Algorithmique
Arbres généraux : implémentations

Pour tous les types présentés ici, on suppose le type `t_element` (le type des contenus des nœuds) défini.

1 Des listes de nœuds

Les nœuds de l'arbre sont dans une liste. Pour chaque nœud on accède à ses fils grace à une liste chaînée. Chaque nœud est donc un enregistrement contenant :
- l'étiquette du nœud
- le pointeur sur la liste des fils (NUL si vide)

1.1 Liste des nœuds statique : implémentation statique-dynamique

- La liste des nœuds est un tableau (vecteur) ;
- L'accès à un nœud se fait par sa position dans ce tableau ⇒ la liste chaînée des fils contient des entiers (les indices des nœuds fils).

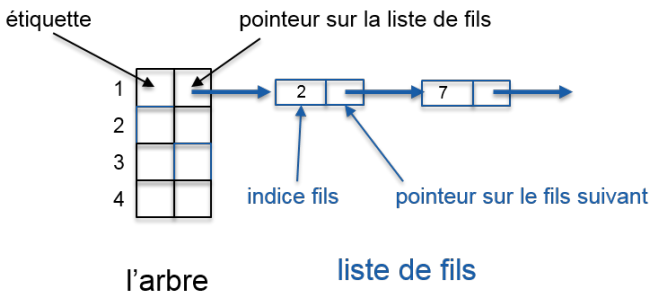
constantes
```
Nbmaxnoeud = 16
```

types
```
t_pfils = ↑ t_fils

t_fils = enregistrement
    entier      noeudfils
    t_pfils     filssuiv
fin enregistrement t_fils

t_noeud = enregistrement
    t_element   elt
    t_pfils     premierfils
fin enregistrement t_noeud

t_arbre = Nbmaxnoeud  t_noeud
```



Cette implémentation n'est pas très efficace en terme d'occupation mémoire : elle ne dépend pas de la taille réelle de l'arbre, mais de la taille du tableau (la constante).

1.2 Liste des nœuds dynamique : implémentation dynamique

Pour éviter d'avoir le problème d'occupation mémoire, on remplace le tableau statique par une liste dynamique.

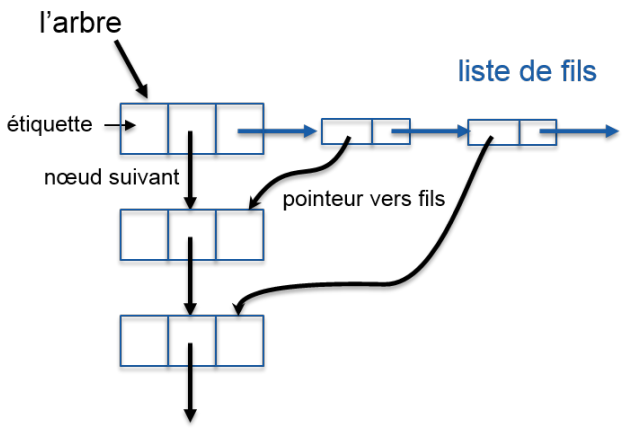
- La liste des nœuds est une liste chaînée ;
- L'accès à un nœud se fait par le pointeur sur l'enregistrement le représentant ⇒ la liste chaînée des fils contient des pointeurs (à la place des indices).

types
```
t_pnoeud = ↑ t_noeud   /* l'arbre */

t_pfils = ↑ t_fils

t_fils = enregistrement
    t_pnoeud    noeudfils
    t_pfils     filssuiv
fin enregistrement t_fils

t_noeud = enregistrement
    t_element   elt
    t_pnoeud    noeudsuiv
    t_pfils     premierfils
fin enregistrement t_noeud
```



2 Un pointeur : calque de la structure récursive

Un arbre général est défini par un nœud racine et une liste d'arbres. Ici, chaque nœud est représenté par un enregistrement qui contient un élément (l'étiquette) et une liste de fils : une liste de pointeurs vers les nœuds fils. L'arbre est un pointeur sur le nœud racine.

Deux implémentations, selon que la liste des fils est statique ou dynamique.

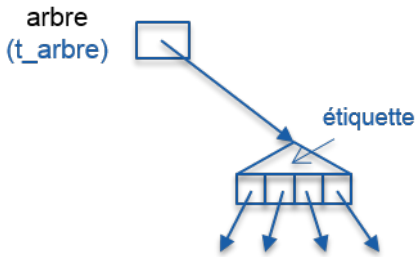
2.1 *N-uplets* de pointeurs

La liste des fils est une liste statique : un tableau. Si le nœud actuel a n fils, alors les n premières cases du tableau contiennent les pointeurs vers ses fils. Le reste du tableau contient NUL.

```
constantes
    Nbmaxfils = 4
types
    t_arbre = ↑ t_noeud

    t_tabfils = Nbmaxfils  t_arbre

    t_noeud = enregistrement
        t_element   elt
        t_tabfils   fils
    fin enregistrement t_noeud
```



À noter que dans certains cas, on ajoute en chaque nœud un entier : le nombre de fils. Dans ce cas, inutile de compléter le tableau par des valeurs NUL à la suite des fils.

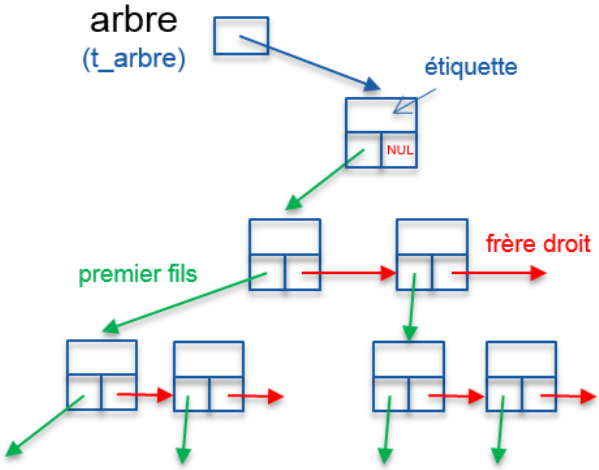
Coté occupation mémoire, on a toujours le problème des implémentations statiques : À moins que les nœuds aient globalement le même nombre de fils, c'est peu intéressant parce que très coûteux en mémoire.

2.2 Arbre binaire : *premier fils - frère droit*

La liste des fils est une liste dynamique : une liste chaînée. Du coup, chaque nœud contient deux pointeurs :
— un pointeur vers la liste de ses fils : vers le premier fils (NUL pour les feuilles) ;
— si ce n'est pas la racine, il fait partie d'une liste de fils, donc un pointeur vers le fils suivant : son frère droit.

```
types
    t_arbre = ↑ t_noeud

    t_noeud = enregistrement
        t_element   elt
        t_pnoeud    premierfils, freredroit
    fin enregistrement t_noeud
```



Si la "structure logique" de l'arbre reste celle d'un arbre général, sa "structure physique" est celle d'un arbre binaire Cette implémentation est quelquefois appelée la *bijection premier fils-frère droit.*

Ici, l'occupation mémoire est optimale : il y a un enregistrement pour un nœud.