

LIFAP1 – TD 8 : Chaines de caractères

Objectifs : Se familiariser avec les chaînes de caractères, apprendre les manipulations de base et appliquer les techniques et compétences acquises lors des TD sur les tableaux.

1. Écrire l'algorithme d'une fonction qui compte le nombre de caractères dans une chaîne (sans utiliser la fonction prédéfinie *strlen*)

Ex : longueur("bonjour") renvoie 7

```
Fonction longueur (mot : chaine de 20 caractères) : entier
Précondition : aucune
Donnée / résultat : mot
Résultat : longueur de la chaine
Variables locales : i :entier , lg : entier
Début
    lg ← 0
    i ← 0
    tant que mot[i] ≠ '0' faire
        lg ← lg + 1
        i ← i + 1
    fin tant que
    Retourner lg
Fin longueur
```

2. Écrire l'algorithme d'une procédure qui affiche une chaîne de caractère en inscrivant un caractère par ligne

Ex : affiche("hello") ➔ h

e
l
l
o

```
Procédure affichage_Vertical (mot : chaine de 20 caractères)
Précondition : aucune
Donnée / résultat : mot
Variables locales : i :entier
Début
    Pour i allant de 0 à longueur(mot) – 1 par pas de 1 faire
        Afficher (mot[i])
        Afficher (saut de ligne)
    Fin Pour
Fin affichage_vertical
```

3. Écrire l'algorithme d'une procédure qui construit dans une nouvelle chaine le miroir d'une chaîne de caractères.

Ex : le miroir de "bonjour" est "ruojnob"

```
Procédure miroir (mot : chaine de 20 caractères, miroir : chaine de 20 caractères)
Précondition : aucune
Donnée / résultat : mot, miroir
Description : miroir de la chaine
Variables locales : i :entier , lg : entier
```

```

Début
  lg ← longueur(mot)
  i ← 0
  tant que mot[i] ≠ '\0' faire
    miroir[lg-i-1] ← mot[i]
    i ← i + 1
  fin tant que
  miroir[lg] ← '\0'
Fin miroir

```

4. Écrire l'algorithme d'une fonction qui retourne le nombre d'occurrences d'une lettre dans une chaîne de caractères

Ex : nb_occurrence("bonjour",'o') → 2 nb_occurrence("bonjour",'z') → 0

```

Fonction nb_occurrence (mot : chaine de 20 caractères, c : caractere) : entier
Précondition : aucune
Donnée : c
Donnée / résultat : mot
Résultat : nbre d'occurrences de c dans mot
Variables locales : i : entier , nbc : entier
Début
  i ← 0
  nbc ← 0
  tant que mot[i] ≠ '\0' faire
    si mot[i] = c alors nbc=nbc+1
  fin si
  i ← i+1
  fin tant que
  Retourner nbc
Fin nb_occurrence

```

5. Écrire l'algorithme d'une fonction qui teste si une chaîne passée en paramètre est un palindrome ou non.

Ex : palindrome("eluparcettecrapule") → Vrai palindrome("bonjour") → Faux

Version utilisant la procédure miroir (Q3) :

```

Fonction palindrome (mot : chaine de 20 caractères) : booléen
Précondition : aucune
Donnée / résultat : mot
Résultat : booleen indiquant si mot est un palindrome
Variables locales : mir : chaine de 20 caractères
Début
  miroir(mot,mir)
  si mot = mir
    alors retourner vrai
    sinon retourner faux
  fin si
Fin
Attention la comparaison de mots en C++ : if (strcmp (mot,miroir) == 0)...

```

Version sans utiliser la procédure miroir :

```

Fonction palindrome (mot : chaine de 20 caractères) : booléen
Précondition : aucune
Donnée / résultat : mot
Résultat : booleen indiquant si mot est un palindrome
Variables locales : res: boolean
Début
  lg ← longueur(mot)

```

```

i ← 0
res ← vrai
tant que ((i < lg/2) et res) faire
    si mot[i] ≠ mot[lg-i-1]
        alors res ← false
    fin si
    i ← i+1
fin tant que
retourner res
Fin

```

6. Écrire l'algorithme d'une procédure qui prend une chaîne donnée en minuscules et construit la chaîne équivalente en majuscules, sans changer les caractères non-alphabétiques.

