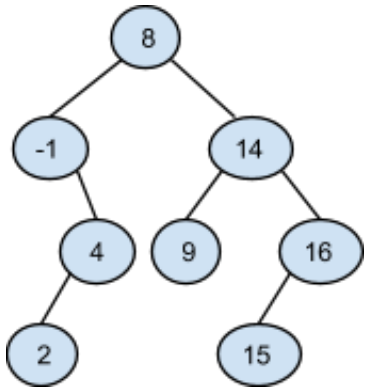
TP 8 SDA

Parcours en largeur des arbres binaires. Files.

Le parcours en largeur d'un arbre consiste à afficher les étiquettes des noeuds par niveau : d'abord le niveau 0 (la racine) puis le niveau 1 (tous les fils de la racine), etc. Dans l'exemple suivant cela consiste à afficher les noeuds dans l'ordre suivant :
8 -1 14 4 9 16 2 15



Pour réaliser un tel parcours, on utilise une structure de données nommée *file* (*queue* en anglais). L'objectif de ce type est de représenter les files d'attente. Concrètement il s'agit d'une liste dans laquelle les insertions ne se font qu'en fin de liste et les suppression en début de liste.

File: 1 -> 2 -> 3
enfile(4): 1 -> 2 -> 3 -> 4
enfile(5): 1 -> 2 -> 3 -> 4 -> 5
x=defile(); 2 -> 3 -> 4 -> 5
x=defile(); 3 -> 4 -> 5
...

Implémenter une file de pointeur (void*) en partant du code des listes doublement chaînée, que vous simplifierez. *push_back* devient *enfile*, *pop_front+front_val* devient *defile*.

Vous définirez les structure FileNoeud et File et vous implémenterez les fonctions suivantes
FileNoeud* nouveau_noeud(void* v);
File* file_vide();
Bool est_file_vide(File* fi);
void* defile(File* fi);
void enfile(File* l, void* v);

L’algo de parcours en largeur est le suivant:

```
parcours_largeur(arbre a) :
  file f
  enfiler(noeud racine de a)
  tant que f non vide
      n = defile(f)
      afficher l’étiquette de n
      ajouter tous les fils de n dans f
```

Implementer et tester le parcours en largeur en utilisant les files pour stocker des pointeurs de structures Arbre.