Ecole Polytechnique Universitaire
de Lille

Programmation Avancée

Rapport Projet Correcteur Orthographique

Corto Callerisa Sébastien Dardenne

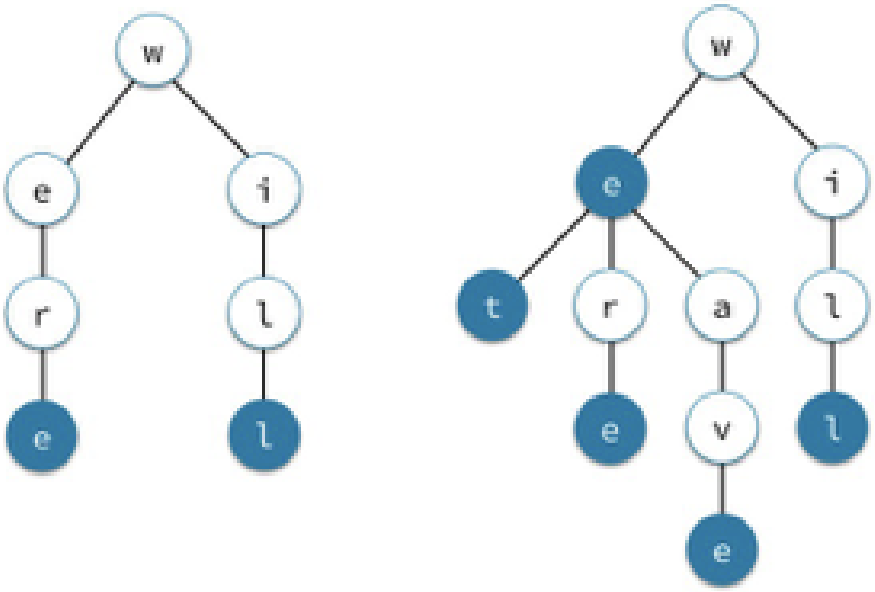
May 5, 2019

1 Introduction

Ce projet a pour but d'appliquer les connaissances apprises lors du semestre 6 en programmation avancée. Il s'agit de réaliser un correcteur orthographique en utilisant un dictionnaire que l'on a préalablement chargé. De plus ce programme doit pouvoir extraire les mots d'un texte ou d'une liste afin de les vérifier.

2 Principe

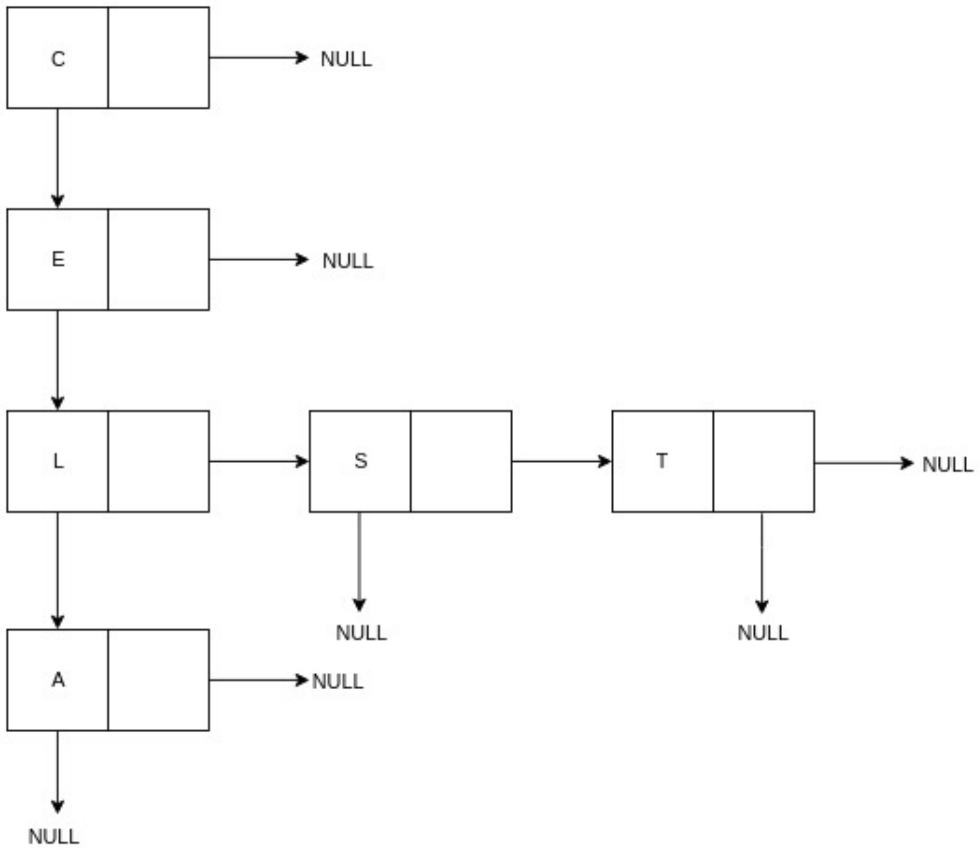
Afin de minimiser l'espace mémoire nécessaire au stockage du dictionnaire tout en fournissant un temps de recherche bas, la structure de données que vous utiliserez sera un arbre préfixe (encore appelé trie). Il s'agit d'une structure arborescente pour laquelle des mots ayant des préfixes communs sont factorisés: chaque noeud de l'arbre est une lettre qui peut être terminale (i.e. dernière lettre d'un mot) ou pas.
Considérons les mots: were et will. L'arbre (ou trie) correspondant est affiché ci-dessous à gauche. Si on ajoute les mots we, wet et weave l'arbre est affiché à droite; les noeuds colorés représentant des lettres terminales.



