ANTEUR Neil
HUDELOT Matthias

PROJET : Arbre lexicographique

Modalité d'utilisation :

Pour que le programme se lance correctement, il faut impérativement fournir en ligne de commande un fichier à analyser (comme précisé dans le sujet du projet).

Toutes les erreurs de ligne de commande sont capturées et expliquées à l'utilisateur par le biais du terminal pour que celui-ci arrive à lancer correctement le programme.

Si l'utilisateur entre une option qui n'existe pas comme « ./Lexique **-Z** nom_du_fichier » l'option est ignorée.

Si l'utilisateur rentre une option valide, l'action correspondant à l'option est exécutées puis, le menu est affiché pour offrir la possibilité à l'utilisateur de continuer son utilisation ou quitter le programme

Dans la suite de ce rapport, **« nom_du_fichier »** représente le fichier que l'utilisateur désire traité.

Le programme offre les fonctionnalités suivantes :

1)Afficher le lexique de « nom_du_fichier » par ordre alphabétique

2)Sauvegarder l'ensemble du lexique par ordre alphabétique dans « nom_du_fichier**.LEX** »

3)Sauvegarder l'arbre lexicographique correspondant au lexique de « nom_du_fichier » au format spécifié dans l'énoncé dans « nom_du_fichier**.DIC** »

4)Rechercher un mot dans le lexique. Après avoir sélectionner l'action « recherche », l'utilisateur est invité par le programme à indiquer quel mot il souhaite rechercher .

5)Ouvrir un autre fichier et donc créer un autre lexique. De cette manière l'utilisateur peut gérer les lexiques de différents fichiers sans avoir à relancer le programme.

Toute ces fonctions sont présentées clairement à l'utilisateur dans un menu à l'aide de la fonction

int choix_menu(char* nom_fichier).

Cette fonction indique également à l'utilisateur le nom du ficher actuellement traité, dans un soucis de clarté pour l'utilisateur dans le cas où ce-dernier change de fichier a l'aide de l'action 5 « ouvrir un autre fichier »

Format de sauvegarde de l'arbre dans un fichier :

Les « \0 » sont sauvegardés par le caractère « ~ »

Nous avons rencontré un problème pour charger les sauvegardes de l'arbre.
Selon le format de sauvegarde indiqué dans l'énoncé, chaque caractère de chaque nœud est sauvegardé sous sa forme d'origine, y compris le caractère EOF : « \0 ».

Ceci nous rendait impossible le chargement de sauvegarde avec ce format car la lecture du fichier de sauvegarde s'interrompait dès la première rencontre avec un « \0 ».

Pour outrepasser ce problème nous avons pris la décision de sauvegarder les caractères « \0 » d'un nœud sous la forme « ~ » dans le fichier de sauvegarde.

Pour que la fonction de chargement construise un arbre valide, à chaque fois qu'elle rencontre le caractère « ~ » dans la sauvegarde, elle crée un nœud contenant « \0 » comme dans l'arbre d'origine.

Méthodes employées :

-Méthode d'insertion des mots dans l'arbre :

Cette fonction est utilisé récursivement de manière à insérer le mot lettre par lettre.
Int insertion (Arbre *a, char * mot) ;

Si le nœud (*a) n'existe pas. On en créer un nouveau avec le caractère à insérer,
Si c'est le caractère '\0', on a fini l'insertion
Sinon on rappel la fonction avec le caractère suivant et le nœud fils.

Si le nœud (*a) contient le caractère à insérer,
ET que ce caractère est '\0', on a fini l'insertion
sinon on insert la suite du mot dans le nœud fils.

Si le caractère a insérer est « plus petit » que (*a)->lettre.
on rappel la fonction avec le même caractère et le fils gauche

Si le caractère a insérer est « plus grand » que (*a)->lettre.
on rappel la fonction avec le même caractère et le fils droit.

-Recherche d'un dictionnaire pré-existant :

Dès le début du programme, les paramètres entrés par l'utilisateur sont étudiés.
Selon l'énoncé, le nom du fichier doit toujours être spécifié en dernier sur la ligne de commande

Selon le nombre de paramètres entré par l'utilisateur, le programme identifie celui correspondant au nom du fichier à ouvrir.
A partir du celui-ci, le programme crée le nom de la sauvegarde de l'arbre(avec l'extension .DIC) et du lexique potentiel (avec l'extension .LEX).

En premier lieu, le programme cherche dans le dossier courant si il existe déjà une sauvegarde de l'arbre correspondant à « nom_du_fichier.DIC ». Ainsi le lexique sera reconstruit a partir de la sauvegarde sans retraiter le fichier si le DIC ,existe.

Le programme tente d'ouvrir la sauvegarde de l'arbre :
Si l'ouverture est un succès, le programme reconstruit directement le lexique a partir de la sauvegarde.

Si l'ouverture échoue, c'est que le fichier n'existe pas
ALORS on ouvre le fichier nom_du_ficher et on créer le lexique en traitant tout le texte du fichier.