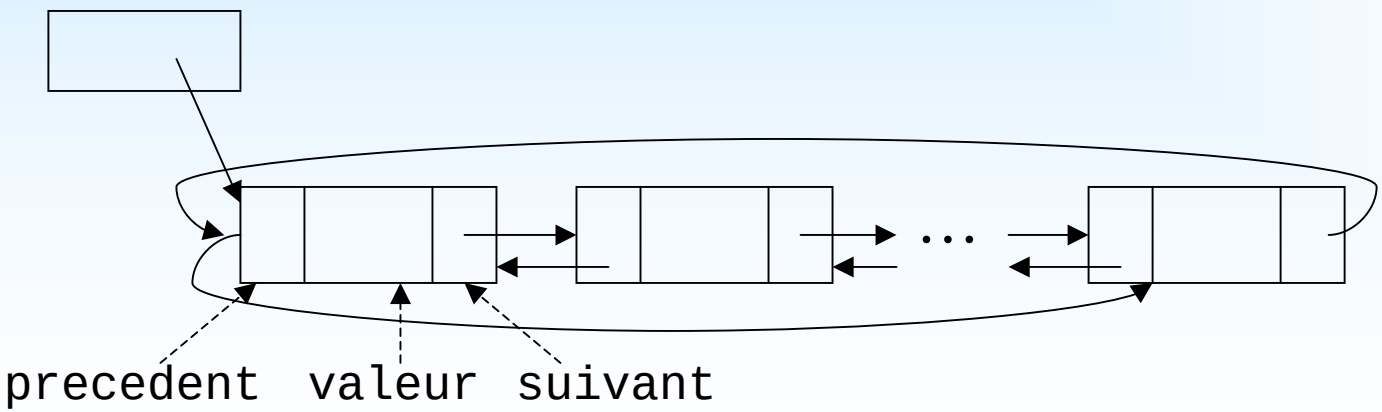
```
type Cellule;
type Liste_int is access Cellule;
type Cellule is
  record
    valeur : Integer;
    suivant: Liste_int;
  end record;
```

Le type `Cellule` est d'abord déclaré. Son nom est introduit dans l'environnement associé à une valeur indéfinie

Le type `Liste_int` est défini récursivement

Listes doublement chaînées

```
type Cellule;
type Liste_int is access Cellule;
type Cellule is
  record
    valeur    : Integer;
    suivant   : Liste_int;
    precedent: Liste_int;
  end record;
```



Intérêt

Comment rassembler en une unique structure, des données dont le nombre sera déterminé dynamiquement ?

Exemple: programme qui parcourt un texte et ajoute chaque mot nouveau à une liste. Si ce mot est déjà présent, il incrémente son nombre d'occurrences.

Exemple : typage

On suppose que le paquetage `ada.strings.unbounded` est présent dans l'environnement