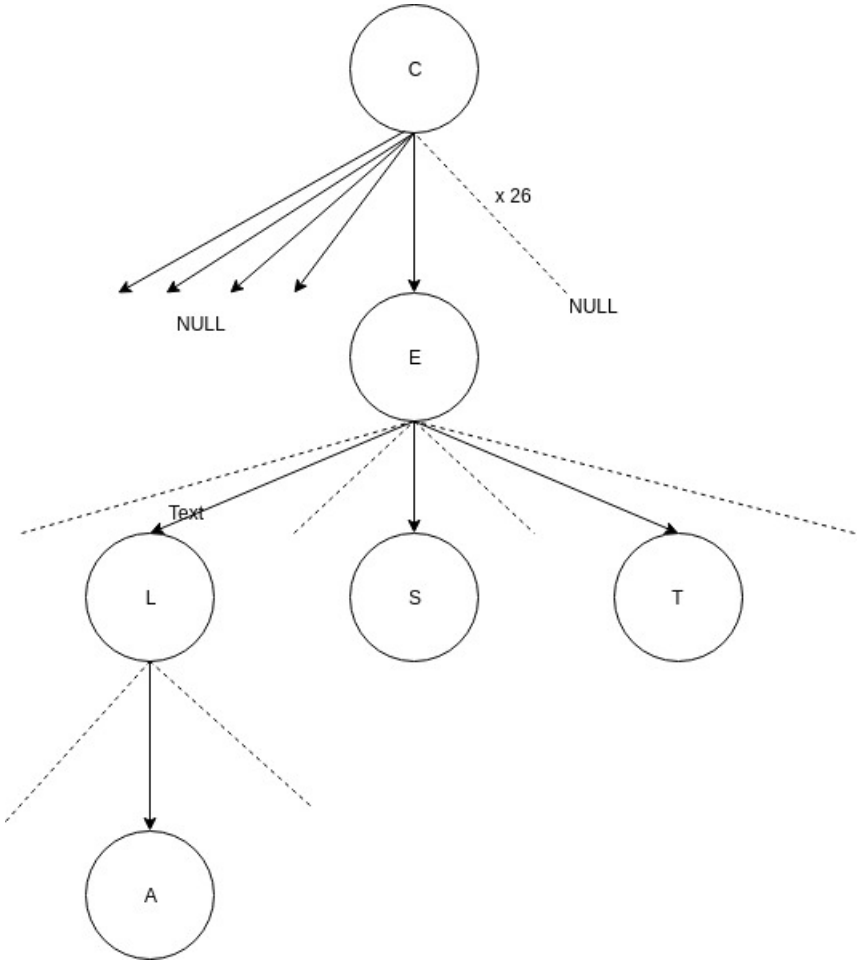
3 Structure de donnée

Votre structure doit permettre de stocker et de manipuler un dictionnaire sous forme d'arbre préfixe / trie

Pour la structure, nous avons eu deux idées principales, la première étant une structure sous forme de listes chainées néanmoins cette solution n'est pas optimale pour la recherche de caractère car la fonction parcours toute la liste dans le pire cas.
Exemple avec les mots : cela, ces et cet:



