Deuxième exercice : - Motus

Pour découvrir le jeu: https://motus.absolu-puzzle.com/

Description :

Dans le cadre de ce deuxième exercice, nous utiliserons un dictionnaire de mots sous la forme d'un fichier texte utilisant uniquement des caractères de la table d'encodage ASCII (en majuscules, pas de caractères accentués par exemple).

Les mots ne doivent contenir que 6 caractères alphabétiques, pour être plus pratique les minuscules seront converties en majuscules lors du chargement du fichier dictionnaire.

Comme dans l'exercice précédent, le fichier dictionnaire contient un mot par ligne, une ligne commençant par le caractère '#' désigne un commentaire et doit être ignorée lors du chargement des mots. Ce fichier vous est fourni et contient 17991 mots valides ordonnés.

Le joueur dispose d'un indice : le premier caractère du mot à trouver.

Le joueur doit deviner un mot en proposant des mots valides (présents dans le dictionnaire), le jeu lui indique pour chaque caractère s'il est bien placé, mal placé ou non utilisé dans le mot à deviner. Le joueur dispose d'un nombre de tentatives limité.

Travail attendu :

0- prendre en main le code fourni (fichier .h et de test)

1 – Répondez aux questions suivantes en prenant les hypothèses suivantes :

- On considère que le mot proposé est présent (même si on cherche à le vérifier) dans le dictionnaire, et qu'il suit ***une loi uniforme***, c'est à dire que ***chaque mot du dictionnaire à la même probabilité d'être proposé***.
- Dans un premier temps le dictionnaire ***n'est pas ordonné.***
- On code la fonction chercheMot de manière simple, c'est à dire que l'on ***parcourt tout le dictionnaire*** pour comparer le mot proposé avec chaque mot du dictionnaire l'un après l'autre.

Quel nombre d'itérations minimum, maximum et moyen doit on faire pour vérifier la présence du mot proposé dans le dictionnaire ?

- On considère maintenant que le dictionnaire ***est ordonné.***

Quel nombre d'itérations minimum, maximum et moyen doit on faire pour vérifier la présence du mot proposé dans le dictionnaire ?

Avec ces considérations, pouvez-vous envisager une amélioration de l'algorithme de recherche d'un mot ? Il y a plusieurs pistes possibles, mais certaines peuvent être difficiles à implémenter avec les seules notions de C abordées jusqu'à présent. Choisissez donc une solution que vous pourrez implémenter. Vous devrez tout de même justifiez votre choix par des calculs précis (en vous aidant par exemple des données du contenu du dictionnaire). Vous devrez aussi estimé les structures des données supplémentaires à mettre en place.

2- Vous devez coder ensuite les fonctions suivantes (voir le détail dans le fichier fonction.h) :

int estMotValide(char mot[]);	Difficulté algorithmique : ★☆☆☆	Difficulté technique : ★☆☆☆
int chargeMots(char chemin[]);	Difficulté algorithmique : ★☆☆☆	Difficulté technique : ★★☆☆
int chercheMot(char mot[],int nbMots);	Difficulté algorithmique : ★★★☆	Difficulté technique : ★☆☆☆

int lireMot();	Difficulté algorithmique : ☆☆☆☆	Difficulté technique : ★★☆☆
int verifieMot(char proposition[],char reference[]);	Difficulté algorithmique : ★★★☆	Difficulté technique : ★☆☆☆

3 -Passer les ***tests*** fournis et vérifier les résultats. Éventuellement enrichir les ***jeux de tests***.

4- Générer à partir des c***ommentaires du code***, la documentation automatique via le wizard ***doxygen***, pour cela vous devrez étudier les différentes options de configuration.

5- Être capable de réaliser une ***démonstration de votre programme et une explication*** de votre code.