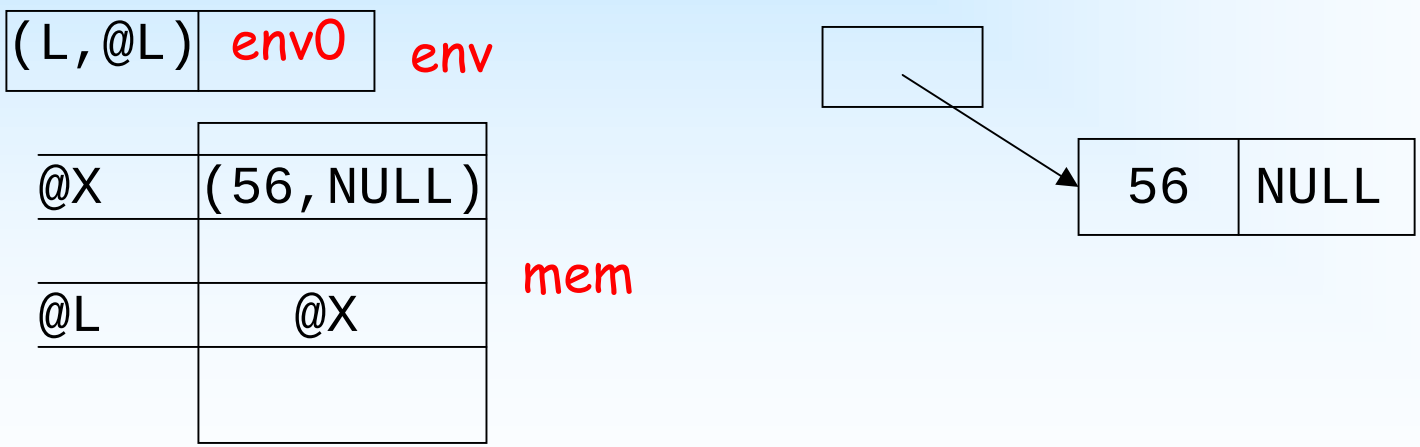
```
type Donnee is
   record
     mot : ada.strings.unbounded.Unbounded_String;
     occ : Natural:=0;
   end record;
type Cellule;
type Liste is access Cellule;
type Cellule is
   record
     valeur : Donnee;
     suivant: Liste;
   end record;
```

Manipulation de listes en Ada (1/3)

```
L:Liste_int:= NULL;
L:= new Cellule;
L.valeur:=56;
-- le champ suivant est initialisé à NULL par défaut
```

Ou bien

```
L:= new Cellule'(56,NULL);
```



Manipulation de listes en Ada (2/3)

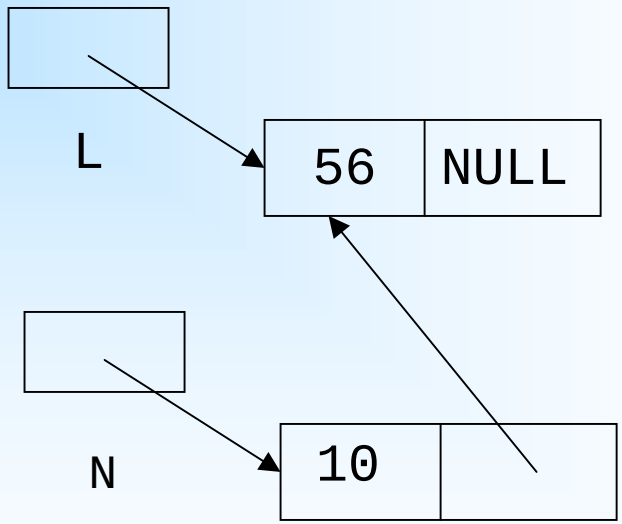
```
N:Liste_int:= new Cellule'(10,L);
```

env

(N,@N)	(L,@L)	env0

@Y	(10,@X)
@X	(56,NULL)
@L	@X
@N	@Y

mem



Manipulation de listes en Ada (3/3)

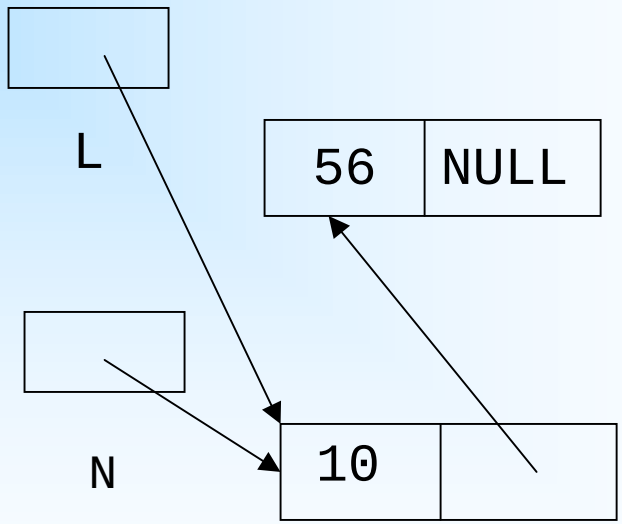
```
L:=N;
-- copie des pointeurs et non des objets
```

env

(N,@N)	(L,@L)	env0

@Y	(10,@X)
@X	(56,NULL)
@L	@Y
@N	@Y

mem



Copie d’objets

`L` désigne le pointeur et `L.all` l’objet pointé

Pour accéder à un champ de l’objet, on utilise l’opérateur d’accès : `.`

```
L.all.valeur:= N.all.valeur;
L.all.suivant:= N.all.suivant;
```