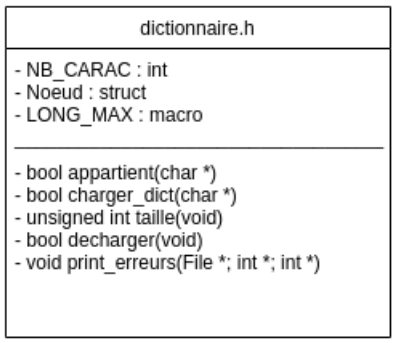
Nous avons alors changé notre idée principale et donc nous avons choisit de partir sur une seconde solution qui est une structure avec un noeud composé de 26 pointeurs (un par lettre) et un booléen indiquant si le noeud fini un mot. Cette structure étant plus optimisé car elle s'actualise en fonction des lettres à ajouter en allouant la mémoire nécessaire.
Exemple avec les mots : cela, ces et cet:

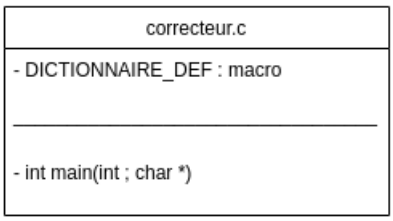


4 Structure du programme

Le programme permet de charger un dictionnaire à partir d'un fichier texte de données et analyse l'orthographe du fichier texte en indiquant les nombres de mots qui ne sont pas reconnus par le dictionnaire.

Notre programme ce compose de 2 script principaux qui sont le dictionnaire et le correcteur, le dictionnaire permet d'importer un dictionnaire depuis un fichier texte et de vérifier l'appartenance d'un mot au dictionnaire importer. Le correcteur permet de vérifier l'orthographe de l'ensemble des mots d'un texte. Deplus, nous avons un MAKEFILE qui permet de compiler automatiquement les scripts.





5 Respect du cahier des charges

Cahier des charges:

- Définir et implémenter une structure de données permettant de stocker et de manipuler un dictionnaire sous forme d'arbre préfixe / trie.
- Charger un dictionnaire à partir d'un fichier texte de données. Ce fichier texte pouvant être un texte court, un roman ou une liste de mots.
- Par exemple le fichier /etc/dictionaries-common/words est un dictionnaire de langue anglaise.
- Analyser l'orthographe d'une phrase ou d'un texte en indiquant les nombres de mots qui ne sont pas reconnus par le dictionnaire.

1. Structure de trie pour stocker et manipuler le dictionnaire :

```
1 /*
2    La structure principale du programme est un trie ou chaque
     Noeud possede jusqu'a 26 enfants ainsi qu'un booleen
     indiquant si le noeud termine un mot valide.
3  */
4 typedef struct Noeud {
5     bool mot_fini;
6     struct Noeud *enfants[NB_CARAC];
7 } Noeud;
```

2. Charger un dictionnaire :

Notre algorithme pour charger un dictionnaire est le suivant : Après avoir chargé l'ensemble du fichier dictionnaire dans un buffer on rempli le trie caractère par caractère afin d'insérer les mots. L'allocation de la mémoire permettant de charger le fichier est faite en une fois car l'on connait la taille en octets du fichier par l'appel à ftell() après avoir deplacé le pointeur de fichier à la fin. Si un prefixe du mot courant a déja été importé on parcourt le chemin lorsque qu'il y a une déviation on initialises les noeuds correspondants. L'apostrophe est gérée individuellement car son indice dans le tableau est un choix arbitraire. Chaque fois que l'on recontre le caractère ", on l'absorbe et on boucle sur l'ajout du prochain mot.
La fonction importer_dict() est contenue dans le fichier dictionnaire.c. Elle est commentée en détail.

3. Détection d'erreurs :
La détection des erreurs dans un texte est traitée dans le fichier correcteur.c qui fait des appels extensif à la fonction appartient() du fichier dictionnaire.c L'algorithme est similaire à celui pour l'insertion : On suit le parcours de l'arbre correspondant au lettres du mot à verifier. Si chaque lettre correspond à un fils on renvoie la valeur du booléen mot_fini du dernier noeud, et si le chemin entier n'existe pas on renvoie faux. Correcteur.c appel cette fonction pour chaque mot du dictionnaire traite les mots malorthographié en accord.

6 Limites du projet et tests

- Notre programme ne gère pas les mots avec accent et/ou caractère spéciaux. C'est pourquoi nous utilisons la langue anglaise pour les dictionnaires et les textes afin d'avoir que des lettres de l'alphabet et des apostrophes.

Le programme compile sans erreurs et reporte pas de fuites mémoire en utilisant l'utilitaire valgrind.