LIFAP1 – TD 10 : Chaînes 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

Déclaration : mot chaine[100] de caractères
Lecture au clavier : lire (mot)
Affichage de la chaine : afficher(mot)
Accès à la lettre de la case 4 : mot[4]
Remarques
Toutes les cases ne sont pas nécessairement remplies
Caractère de fin de chaîne : '\0'
Elles sont toujours données/résultats, mais on ne met pas "&" en C/C++
Le premier indice est 0

Faire un programme principal qui appelle les différents sous programmes au fur et à mesure.

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

```
Procedure 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
    finsi
    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" min2maj("a2mains") ➔ "A2MAINS"

```
Procédure min2maj (mot : chaine de 20 caractères, mot_maj : chaine de 20
caractères)
Précondition : aucune
Donnée / résultat : mot, mot_mat
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
```

Pour s'entraîner

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. Ex : CESAR ➔ FHVDU