MINI-PROJET:

REALISATION D'UN CORRECTEUR DE TEXTE

RAOUTO Emilie Joyce

TUTEURS:

-DEQUIDT Jeremie
-RUDAMETKIN Walter
-FORGET Julien
-ARANEGA Vincent

COMPTE-RENDU:

I/ INTRODUCTION:

l'Objectif du projet est de réaliser un programme qui permet de détecter les mots mal orthographiés dans un texte. De ce fait, nous utiliserons, un dictionnaire construit à partir d'un fichier texte ou d'un fichier de référence.

I/PRINCIPAUX-ALGORITHMES:

**STRUCTURE CHOISIE:

J'ai choisi de faire une structure en utilisant une liste chainée avec plusieurs champs, car cette structure permet de ne pas fixer au préalable une zone mémoire, mais l'alloue lorqu'elle en a besoin. Ainsi, elle est constituée d'un noeud contenant une valeur de type char, un booléen complet qui précise si le mot existe, un entier nbfils qui s'incrémente à chaque noeud ajouté et enfin un pointeur vers les noeuds suivants.

En effet, le dictionnaire a pour champs un entier nbelem qui compte le nombre de noeuds de celui-ci et enfin un pointeur vers le premier noeud.

"struct noeud{ char valeur; struct noeud * liste_noeud; int nbfils; bool complet;};"

“struct dictionnaire{ struct noeud * tetes; int nbelem;};

**Fonction1: init_dictionnaire

Cette fontion prend en paramètre un pointeur de struct dictionnaire, elle permet d’initialiser le dictionnair en mettant tete à NULL et nbelem=0.

**Fonction2: cons_dictionnaire

Cette fonction prend en paramètre un pointeur de struct dictionnaire et le nom du fichier. Elle permet de construire un dictionnaire en ajoutant lettre par lettre un mot lu , qui est contenu dans le fichier en utilisant la fonction ‘ajout_mot’.

**Fonction3: ajout_mot

Cette fonction prend en paramètre un pointeur de struct dictionnaire et un mot de type char *. Elle consiste à ajouter un mot dans le dictionnaire. Pour commencer, je cherche le noeud qui contient la 1ere lettre du mot, en utilisant une boucle ‘while’ et en vérifiant que l’indice I n’est pas supérieur au nombre de mots qui existe (nbelem) puis en utilisant cette fois-ci une boucle ‘for’ je parcours le mot caractère par caractère en inserant chacun d’eux dans le bon noeud. Enfin, je modifie la valeur du booléen à true car elle est initialement à false pour préciser que nous sommes arrivés à la fin du mot.

**Fonction4: appartient_mot

La fonction “appartient_mot” est de type bool et prend en paramètres, un dictionnaire et mot. Elle permet de vérifier si le mot appartient au dictionnaire. Si c’est le cas , elle renvoie true sinon elle renvoie false. Pour commencer, elle vérifie si la première lettre du mot est contenu dans le tableau dynamique , si c’est le cas , elle continue la comparaison avec les fils du noeud sélectionné.

**Fonction5: correction

Cette fonction corrige un texte ou une phrase, selon les mots contenus dans le dictionnaire. En effet, elle prend en paramètre une phrase de type char * et un pointeur de dictionnaire. L’objectif est de parcourir la phrase et de relever tous les mots qui n’appartiennent pas au dictionnaire.
Ainsi, le mot peut être bien orthographié, si il n’est pas dans le dictionnaire le texte n’est pas totalement juste.

II/LIMITES:

Mon programme a plusieurs limites. Premièrement, je ne prends pas en compte les phrases ou les textes accentués, les majuscules et les caractères spéciaux tels que l'apostrophe, le point d'interrogation , le tiret. De plus, la fonction ajout_mot qui comporte plusieurs lignes, aurait pu être factoriser par une fonction récursive, cependant, j'avais le soucis de prendre en compte tous les cas possibles pour ne pas avoir un noeud mal rempli ou alors une erreur telle que 'segmentation fault'.

III/CONCLUSION:

Ce projet m'a permis de comprendre qu'il était possible de résoudre plusieurs problèmes en choisissant la bonne structure adaptée au problème. J'ai fait face à plusieurs obstacles, tout d'abord j'ai dû changé trois fois de structures d'où mon projet 'inachevé' et j'ai eu un milliard de 'segmentation fault' , et enfin j'ai pris beaucoup de temps pour écrire la fonction ajout_mot, qui est sans doute la fonction principale du projet.

En conclusion, en choisissant cette structure je dirai que j'ai fait un compromis entre mémoire et rapidité.