La simplification suivante est possible

```
L.valeur:= N.valeur;
N.suivant:= N.suivant;
```

Copie d’un objet référencé dans un autre

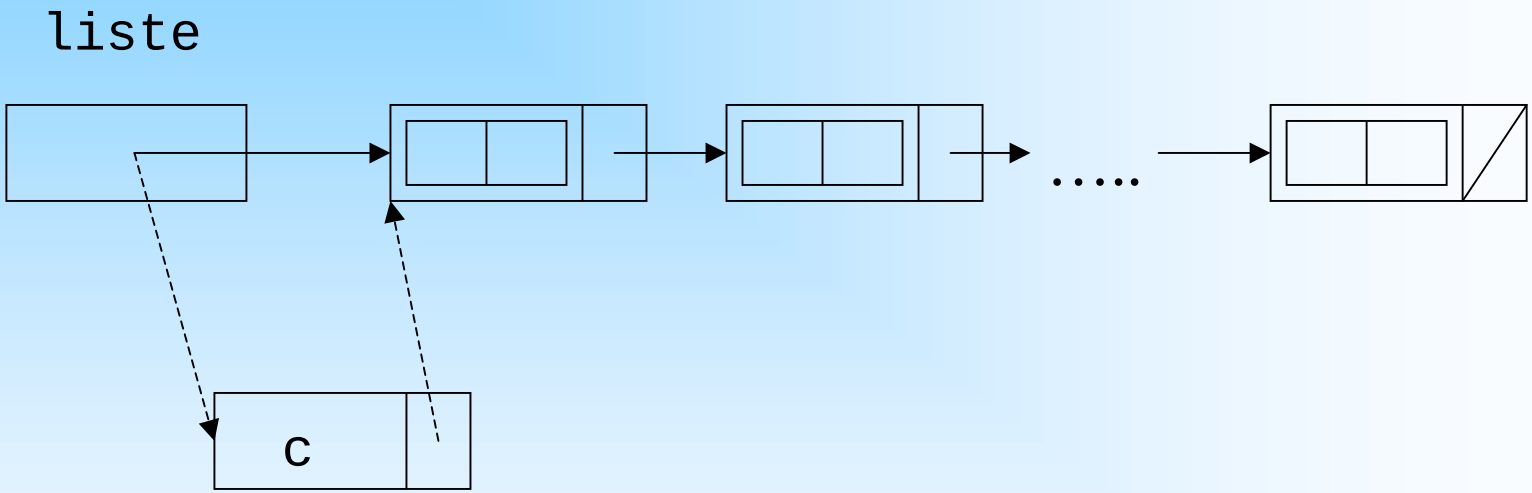
```
L.all:= N.all;
```

Comparaisons

```
L = N
-- vaut true si L et N réfèrent le même objet
L.all = N.all
-- vaut true si les objets pointés par N et L ont
-- même valeur
L = NULL
-- vaut true si la liste L est vide
```

```
type Complexe is
 record
  re,im : Float;
 end record;
type Cellule;
type ListeComplexe is access Cellule;
type Cellule is
 record
  valeur  : Complexe;
  suivant : ListeComplexe;
 end record;
procedure ajouter
       (c:in Complexe;liste:in out ListeComplexe) is
begin
 liste:= new Cellule'(c,liste);
end ajouter;
```

ajouter(c:in Complexe; liste:in out ListeComplexe)



Exemple (1/10)

Programme qui dresse la liste des mots d'un texte accompagnés de leur nombre d'occurrences

On considère (pour simplifier) que le texte ne compte qu'un mot par ligne.

