TP N°6
Un lexique français-Anglais

Objectifs :

- ✓ Utilisation des structures et des listes chainées
- ✓ Manipulation d'arbre
- ✓ Manipulation de chaines de caractères
- ✓ Allocation de mémoire statique et dynamique

Préambule

Après avoir ouvert virtualBox et vous être logué sous l'environnement CentOS5, on vous demande de créer un répertoire **`tp6`** dans votre répertoire de travail **`algo.`** Vous vous placerez ensuite dans le répertoire **`tp6`**. Vous allez créer un fichier nommé tp5.c qui contiendra le code source de votre TP (vous resterez dans ce fichier pour tout le TP).

Un lexique français-anglais est un ensemble de couples de mots dont le 2ème est une traduction du 1er (traduction supposée unique).

La représentation logique habituelle est une table. Mais comme la probabilité de rechercher un mot n'est pas uniforme, on peut préférer une représentation par **un arbre binaire**. Ceci permet d'optimiser le temps moyen d'accès aux mots.

Les mots sont placés aux nœuds de telle façon que, pour tout nœud n de valeur v, on trouve à gauche de n tous les mots plus petits que v et à droite de n tous les mots plus grands que v[1].
Remarque : Dans cet exercice, la comparaison de deux mots porte sur l'ordre alphabétique.

On trouvera en Figure 1 un exemple de lexique sous forme d'arbre binaire.

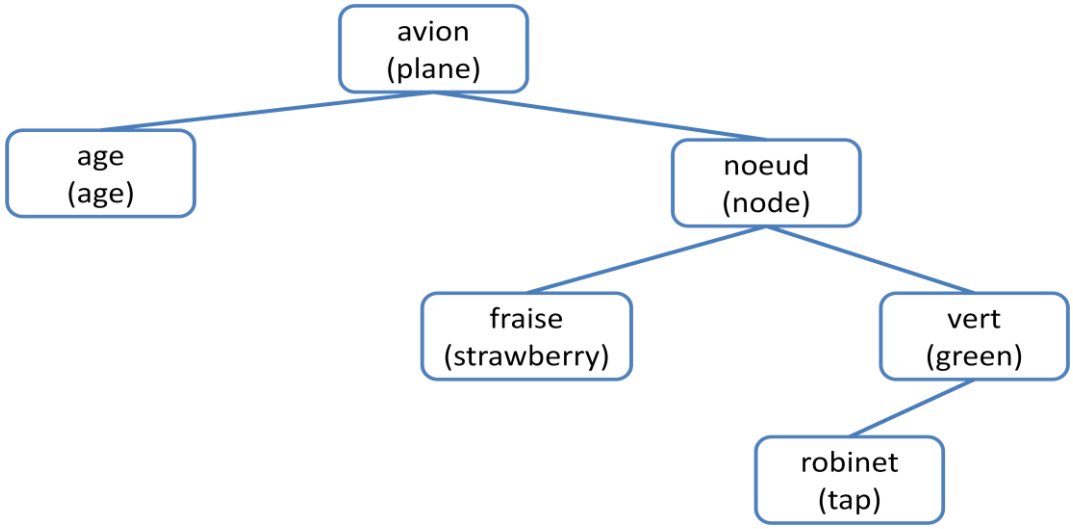

Figure 1 : un exemple de lexique

[1] Dans cet exercice, la comparaison porte sur l'ordre alphabétique

1. Structure de donnée

Proposer en langage C une structure de donnée que vous appellerez **`struct noeud`** permettant de représenter un nœud de l'arbre. Vous insérerez cette structure en en-tête de votre fichier `tp6.c`

2. Traduction

Ecrire en langage C la fonction **`void traduction(char* mot, struct noeud* lexique)`** qui affiche la traduction de la chaine de caractères **mot** si ce mot existe dans le lexique ou « `pas de traduction` » sinon.

On pourra utiliser la fonction C : **`int strcmp(char* mot1, char* mot2)`** disponible dans « **`string.h`** » qui renvoie un nombre négatif si mot1 est avant mot2 dans l'ordre alphabétique, un nombre positif dans le cas contraire et enfin 0 si les 2 mots sont identiques.

Dans votre fichier tp6.c, vous testerez la fonction **`traduction`** dans une fonction **`main`** en utilisant une liste chainée de structure **`struct noeud`** que vous initialiserez avec un seul mot.

3. Insertion d'un mot

Ecrire en langage C, la fonction **`inserer`** qui permet d'insérer un mot et sa traduction dans le lexique, son prototype devra être :

```
struct noeud* inserer(char* mot,char* traduction, struct noeud* lexique)
```

où les paramètres **`mot`** et **`traduction`** correspondent respectivement au mot et à la traduction à ajouter au lexique tandis que le paramètre **`lexique`** représente l'adresse du premier élément du lexique. La fonction retourne un nouveau pointeur sur structure **`struct noeud`** représentant la nouvelle adresse du premier élément du lexique (si tant est qu'elle a changé !)

L'insertion se fera à un nœud qui n'a pas encore de fils gauche en cas d'adjonction à gauche ou pas de fils droit en cas d'adjonction à droite.

Vous testerez votre fonction en tentant d'insérer un mot et sa traduction.

4. Programme complet

Ecrire en langage C, un programme principal qui construit le lexique décrit dans la Figure 1 et qui demande à l'utilisateur de saisir un mot à traduire. Le programme se termine quand l'utilisateur tape « fin ».

5. Annexes : affectation et copie de chaine de caractères

```
char* chaine = "Bonjour";
char* copie1 ; // simple pointeur
char* copie2 = malloc(sizeof(char)*(strlen(chaine)+1)); //alloc dynamique
char copie3[10] ; // alloc statique

copie1 = chaine ; // copie du pointeur, l'espace mémoire reste le même

strcpy(copie2,chaine); // recopie de chaine dans un nouvel espace mémoire
                          pointé par copie2

strcpy(copie3,chaine); // recopie de chaine dans un nouvel espace mémoire
                          pointé par copie3
```