Ex : min2maj("bonjour") → "BONJOUR"

```

Procédure min2maj (mot : chaîne de 20 caractères, mot_maj : chaîne de 20
caractères)
Précondition : aucune
Donnée / résultat : mot, mot_maj
Description : mot en majuscules
Variables locales : i : entier , lg : entier
Début
    lg ← longueur(mot)
    Pour i allant de 0 à lg -1 par pas de 1 faire
        Si mot[i] ≥ 'a' et mot[i] ≤ 'z'
            alors mot_maj[i] ← mot[i] - 'a' + 'A'
            sinon mot_maj[i] ← mot[i]
        FinSi
    Fin pour
    mot_maj[longueur(mot)] ← '\0'
Fin

```

7. Écrire l'algorithme d'une procédure qui permettant prend une chaîne de caractères et construite une nouvelle chaîne où toutes les voyelles de la chaîne donnée ont été supprimées.

Exemple : sans_voyelle("programmation") → "prgrmmtn"

```

Procédure sans_voyelle (mot : chaîne de 20 caractères, voy : chaîne de 20
caractères)
Précondition : aucune
Donnée / résultat : mot, voy
Variables locales : ind_mot, ind_voy : entier , lg : entier
Début
    ind_voy ← 0
    lg ← longueur(mot)
    Pour ind_mot allant de 0 à lg -1 par pas de 1 faire
        Si (mot[ind_mot] ≠ 'a' et mot[ind_mot] ≠ 'e' et mot[ind_mot] ≠ 'i' et mot[ind_mot] ≠ 'o' et
mot[ind_mot] ≠ 'u' et mot[ind_mot] ≠ 'y')
            alors voy[ind_voy] ← mot[ind_mot]
            ind_voy ← ind_voy +1
        FinSi
    Fin pour
    voy[ind_voy] ← '\0'
Fin

```

8. 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.

```
Procédure compte (mot : chaîne de 20 caractères, min : entier, maj : entier, voy :
entier)
Précondition : aucune
Donnée / résultat : mot, min, maj, voy
Variables locales : i, lg : entier
Début
    min ← 0
    maj ← 0
    voy ← 0
    lg ← longueur(mot)
    Pour i allant de 0 à lg -1 par pas de 1 faire
        Si (mot[i] >= 'a' et mot[i] <= 'z')
            alors min ← min + 1
        Sinon si (mot[i] >= 'A' et mot[i] <= 'Z')
            alors maj ← maj + 1
        FinSi
    Fin si
    Si (mot[i] = 'a' ou mot[i] = 'e' ou mot[i] = 'i' ou mot[i] = 'o' ou mot[i] = 'u' ou mot[i] = 'y' ou
    mot[i] = 'A' ou mot[i] = 'E' ou mot[i] = 'I' ou mot[i] = 'O' ou mot[i] = 'U' ou mot[i] = 'Y')
        alors voy ← voy + 1
    FinSi
Fin pour
Fin
```

Pour s'entraîner

1. Codage / décodage de césar : il s'agit de transformer une chaîne de caractères en remplaçant chaque lettre du texte original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Par exemple si on choisit un décalage de 3 le 'a' sera remplacé par le 'd' le 'b' par le 'e' et ainsi de suite jusqu'au 'z' qui sera quant à lui remplacé par le 'c'.

Écrire en langage algorithmique un sous-programme permettant d'effectuer le codage de césar.

Exemple CESAR → FHVDU

```
Procédure cesar (mot : chaîne de 20 caractères, mot_code : chaîne de 20 caractères)
Précondition : aucune
Donnée / résultat : mot, mot_code
Variables locales : i, lg : entier
Début
    lg ← longueur(mot)
    i ← 0
    Tant que i < lg faire
        Si ((mot[i] >= 'a' et mot[i] <= 'w') ou (mot[i] >= 'A' et mot[i] <= 'W'))
            alors mot_code[i] ← mot[i] + 3
        Sinon
            Si (((mot[i] >= 'x' et mot[i] <= 'z') ou (mot[i] >= 'X' et mot[i] <= 'Z'))
                alors mot_code[i] ← mot[i] - 26 + 3
            sinon mot_code[i] ← mot[i]
        FinSi
    FinSi
    i ← i + 1
Fin Tant que
mot_code[i] = '\0'
Fin
```