Exemple (2/10) :typage

```
type Donnee is
   record
     mot : ada.strings.unbounded.Unbounded_String;
     occ : Natural:=0;
   end record;
type Cellule;
type Liste is access Cellule;
type Cellule is
   record
     valeur : Donnee;
     suivant: Liste;
   end record;
```

Exemple (3/10) : algorithme

```
début
  ouvrir le fichier texte en lecture;
  tant que
    la fin du fichier n'est pas atteinte
  faire
    lire un mot;
    ajouter ce mot à la liste;
  fin faire;
  fermer le fichier;
  afficher la liste obtenue;
 fin.
```

Exemple (4/10) : raffinement

2 procédures apparaissent :

- Ajouter une donnée à la liste.
- `procedure ajouter(l:in out Liste;d:in Donnee);`
- Afficher la liste
- `procedure put(l:in Liste);`

Déclarations

```
laListe:Liste;
motMax:String(1..80);
long:Natural;
f:FILE_TYPE;
```

Exemple (5/10) : programme

```
begin
 open(f,IN_FILE,"texte.txt");
 while not END_OF_FILE(f) loop
   get_line(f,motMax,long);
   declare
     d:Donnee:=
        (to_unbounded_string(motMax(1..long)),0);
   begin
     ajouter(laListe,d);
   end;
 end loop;
 close(f);
 put(laListe);
end mots;
```

Exemple (6/10) : commentaires

Un fichier texte est une suite de lignes séparées par un caractère de contrôle (RC)

La procédure `open` positionne le pointeur du fichier au début de celui-ci. Le type d'accès est précisé (ici `in_file`)

La fonction `end_of_file` retourne `true` si le pointeur a atteint la marque de fin de fichier(^D)

Exemple (7/10) : **ajouter**

```
début
si la liste n'est pas vide
alors
 si le mot lu est le premier de la liste
 alors
  incrémenter (de 1) le nombre d'occurrences de ce mot;
 sinon ajouter la donnée à la queue de la liste;
 fin si;
sinon
 créer une nouvelle cellule initialisée avec la donnée;
 chaîner la nouvelle cellule en tête de la liste;
fin si;
fin;
```

Exemple (8/10) : code de **ajouter**

```
procedure ajouter(l:in out Liste;d:in Donnee) is
   n:Liste:= new Cellule;
begin
   if not vide(l)
   then
     if enTete(d.mot,l)
     then
       l.valeur.occ:= l.valeur.occ+1;
     else
       ajouter(l.suivant,d);
     end if;
   else
     n.valeur.mot:= d.mot; n.valeur.occ:= 1;
     n.suivant:= NULL; l:= n;
   end if;
end ajouter;
```

Exemple (9/10) : code de **put**

```
procedure put(l:in Liste) is
      x:Liste:=l;
   begin
      put_line("---- affichage de la liste ----");
      while not vide(x) loop
         put(to_string(x.valeur.mot));put(", ");
         put(x.valeur.occ,WIDTH=>3);
         new_line;
         x:=x.suivant;
      end loop;
   end put;
```

Exemple (10/10) : résultats

Fichier texte

```
Un
arbre
binaire
est
soit
vide
soit
possede
un
sous
arbre
binaire
gauche
et
un
sous
arbre
binaire
droite
```

```
---- affichage de la liste
un, 3
arbre, 3
binaire, 3
est, 1
soit, 2
vide, 1
possede, 1
sous, 2
gauche, 1
et, 1
droite, 1
```

Arbre binaire

Structures naturellement adaptées à une grande variété d'algorithmes

- Recherche du meilleur coup à jouer (go,échecs)
- Recherche d'une donnée associée à une clé

