TP - Les chaines de caractères

Objectif(s)

- ✯ Découvrir les traitements sur les chaînes de caractères et l'utilisation d'une pile.
- ✯ Réutiliser un peu de structures et de passage par adresse.

Exercices obligatoires

Exercice 1

Rappelons qu'une chaîne de caractères est un tableau de caractères normal, dont la fin est marquée par le caractère spécial `'\0'`. Elle peut se déclarer comme un tableau ou simplement en utilisant les `"`, avec `char chaine[] = "bonjour";`.

Question 1

Écrivez une fonction `longueur` qui prend en paramètre une chaîne de caractères (un tableau de caractères) et qui retourne sa longueur, en comptant le nombre de caractères qui précède le caractère '`\0`'.

Question 2

Écrivez une fonction `teste_longueur` qui teste la fonction précédente sur quelques chaînes de caractères et n'affiche un message d'erreur que si la longueur obtenue n'est pas celle attendue. Vérifiez la longueur de la chaîne correspondant à un simple caractère accentué ou à un retour à la ligne.

Question 3

Écrivez deux fonctions `en_minuscule` et `en_majuscule` qui prennent une chaîne de caractère en paramètre et convertissent tous les majuscules en minuscules et vice versa. Il faut noter que les caractères sont codés comme des entiers (leur valeur ascii), et on peut donc additionner des caractères (`'c'- 'a'+ 'A'` donne `'C'`), et les comparer (`'a'< 't'` est vrai).

On peut lire une chaîne dans l'entrée standard avec un simple `scanf("%s", tab)`. Malheureusement, en procédant ainsi, on ne sait pas limiter le nombre de caractères lus et on risque de déborder de la chaîne et de déclencher une erreur. Vous pouvez tester avec le code déposé sur e-campus...

Question 4

Écrivez une fonction `lire_chaine_protegee` qui prend en paramètre une longueur maximum de chaine et un tableau, et lis une chaîne de l'entrée standard caractère par caractère jusque la fin de ligne ou la longueur max autorisée, afin de ne pas dépasser du tableau. La chaîne est enregistrée dans le tableau reçu en paramètre, sans oublier le caractère `'\0'` pour finaliser la chaîne. La fonction retourne la longueur de la chaîne lue.

Par exemple, si le tableau passé en paramètre est de taille 4 et que l'utilisateur écrit sur le terminal `"bonjour"`, le tableau retourné contiendra `"bon\0"` et la valeur retournée sera 3. Si en revanche le tableau passé en paramètre est de taille 10 pour la même entrée dans le terminal, le tableau retourné contiendra `"bonjour\0"` et la valeur retournée sera 7.

Question 5

Dans la suite, nous allons utiliser des chaînes de caractères assez longues, autant les collecter dans un fichier. Testez la redirection de l'entrée d'un programme avec un fichier (texte), en utilisant dans le terminal la commande `./programme < fichier.txt` (ou programme est le nom de votre exécutable, et fichier.txt le nom de votre fichier texte). Mettez à jour votre fonction `lire_chaine_protegee` pour qu'elle gère les fins de chaînes (`'\0'`) mais aussi les fins de fichiers (`EOF`) qu'elle convertit en fin de chaîne.

Exercice 2 – Codage de César

Question 1

Écrivez une fonction qui prend en paramètre une chaîne de caractère, et permet de récupérer le caractère le plus fréquent et sa fréquence. On peut convertir facilement un caractère en entier, et récupérer son numéro ascii, en le traitant comme un `int`.

Question 2

Écrivez un programme qui prend en entrée une chaîne de caractère et décale toute les lettre (alphabétiques) d'un décalage passé en paramètre. Chiffrez ainsi une phrase sans indiquer le décalage, et envoyez la à votre voisin. (Le calcul `'a'+ 3` donne un `'d'`)

Question 3

Déchiffrez la phrase reçue en supposant que la lettre la plus fréquente est le 'e'.

Question 4

Écrivez un programme d'aide au déchiffrement qui affiche le premier mot pour chaque décalage et demande à l'utilisateur de donner le décalage.

Exercice 3 – En voiture

On va définir une structure voiture qui a une position et une vitesse, considérées entières.

Question 1

Déclarez une structure `voiture` qui comportera 4 champs entiers, deux de position (posX et posY), deux de vitesse (vitX et vitY).

Question 2

Écrivez une fonction `un_pas` qui prend la structure en paramètre et retourne la structure dans laquelle la position est mise à jour (la position dans chaque direction devient la position + la vitesse).

Question 3

Écrivez une fonction `teste_un_pas` qui teste la précédente, en déclarant une structure avec comme position $(X = 3, Y = 7)$ et comme vitesse $(vX = 1, vY = -2)$ et qui vérifie qu'après un appel de la fonction, la voiture retournée a les bonnes positions et une vitesse inchangée. Elle affichera "echec" en cas d'échec.

Question 4

On dit que deux voitures se rapprochent si après un pas, la distance selon chacune des coordonnées diminue. Écrivez une fonction `se_rapprochent` qui prend en paramètres deux voitures, et qui vérifie si elles s'éloignent ou se rapprochent. La fonction retournera 1 si les voitures se rapprochent, 0 sinon.

Question 5

Deux voitures entrent en collision si elles occupent le même emplacement, à un certain pas de temps. Écrivez une fonction `collision` qui prend en paramètres deux voitures, et qui vérifie si elles entrent en collision. Si elles entrent en collision, la fonction retourne le temps restant avant la collision (donc 0 si elles sont sur le même emplacement), et retourne -1 sinon.

Renforcements

Exercice 4 – Chiffrement ++

Nous avons débuté le codage de César en début de TP. Il faut tout de même formater un peu le texte d'entrée pour rendre le chiffrement plus efficace.

Question 1

Écrivez une fonction `formate_chaine` qui prend en paramètre une chaîne de caractères et retire les caractères alphabétiques accentués, en les convertissant en les caractères sans accents.

Question 2

Dans le chiffrement de Vigenère, on utilise une clé (un mot). Chaque caractère du message original est décalé avec un décalage dépendant de la clé. Par exemple, si la clé est "iut", la première lettre subira le décalage qui transforme le 'a' en 'i', la deuxième le 'a' en 'u', la troisième le 'a' en 't', puis on cycle avec de nouveau le 'a' en 'i', et ainsi de suite. Écrivez la fonction `chiffre_vigenere` qui prend en paramètre le message et la clé, et encode le message, ainsi que la fonction `dechiffre_vigenere` qui fait l'opération inverse.