LIFAPI – TD 10 : Encore des chaînes de caractères

Objectifs : Utiliser les chaînes de caractères dans des programmes plus complexes
Manipuler des chaînes de caractères dans des tableaux

En C/C++, bibliothèque contenant des sous-programmes facilitant la gestion des chaînes de caractères ➔ **string.h**

Quelques fonctionnalités

- lg = strlen (chaine) ➔ calcule et retourne la longueur de la chaîne passée en paramètre
- strcat(ch1,ch2) ➔ concatène ch1 et ch2. Résultat dans ch1
- strcpy (ch1,ch2) ➔ copie ch2 dans ch1, en écrasant ch1
- strcmp (ch1,ch2) ➔ compare ch1 et ch2. 0 si elles sont identiques, 1 si ch1> ch2, -1 sinon

1. Écrire l'algorithme d'une procédure qui prend une chaîne de caractères et construit une nouvelle chaine où toutes les voyelles de la chaîne donnée ont été supprimées.
   Exemple : sans_voyelle("programmation") ➔ "prgrmmtn"

2. Écrire en langage algorithmique un sous-programme permettant de compter et "renvoyer" au programme appelant le nombre de majuscules, de minuscules et de voyelles dans une chaîne de caractères passée en paramètre.

3. Nous allons écrire quelques outils de manipulation de chaînes de caractères. Dans cet exercice, vous pourrez utiliser la fonction `aleatoire()` qui retourne un entier compris entre 0 et une constante `MAX = 32767` exclue (fonction équivalente à la fonction `rand()` du C++), ainsi que la fonction `longueur` qui retourne la longueur d'une chaîne de caractères passée en paramètre.
   a. Ecrire l'algorithme d'un sous-programme `genere_chaine` qui construit et "retourne" une chaîne de caractères de longueur `lg`, telle que tous les caractères d'indices **pairs** seront écrits en **minuscules** et tous les caractères d'indices **impairs** seront écrits en **majuscules**. La longueur de la chaîne sera passée en paramètre. Les caractères seront générés de manière aléatoire uniquement parmi les caractères alphabétiques. On suppose qu'une constante `CHMAX = 64` a été définie au préalable pour la taille de la chaîne.
      Exemple de chaîne générée avec `lg = 7` ➔ `zMsVaUq`
   b. Ecrire l'algorithme d'un sous-programme `inverse_casse` qui, à partir d'une chaine de caractères passée en paramètre, construit une nouvelle chaîne dans laquelle toutes les minuscules sont transformées en majuscules et vice-versa.
      Exemple sur la chaîne précédente ➔ `ZmSvAuQ`
   c. Ecrire l'algorithme d'une **fonction booléenne** `compare` qui retourne vrai si les deux chaînes de caractères passées en paramètres sont identiques **quelle que soit la casse** (majuscule ou minuscule), faux sinon. L'utilisation d'un équivalent de `strcmp` est interdite.
      Par exemple `compare("zMsVaUq", "ZmSvAuQ")` retournera vrai mais `compare("HeLlO", "BoNjOuR")` renverra faux.

.