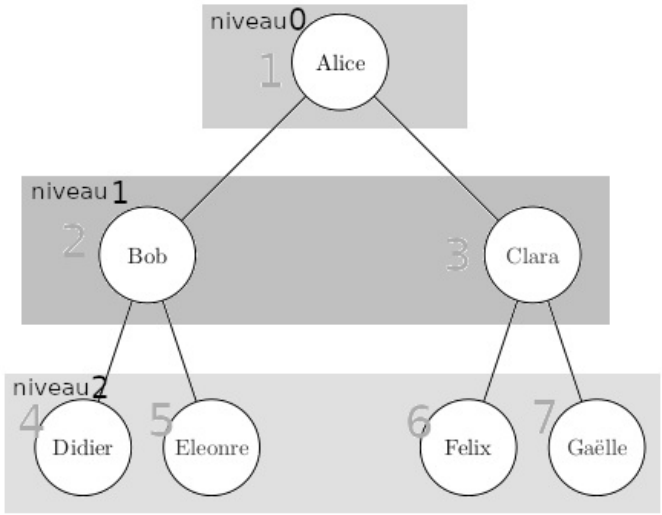
RÉSUMÉ 16 - PARCOURS EN LARGEUR

Lien vers l'activité : Parcours en largeur
www.infoforall.fr - Dernière modif. : 30 01 2021

1 - PRINCIPE

Le **parcours en largeur** consiste à lire les étiquettes des noeuds de l'Arbre Binaire en les listant étage par étage, souvent de la gauche vers la droite.



L'ordre de lecture des noeuds donne donc Alice - Bob - Clara - Didier - Eleonore...

2 - ALGORITHME

L'algorithme de lecture nécessite d'avoir :
- un Arbre à lire
- une File permettant de sortir naturellement le premier arrivé dans la File

Objectif : explorer les noeuds de l'arbre niveau par niveau.
ENTREE : **arbre**, l'arbre qu'on veut explorer
SORTIE : Vide (sur cet exemple), ou une réponse correspondant à la séquence des noeuds rencontrés.
La fonction **explorer** est à définir. Ici, on affichera juste la **clé** ou l'**étiquette** du noeud.

```
→ arbre doit contenir l'Arbre Binaire à explorer
SI NON estArbreVide(arbre)
    f ← nouvelleFile()
    enfiler(arbre, f)
    TANT QUE NON estFileVide(f)

        → étape 1 : on extrait le prochain AB et on affiche sa racine
        arbre_en_cours ← defiler(f)
        explorer(racine(arbre_en_cours))

        → étape 2 : on voit si on rajoute le sous-arbre gauche dans la File
        g ← gauche(arbre_en_cours)
        SI NON estArbreVide(g)
            enfiler(g, f)
        Fin Si

        → étape 3 : on voit si on rajoute le sous-arbre droite dans la File
        d ← droite(arbre_en_cours)
        SI NON estArbreVide(d)
            enfiler(d, f)
        Fin Si
    Fin Tant Que
Fin Si
Renvoyer VIDE (∅)
```

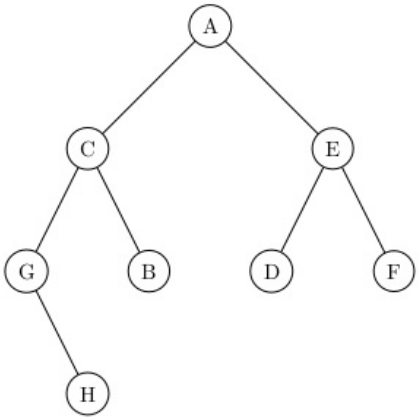
3 - IMPLÉMENTATION EN VERSION ITÉRATIVE ET RÉCURSIVE

```
 1  def parcours_largeur(arbre:Arbre) -> None :
 2      '''Exploration (et affichage) en largeur de l'arbre (version iterative)'''
 3      if not estArbreVide(arbre):
 4          f = nvFile()
 5          enfiler(arbre, f)
 6          while not estFileVide(f):
 7              arbre_en_cours =  defiler(f)
 8              print(cle(racine(arbre_en_cours)))
 9              g = gauche(arbre_en_cours)
10              if not estArbreVide(g):
11                  enfiler(g, f)
12              d = droite(arbre_en_cours)
13              if not estArbreVide(d):
14                  enfiler(d, f)
```

On peut transformer le TANT QUE en appel récursif, plutôt que d'utiliser la version itérative.

```
 1  def parcours_largeur_rec(arbre:Arbre) -> None:
 2      '''Exploration (et affichage) en largeur de l'arbre (version récursive)'''
 3      if not estArbreVide(arbre):
 4          f = nvFile()
 5          enfiler(arbre, f)
 6          exploration_suivante(f)
 7
 8  def exploration_suivante(f:File) -> None:
 9      '''Affiche la prochaine racine et stocke dans f les sous-arbres détectés''':
10      if estFileVide(f):  # Condition d'arrêt
11          return None     # Cas de base
12      else:               # Cas récursif
13          arbre_en_cours =  defiler(f)
14          print(cle(racine(arbre_en_cours)))
15          g = gauche(arbre_en_cours)
16          if not estArbreVide(g):
17              enfiler(g, f)
18          d = droite(arbre_en_cours)
19          if not estArbreVide(d):
20              enfiler(d, f)
21          exploration_suivante(f)
```

4 - EXERCICE



Déterminer l'ordre des noeuds affichés en montrant l'état de la File au fur et à mesure de l'avancement de la progression de l'algorithme.