R1.01 – TD6 – partie #2

Exercice 1 : comptage de doublons dans une chaîne de caractères - V2

L'approche envisagée ici en est totalement différente : l'algorithme de haut niveau proposé consiste à se positionner successivement sur les doublons successifs, et à les comptabiliser au fur et à mesure. L'algorithme se termine lorsque plus aucun doublon n'est trouvé.

L'analyse est rigoureusement analogue à celle qui a été étudiée dans "Fonction de comptage d'un caractère dans une chaîne de caractères". Elle passe par l'écriture de la fonction `findFirstDoublon()`, de profil :

```
fonction findFirstDoublon (chaine : in string,
                           debut  : in entier_naturel)
    renvoie entier_naturel;
```

qui renvoie la position du **premier** caractère du **premier** doublon trouvé dans `Chaine` **à partir** de la position initiale `debut`. Elle renvoie `taille (chaine)` si aucun doublon n'est trouvé.

Travail demandé

Ecrire le corps de la fonction `findFirstDoublon()`, puis écrire l'algorithme qui compte le nombre de doublons dans une chaîne saisie au clavier.

Remarques

1. prendre n'importe laquelle des définitions des doublons de l'exercice précédent;
2. la ligne peut être vide.

Exercice 2 : recherche d'une sous-chaîne dans une chaîne de caractères

Comme on l'a vu dans l'exercice "Comptage de doublons dans un tableau - V2", le comptage d'un doublon (couple de lettres consécutives identiques) peut être considéré comme la répétition de recherches successives de ce doublon à partir de la position du doublon précédemment trouvé.

Un doublon peut être considéré comme un **motif** (***pattern***) connu par sa **propriété**.

On peut envisager des motifs définis par leur **valeur**, exemple "Comptage des suites le, les, lle".

La forme générale de l'algorithme est totalement identique, seule change la reconnaissance du **motif**.

Lorsque le motif dépasse deux caractères, il devient lourd et maladroit de mémoriser individuellement les caractères qui composent le motif. Il est préférable de considérer que l'on cherche une sous-chaine (de 2, 3 lettres ou plus) dans une chaîne (ou un sous-tableau dans un

tableau). C'est ce qui vous est proposé dans l'exercice ci-dessous.

Ecrire la fonction `findSubstrInStr()` qui renvoie le rang de la première apparition d'une sous-chaîne dans une chaîne de caractères, à partir d'un rang de début de recherche, tous trois passés en paramètres. Plus précisément, elle renvoie le rang du premier caractère de la sous-chaîne dans la chaîne.

La valeur de retour est obligatoirement dans l'intervalle `[0, taille (chaine) - 1]` si la sous-chaîne est présente. On choisira donc de renvoyer la valeur `taille (chaine)` si la sous-chaîne n'a pas été trouvée.

Ecrire l'algorithme qui saisit au clavier une ligne, puis dans une boucle :

- saisit une sous-chaîne jusqu'à une sous-chaîne vide,
- l'affiche **entre guillemets**, suivie de son rang dans la chaîne si elle est présente, ou d'un message indiquant qu'elle n'a pas été trouvée. Utiliser évidemment la fonction `findSubstrInStr()` !

La solution qui est demandée ici est la plus simple : la sous-chaîne est comparée à la chaîne à partir de la première position de la chaîne. Si la coïncidence n'est pas totale, elle est de nouveau comparée à la chaîne à partir de la deuxième position, etc.