

LIFAP1 – Séquence 1

TP Noté #2 - durée 1h30 mn

Mardi 14 décembre 2021

Sujet A

Consignes

Aucun accès au WEB, aux pages de l'UE, ni à vos anciens TP n'est autorisé. Dans votre fichier, vous mettrez en commentaire vos nom et prénom ainsi que votre numéro d'étudiant.

La note tiendra compte du respect des consignes, de la qualité de la présentation et de la lisibilité du code, des algorithmes, et du bon fonctionnement du programme. **Seules les notions vues en cours devront être utilisées.**

Une fois le programme terminé et testé (ou à la fin du temps imparti), vous devrez déposer le fichier source (.cpp) via **TOMUSS** (en cliquant sur "déposer" dans la case Depot_TP_NOTE de l'UE LIFAP1). Aucun retour par mail ne sera accepté.

Travail à réaliser

Pour réaliser ce TP, vous pourrez utiliser toutes les fonctionnalités de la bibliothèque `string.h` et vous devrez utiliser les bibliothèques `stdlib.h` et `time.h` pour utiliser la fonction `rand()`.

Nous allons développer une application de génération automatique de phrases.

Un mot est caractérisé par une chaîne de caractères `chaîne`, sa longueur `lg_mot` et le nombre de voyelles `nb_voy` qui le constituent. Une phrase est composée d'un tableau de mots noté `tab_mots` et du nombre de mots `nb_mots` dans la phrase.

- 1- Définir en langage C/C++ deux constantes nommées `MAXCH` et `MAXMOTS` ayant pour valeurs respectives 100 et 50.
- 2- Définir les structures `mot` et `phrase`.
- 3- Ecrire une **fonction** `compte_voyelles` qui compte et retourne le nombre de voyelles dans une chaîne de caractères passée en paramètre.
- 4- Ecrire une **fonction** `genere_mot` qui génère automatiquement un mot en respectant les contraintes suivantes et le retourne.
 - La longueur du mot `lg_mot` sera déterminée aléatoirement entre 2 et 10 lettres.
 - Chaque lettre sera générée aléatoirement parmi **les caractères alphabétiques minuscules**.
 - La génération des caractères sera recommencée tant que le mot ne comporte pas un minimum de `lg_mot/2` voyelles. On pourra utiliser le sous-programme précédent.
- 5- Ecrire une procédure `affiche_mot` qui affiche le mot généré ainsi que ses caractéristiques.
- 6- Ecrire une **procédure** `ajoute_un_mot` qui ajoute un mot à la phrase s'il reste de la place dans le tableau de mots. On utilisera le sous-programme écrit en 4-.
- 7- Ecrire une procédure `affiche_phrase` permettant d'afficher tous les mots d'une phrase en les séparant par des espaces.
- 8- Ecrire un sous-programme `statistiques` qui à partir d'une phrase passée en paramètre, "retourne" le nombre de voyelles totales `total_voyelles` ainsi que la moyenne des longueurs des mots `moy_lg` composant cette phrase.
- 9- Ecrire le programme principal de l'application permettant
 - d'ajouter autant de mots que l'utilisateur le voudra,
 - d'afficher à chaque ajout la phrase générée automatiquement (en utilisant les sous-programmes écrits précédemment),
 - et d'afficher le nombre total de voyelles et la longueur moyenne des mots de la phrase.

```
ypesuoiaw
encore un mot ? 'o' ou 'o' pour oui
o
ypesuoiaw chizueg
encore un mot ? 'o' ou 'o' pour oui
o
ypesuoiaw chizueg teupeqpig
encore un mot ? 'o' ou 'o' pour oui
o
ypesuoiaw chizueg teupeqpig ohysa
encore un mot ? 'o' ou 'o' pour oui
o
ypesuoiaw chizueg teupeqpig ohysa tdxyyioai
encore un mot ? 'o' ou 'o' pour oui
n
longueur moyenne des mots : 8.2 et nombre total de voyelles : 23

Process returned 0 (0x0)   execution time : 5.516 s
Press any key to continue.
```