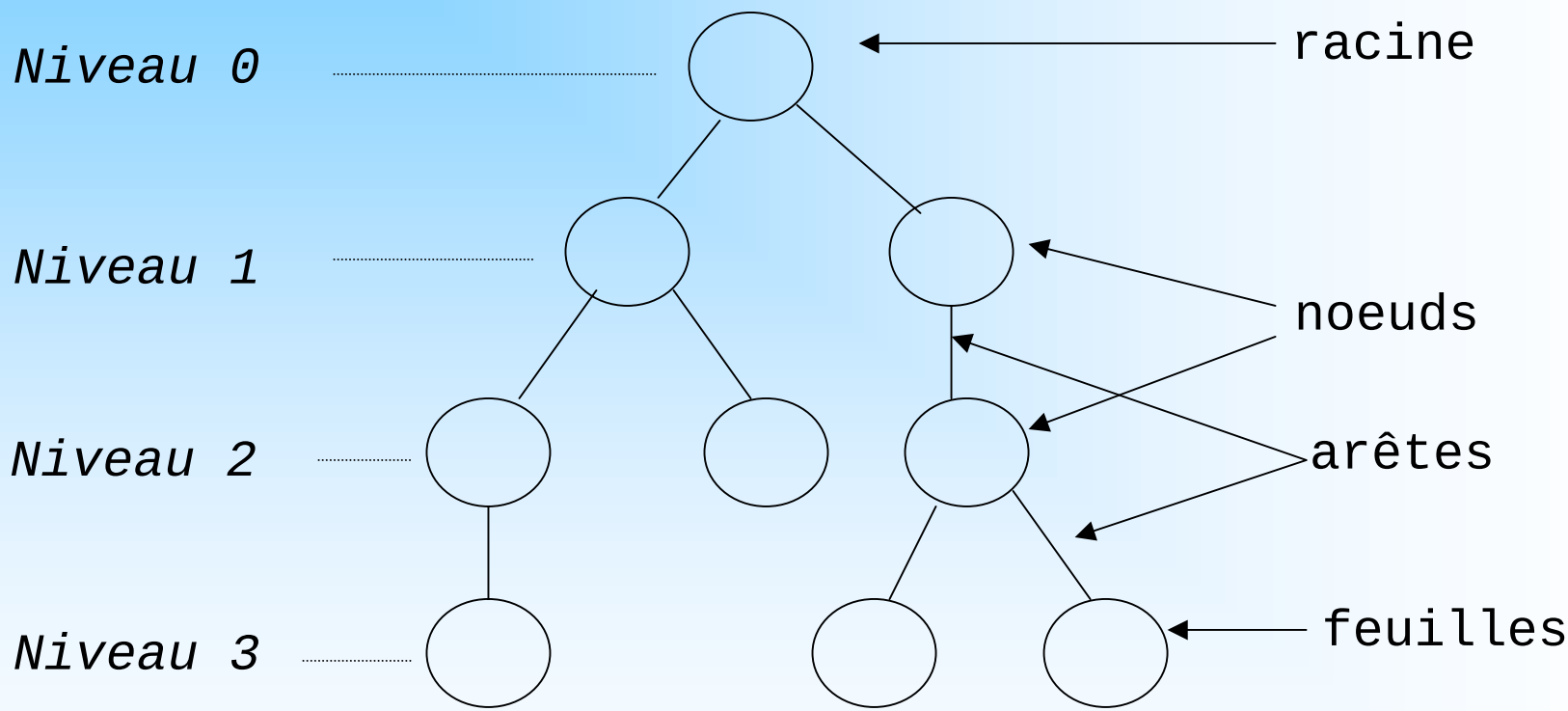
Définition récursive

Un arbre binaire est :

Soit vide

Soit un nœud qui possède un sous-arbre gauche et un sous-arbre droite eux mêmes arbres binaires

