Projet : Langage C

3 ESGI

L'objectif de ce projet est de construire une application de gestion d'un dictionnaire au travers de nombreuses fonctionnalités de recherche. On considèrera ici qu'un dictionnaire est un simple ensemble de mots triés par ordre alphabétique, écrits dans un fichier les uns à la suite des autres séparés par un retour charriot.

Partie 1 : création et gestion du dictionnaire

Construire un ensemble de fonctions permettant de :

- créer un fichier dictionnaire,
- utiliser un dictionnaire existant,
- fabriquer un dictionnaire à partir d'un fichier texte,
- détruire un fichier dictionnaire,
- insérer un mot dans un dictionnaire,
- rechercher un mot dans un dictionnaire.

Ces fonctions devront être construites dans le fichier appelé gestbib.c et testées dans un programme appelé main1.c

Remarque : utiliser un gros fichier pour effectuer des simulations de performance.

Partie 2 : recherche avancée

Afin d'optimiser la recherche, il est maintenant souhaité de rajouter un outil permettant de proposer à l'utilisateur une liste de mots proches de celui qu'il utilise. La notion de proximité se traduira ici par le calcul du nombre de lettres différentes entre deux mots. Ce nombre sera appelé seuil.

Construire un ensemble de fonctions permettant de rechercher et d'afficher une liste de mots proches d'un mot fourni par l'utilisateur, le seuil devant être déterminé par celui-ci.

Ces fonctions devront être construites dans le fichier appelé gestrech.c et testées dans un programme appelé main2.c

Partie 3 : proposition de correction

Le rôle d'un dictionnaire étant souvent de vérifier l'orthographe d'un mot, il est maintenant proposé d'utiliser le dictionnaire pour vérifier tous les mots d'un fichier texte.

Construire une liste de fonctions permettant de lire un fichier texte, et effectuant le travail suivant :

- afficher la liste des mots du fichier n'existant pas dans le dictionnaire, ainsi que la ligne du fichier où ils sont utilisés

- afficher la liste des mots n'existant pas dans le dictionnaire et proposer une liste de mots proches (seuil deux lettres maximum) pour chacun d'entre eux

- construire automatiquement un fichier corrigé, en remplaçant les mots erronés par la première suggestion de la liste (seuil une lettre maximum)

Ces fonctions devront être construites dans le fichier appelé gestorth.c et testées dans un programme appelé main3.c

Partie 4 : recherche par expression régulière

Les expressions régulières forment une famille de notations compactes et puissantes pour décrire certaines chaînes de caractères. Ces notations sont utilisées par plusieurs éditeurs de textes et/ou langages de programmation, pour tout ce qui est recherche ou remplacement de motifs.

Construire maintenant un ensemble de fonctions permettant de déclencher une recherche de mots dans le dictionnaire au travers d'une expression régulière. Le jeu d'expressions devra être un sous-ensemble de l'ensemble normalisé de ces expressions comme fourni en annexe.

Ces fonctions devront être construites dans le fichier appelé gestexpr.c et testées dans un programme appelé main4.c

BIEN EVIDEMMENT L'UTILISATION DE BIBLIOTHÈQUES DE MANIPULATION D'EXPRESSIONS RÉGULIERES EST INTERDITE, TOUT DEVANT ETRE DÉVELOPPÉ COMPLÈTEMENT.

Chaque partie doit donc être traitée indépendamment, et proposer obligatoirement un menu de tests.

Règles de rendu de projet

L'évaluation du projet sera effectuée sur les points suivants :

- la réalisation de l'application en elle-même : les différentes parties doivent correspondre exactement à l'énoncé, et ne pas donner lieu à des plantages à l'exécution lors d'éventuelles mauvaises manipulations : l'utilisation de bibliothèques extérieures est interdite.

  Préférer rendre une partie incomplète plutôt qu'erronée.
- la construction d'un rapport de projet, fourni sous la forme d'un fichier pdf, contenant au minimum les informations suivantes :
  - une introduction, rappelant le sujet du projet, et la liste des étudiants y ayant participé,
  - une analyse rapide de l'application : liste des structures de données, description des fonctions principales, des choix d'implémentation et de tout détail technique important pour la compréhension du sujet,
  - un dossier d'installation de votre application,
  - un dossier d'utilisation précis (considérer que l'utilisateur final n'est pas un informaticien)
  - un bilan du projet, listant notamment les points non résolus, les difficultés techniques (ou humaines) rencontrées.

Prévoir de rendre l'intégralité des fichiers sources et compilés de l'application sur MYGES (il doit donc y avoir 4 fichiers exécutables), ainsi que les dictionnaires tests construits.

ANNEXE : MODELE DES EXPRESSIONS REGULIERES A PROGRAMMER

Expressions	Significations
Section \d	Section 0, Section 1, Section 2 ....
Section\s\d	Section suivi par un simple caractère d'espacement (space, tab, saut de ligne, etc.), suivi par un simple chiffre
Section\s\w	Section suivi par un simple caractère d'espacement (space, tab, saut de ligne, etc.), suivi par un caractère-mot (lettre ou chiffre XML 1.0)
a*x	x, ax, aax, aaax ... (* répétition de 0 à n fois)
a?x	ax, x (? répétition de 0 à 1 fois)
a+x	ax, aax, aaax ... (+ répétition de 1 à n fois)
(a|b)+x	ax, bx, aax, abx, bax, bbx, aaax, aabx, abax, abbx, baax, babx, bbax, bbbx, aaaax ... (| ou)
[abcde]x	ax, bx, cx, dx, ex (intervalle)
[a-e]x	ax, bx, cx, dx, ex
[-ae]x	-x, ax, ex
[ae-]x	ax, ex, -x
[^0-9]x	n'importe quel caractère non numérique suivi par le caractère x
.x	n'importe quel caractère suivi par le caractère x
.*abc.*	1x2abc, abc1x2, z3456abchooray ... (. vaut 1 caractère)
ab{2}x	abbx
ab{2,4}x	abbx, abbbx, abbbbx
ab{2,}x	abbx, abbbx, abbbbx ....
(ab){2}x	ababx
\n, \t, \r, \\, \{, \}, \), \), \?,\+, \^,	caractère correspondant Rq : \s=un séparateur
\p{L}	une lettre
\p{Lu}	une lettre majuscule
\p{LI}	une lettre minuscule
\p{N}	un nombre
\p{Nd} ou \d	un chiffre
\p{P}	un symbole de ponctuation
\p{Sc}	un signe monétaire
\d{n}	n chiffres
\Dx	n'importe quel caractère non chiffre suivi par le caractère x