13 - Parcours en profondeur des Arbres Binaires

I – Arbre Binaire

Un **arbre binaire** est :

- soit un **arbre vide**
- soit un **noeud qui mène à deux sous-arbres binaires** qu'on peut positionner G et D.

Interface :

```
nvNd(x:Elt) -> Noeud
contenu(noeud:Noeud) -> Elt                    Précondition : Noeud NON VIDE
nvAV() -> Arbre
nvAB(noeud:Noeud, g:Arbre, d:Arbre) -> Arbre
estArbreVide(arbre:Arbre) → bool               Renvoie True si l'Arbre est vide.
racine(arbre:Arbre) -> Noeud                   Précondition : Arbre NON VIDE
gauche(arbre:Arbre) -> Arbre                   Précondition : Arbre NON VIDE
droite(arbre:Arbre) -> Arbre                   Précondition : Arbre NON VIDE
```

II – Parcours Prefixe **R** G D ou **R** D G

Préfixe veut dire qu'on explore d'abord le Noeud actuel avant d'aller voir à gauche et à droite.

1. **R** : on explore d'abord le noeud actuel (la racine de l'arbre en cours donc),
2. **G** : ensuite on explore le sous-arbre gauche
3. **D** : puis on explore le sous-arbre droit

Description des ENTREES / SORTIE

ENTREE : Un Arbre Binaire

Précondition : Arbre non VIDE

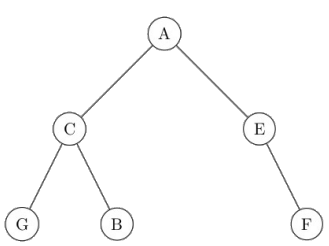
SORTIE : Rien (sur l'exemple)

01° Appliquer l'algorithme à l'Arbre Binaire 1.
02° Appliquer cet algorithme à l'AB 2.
03° Appliquer cet algorithme à l'AB 3.
04° Appliquer cet algorithme à l'AB 4.
05° Fournir un algorithme équivalent mais dont la PRECONDITION est simplement que le paramètre fourni est un ARBRE BINAIRE (potentiellement, il peut donc être un Arbre Vide).

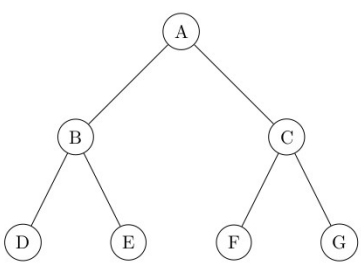
Description de l'algorithme récursif `parcourir(arbre: Arbre)`

```
visiter(racine(arbre))

SI NON estVide(gauche(arbre))
    parcourir(gauche(arbre))
Fin SI

SI NON estVide(droite(arbre))
    parcourir(droite(arbre))
Fin SI

Renvoyer VIDE
```

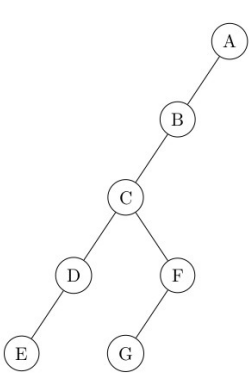
Arbre 1 :



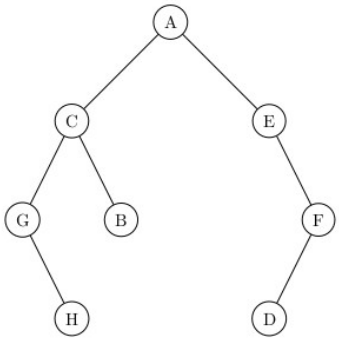
Arbre 2 :



Arbre 3 :



Arbre 4 :



III – Parcours Suffixe (ou postfixe) G D **R** **ou** D G **R**

On explore d'abord le sous-arbre gauche puis droite et on finit par le noeud actuel.

1. **G** : on explore d'abord le sous-arbre gauche
2. **D** : puis on explore le sous-arbre droit
3. **R** : enfin on explore le noeud actuel (la racine de l'arbre en cours donc)

Description des ENTREES / SORTIE

ENTREE : Un Arbre Binaire

Précondition : Arbre non VIDE

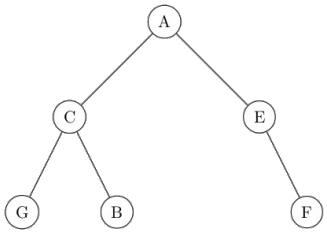
SORTIE : Rien (sur l'exemple)

06° Appliquer cet algorithme à l'AB 1.
07° Appliquer cet algorithme à l'AB 2.
08° Appliquer cet algorithme à l'AB 3.
09° Appliquer cet algorithme à l'AB 4.
10° Fournir un algorithme équivalent mais dont la PRECONDITION est simplement que le paramètre fourni est un ARBRE BINAIRE.

