1 But

Il y deux projets possibles qui utilisent la structure de donnée décrite ici.

- 1. Écrire une bibliothèque pour gérer un dictionnaire de chaîne de caractères.
- 2. Écrire un programme de gestion de lexique.

2 Modèle

En C une **chaine de caractères** un tableau de **char** qui se termine par le caractère '\0' (le caractère nul). En C un mot est représenté (par exemple des) par une zone mémoire qui contient une chaine de caractère. (par exemple \boxed{d} \boxed{e} \boxed{s} $\boxed{\setminus 0}$).

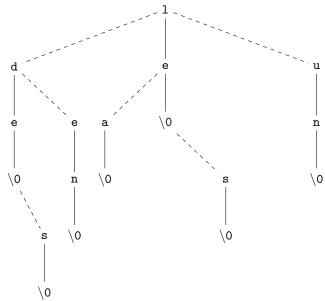
On peut représenter un lexique (un ensemble de chaines de caractères) par un arbre ternaire. Chaque noeud de l'arbre contient au moins 4 champs :

- 1. Un caractère : char
- 2. Un frère droit (pointeur sur un sous-arbre).
- 3. Un fils (pointeur sur un sous-arbre).
- 4. Un frère gauche (pointeur sur un sous-arbre).

Dans cet arbre, chaque chemin partant de la racine et arrivant à un noeud contenant \0 représente une chaine de caractère du lexique : Lorsque le chemin passe par un arc fils alors on ajoute une lettre dans la chaine de caractère : celle du noeud origine de l'arc. Passer par les arcs frère gauche ou frère droit n'ajoute pas de lettre mais permet de ranger dans l'arbre des chaines de caractères ayant un même préfixe puis continuant avec une lettre strictement plus grande (pour frère droit) ou strictement plus petite (pour frère gauche) (selon l'ordre classique sur les octets).

3 Exemple

Par exemple voici l'arbre qu'on obtient lorsqu'on ajoute dans cet ordre les mots : le, la, les, un, de, des, en.



4 Améliorations possibles

1. Adapter les techniques vues pour les arbres AVL sur les arbres ternaires précédent afin d'améliorer la complexité des fonctions d'ajout et de recherche dans l'ensemble.

- 2. Modifier la structure des noeuds afin d'éviter les feuilles ayant \0 comme étiquette
- 3. ...

5 Projet 1

Fabriquer une bibliothèque statique C (ou/et dynamique) de fonctions (deux fichiers: maBiblioteque.a ou maBibliotheque.so et monEntete.h). Les fonctions doivent implanter maps (ou dictionnaire) (voir description d'une map sur le site) dont les clés sont des chaines de caractères de la structure d'arbre ternaire précédemment décrite.

Exemple d'utilisation (pas forcément à suivre).

```
Dictionnaire *dico=initDico();
Lien l=ajouter(dico,"ballon");
associer(l,"Objet en forme de sphère, nécessaire pour certains jeux de plein air.");
Lien l=ajouter(dico,"chaise");
associer(l,"Une chaise est un objet faisant partie des commodités de la conversation.");
l=chercher(dico,"ballon");
if(estLie(l))
    printf("\%s",(char*)recuperer(l)); // imprime <<Objet en forme de ... air.>>
detruireDico(a); // desalloue le dictionnaire
```

6 Projet 2

Dans la suite on considère qu'un mot d'un fichier est une chaine de caractère obtenue avec une fonction *scanf("%s",..) (ou plus sécurisé: *scanf("%50s"..))

En utilisant la structure d'arbre décrite précédemment pour représenter des ensembles de mots, construire une commande/programme lexique qui prendra les options suivantes.

- lexique -s nom sauvegarde l'arbre construit à partir des mots du fichier nom dans un fichier nom.DICO le fichier de sauvegarde utilise un parcours préfixe de l'arbre ternaire. Les fils et frères vides seront codés par \n (saut de ligne) car \n ne peut pas apparaître dans une chaine de caractère obtenue avec scanf.
- lexique -l nom.DICO affiche les mots du fichier nom dans l'ordre alphabétique sur la sortie standard. Par défaut utilise le fichier nom.DICO s'il existe pour fabriquer l'arbre sans s'occuper de nom.
- lexique -r nom monMot affiche une phrase indiquant si le mot monMot est présent dans le fichier nom. Utilise le fichier nom.DICO par défaut.

Exemple :

```
\$Lexique -s LettreDeMonMoulin.txt # création du fichier LettreDeMonMoulin.txt.DICO
\$Lexique -l LettreDeMonMoulin.txt.DICO
. # affiche des mots
.
.
\$Lexique -r LettreDeMonMoulin.txt Seguin
Le mot <<Seguin>> est parmi les mots de <<LettreDeMonMoulin.txt>>
```