Représentation graphique



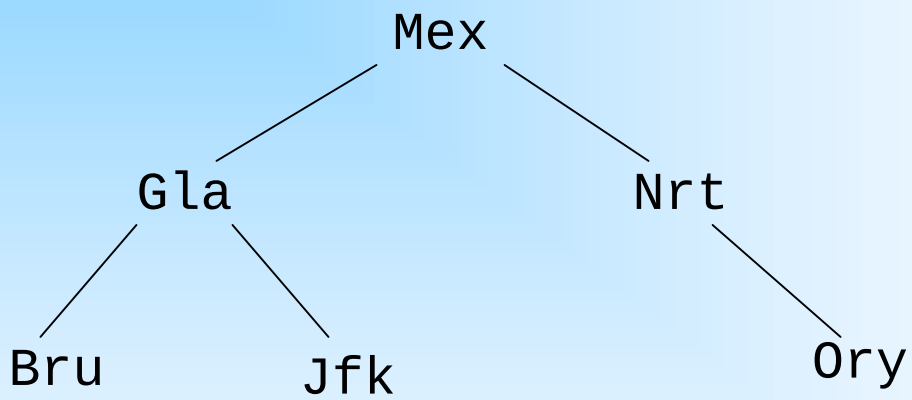
Exemple (1/8) : arbre binaire de recherche

L'ensemble des informations concernant les aéroports internationaux est classé dans un arbre binaire selon l'ordre lexicographique de leur sigle.

Ce choix (arbre plutôt que liste) améliore aussi bien la *recherche* que l'*ajout* d'un nouvel aéroport

La longueur d'un chemin de l'arbre est plus courte que la longueur de la séquence

Exemple (2/8) : un arbre



Propriété conservée :

`cle(fils_gauche)< cle < cle(fils_droite)`

Exemple (3/8) : typage

```
type Noeud;
type Sigle is new String(1..3);
type ArbreBinaire is access Noeud;
type Element is
  record
    leSigle:Sigle;
    info   :Donnee;-- on suppose défini le type Donnee
  end record;
type Noeud is
  record
    valeur:Element;
    filsGauche:ArbreBinaire;
    filsDroite:ArbreBinaire;
  end record;
```

Exemple (4/8) : recherche

Soient `a` l'arbre binaire, `ag` et `ad` respectivement ses fils gauche et droite.

Soit un couple (sigle,info) noté `<s,d>, Element` racine de `a` :

`a=<<s,d>,ag,ad>`

Exemple (5/8) : algorithme

Soit a= <<s,d>,ag,ad>

```
rechercher(s':Sigle,a:ArbreBinaire)=
      si vide(a) alors exception
      sinon
            si s=s' alors d
            sinon si s>s'
            alors rechercher(s',ag)
            sinon rechercher(s',ad)
            fin si;
      fin si;
fin si;
```

Exemple (6/8) :code de **rechercher**

```
function rechercher(s:Sigle;a:ArbreBinaire)
                                    return Donnee is
   begin
      if a=NULL
      then raise arbreVide;
      elsif a.valeur.leSigle=s
         then return a.valeur.info;
         elsif a.valeur.leSigle>s
            then return rechercher(s,a.filsGauche);
            else return rechercher(s,a.filsDroite);
      end if;
   end rechercher;
```

Exemple (7/8) : algorithme de **inserer**

```
inserer
(s':Sigle,d':Donnee,<<s,d>,ag,ad>:ArbreBinaire)=
      si vide(a) alors <<s',d'>,NULL,NULL>
      sinon
            si s/=s'
            alors si s>s'
                  alors <<s,d>,inserer(s',d',ag),ad>
                  sinon <<s,d>,ag,inserer(s',d',ad)>
            fin si;
      fin si;
fin si;
```

Exemple (8/8) : inserer

```
function inserer(e:Element;a:ArbreBinaire)
                                        return ArbreBinaire is
begin
      if a=NULL
      then return new Noeud'(e,NULL,NULL);
      elsif a.valeur.leSigle /= e.leSigle
         then if a.valeur.leSigle > e.leSigle
              then return
new Noeud'(a.valeur,inserer(e,a.filsGauche),a.filsDroite);
              else return
new Noeud'(a.valeur,a.filsGauche,inserer(e,a.filsDroite));
              end if;
         else raise redondant;
      end if;
end inserer;
```