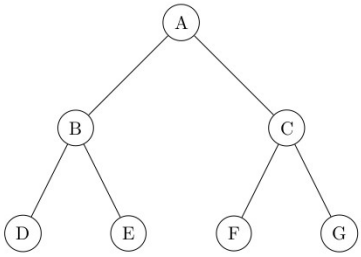
Description de l'algorithme récursif `parcourir(arbre: Arbre)`

```
SI NON estVide(gauche(arbre))
    parcourir(gauche(arbre))
Fin SI

SI NON estVide(droite(arbre))
    parcourir(droite(arbre))
Fin SI

visiter(racine(arbre))
Renvoyer VIDE
```

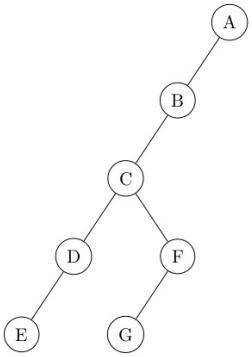
Arbre 1 :



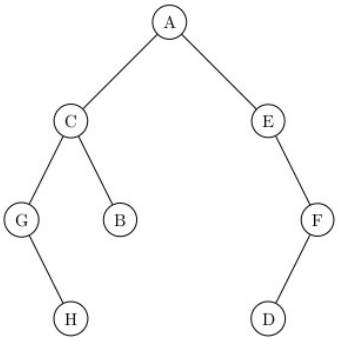
Arbre 2 :



Arbre 3 :



Arbre 4 :



IV – Parcours Infixe G **R** D **ou** D **R** G

On explore d'abord le sous-arbre gauche puis le noeud actuel et on finti par le sous-arbre droit.

1. G : on explore d'abord le sous-arbre gauche
2. R : puis on explore le noeud actuel (la racine de l'arbre en cours donc),
3. D : enfin on explore le sous-arbre droit

Description des ENTREES / SORTIE

ENTREE : Un Arbre Binaire

Précondition : Arbre non VIDE

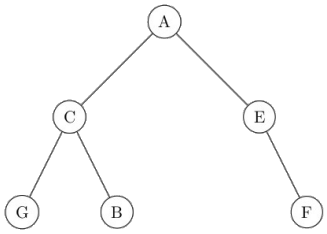
SORTIE : Rien (sur l'exemple)

11° Appliquer cet algorithme à l'AB 1.
12° Appliquer cet algorithme à l'AB 2.
13° Appliquer cet algorithme à l'AB 3.
14° Appliquer cet algorithme à l'AB 4.
15° Fournir un algorithme équivalent mais dont la PRECONDITION est simplement que le paramètre fourni est un ARBRE BINAIRE.

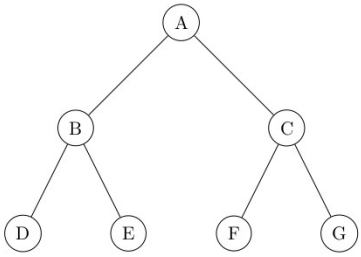
Description de l'algorithme récursif `parcourir(arbre: Arbre)`

```
SI NON estVide(gauche(arbre))
    parcourir(gauche(arbre))
Fin SI

visiter(racine(arbre))

SI NON estVide(droite(arbre))
    parcourir(droite(arbre))
Fin SI

Renvoyer VIDE
```

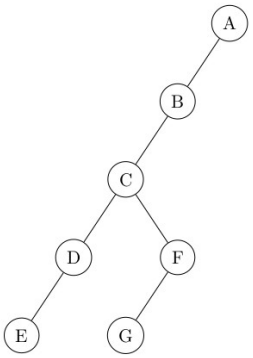
Arbre 1 :



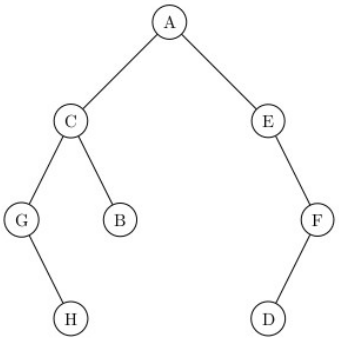
Arbre 2 :



Arbre 3 :

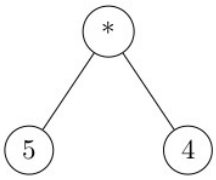


Arbre 4 :



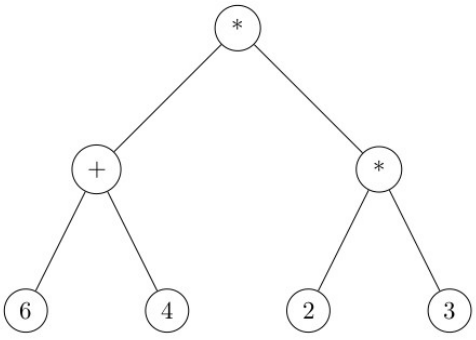
V – Parcours et formules

On peut représenter les formules arithmétiques ou logiques sous forme d'Arbre.
On peut considérer ici que des opérateurs binaires puisqu'on explore des Arbres Binaires.



Il s'agit de la formule 5 * 4.

```
INFIXE :      5 * 4
PREFIXE :   * 5 4          → notation polonaise
SUFFIXE :     5 4 *        → notation polonaise inversée
```



Il s'agit de la formule (6+4) * (2*3)

```
INFIXE :     6 + 4 * 2 * 3

PREFIXE :   * + 6 4 * 2 3

SUFFIXE :    6 4 + 2 3 * *
```

16° Retrouver la formule dont le parcours préfixe donne ceci : **+ 4 * 2 10**

17° Retrouver la formule dont le parcours préfixe donne ceci : **/ * 2 10 - 8 6**

18° Fournir les trois parcours de la formule logique dont l'Arbre est fourni ci-dessous.