2023/2024

Rapport De TP

Manipulation DE MOTS

HEBALI

NOUR EL HOUDA

GROUP : G3

Théorie de Language

Considérons un langage défini sur l'alphabet A= (a,b).

I.1.1 Commentair :

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main() {
    char mot[100]; // Déclare une chaîne de caractères pour stocker le mot

    printf("Entrer un mot: "); // Demande à l'utilisateur de saisir un mot
    gets(mot); // Lit le mot saisi par l'utilisateur

    printf("Le résultat est %u \n", strlen(mot));
      // Affiche la longueur du mot saisi en utilisant la fonction
                  strlen() de la bibliothèque string.h

      // %u est le format de spécificateur pour unsigned int

    system("PAUSE");
    // Met en pause l'exécution pour que l'utilisateur puisse voir le
                résultat avant que la console ne se ferme


    return 0; // Termine le programme avec succès
}
```

Analyse du programme :

1) Problème à Résoudre :

- Demande à l'utilisateur de saisir un mot.
- Affichage de la longueur de ce mot.

2) Conception de la solution :

- Vous pourriez envisager d'inclure des vérifications de validation des entrées pour vous assurer que l'utilisateur saisit un mot valide.
- Pensez à d'autres méthodes potentielles pour calculer la longueur du mot, par exemple en utilisant des boucles plutôt que la fonction strlen().

3) Mise en œuvre :

- Le programme est implémenté en langage C.
- Il utilise les bibliothèques standard stdio.h et string.h.
- La longueur maximale du mot est définie à l'aide d'une macro pour assurer la lisibilité et la maintenance du code.

4) Tests Effectués :

- Test avec les mots fournis dans l'énoncé et d'autres mots pour garantir la fiabilité et la précision.

5) Optimisation du Programme :

- Utilisation de fgets() pour éviter les débordements de tampon.
- Assurer une sortie formatée correcte en utilisant le format de spécificateur approprié dans printf().

I.1.1 Exécution :

1) Pour le mot "aaaba", la sortie sera 5.
2) Pour le mot "bbba", la sortie sera 4.
3) Pour le mot "a", la sortie sera 1.
4) Pour le mot "abab", la sortie sera 4.
5) Pour le mot "aabbaaabb", la sortie sera 9.
6) Pour le mot "bababababababaaa", la sortie sera 14.

I.2 Mots Sélectionnés :

1. Mot : "mascara"
   - La longueur du mot est 7
2. Mot : "informatique"
   - La longueur du mot est 11
3. Mot : "Theorie-De-Language"
   - La longueur du mot est 19
4. Mot : "i"
   - La longueur du mot est 1

I.3 Que Fait ce Programme?

L'utilisateur est invité à saisir un mot. Par la suite, le programme calcule et affiche la longueur de ce mot en comptant le nombre de caractères qu'il contient.

Pour afficher correctement la longueur du mot, nous devons utiliser le format de spécificateur approprié dans l'instruction printf. Actuellement, le format de spécificateur utilisé est %u, destiné à un entier non signé. Cependant, la fonction strlen() renvoie un type size_t, généralement défini comme unsigned long.

Pour remédier à cela, nous devons ajuster le format de spécificateur de printf en conséquence. Voici la modification recommandée :

```
1 printf("La longueur du mot est : %lu\n", strlen(mot));
```

Le format de spécificateur %lu est utilisé pour afficher un unsigned long. Cela garantit que la sortie affiche correctement la longueur du mot.

I.4 Modification :

```c
#include <stdio.h>
#include <string.h>

int main() {
 char mot[100]; // Déclare une chaîne de caractères pour stocker le mot saisi par l'utilisateur

    // Demande à l'utilisateur de saisir un mot
    printf("Entrez un mot : ");

     // Utilisation de fgets pour lire l'entrée de manière sécurisée et éviter les débordements de tampon
    fgets(mot, sizeof(mot), stdin);

      // Supprime le caractère de nouvelle ligne (\n) ajouté par fgets s'il est présent
    if (strlen(mot) > 0 && mot[strlen(mot) - 1] == '\n') {
        mot[strlen(mot) - 1] = '\0';
    }

    int nb_b = 0; // Variable pour stocker le nombre de 'b' dans le mot
    int nb_a = 0; // Variable pour stocker le nombre de 'a' dans le mot

    // Parcourt chaque caractère du mot pour compter les 'b' et les 'a'
    for (int i = 0; mot[i] != '\0'; i++) {
        if (mot[i] == 'b') {
            nb_b++; // Incrémente le compteur si le caractère est 'b'
        } else if (mot[i] == 'a') {
            nb_a++; // Incrémente le compteur si le caractère est 'a'
        }
    }

    // Affiche le nombre de 'b' et de 'a' dans le mot
    printf("Nombre de 'b' : %d\n", nb_b);
    printf("Nombre de 'a' : %d\n", nb_a);

    return 0;
}
```

I.5 **L1 :**

Pour développer un programme en langage C vérifiant si un mot appartient au langage L1, défini comme celui des mots débutant et se terminant par 'a', nous devons procéder comme suit :

1. Invite l'utilisateur à saisir un mot.
2. Vérifie si le premier caractère du mot est 'a' et si le dernier caractère est également 'a'.
3. Affiche un message indiquant si le mot appartient à L1 ou non.

Le Programme :

```c
#include <stdio.h>
#include <string.h>

int main() {
    char mot[100]; // Déclare une chaîne de caractères pour stocker le mot saisi par l'utilisateur

    // Demande à l'utilisateur de saisir un mot
    printf("Entrez un mot : ");

    // Utilisation de fgets pour lire l'entrée de manière sécurisée et éviter les débordements de tampon
    fgets(mot, sizeof(mot), stdin);

    // Supprime le caractère de nouvelle ligne (\n) ajouté par fgets s'il est présent
    if (strlen(mot) > 0 && mot[strlen(mot) - 1] == '\n') {
        mot[strlen(mot) - 1] = '\0';
    }

    // Vérifie si le mot commence et se termine par 'a'
    if (mot[0] == 'a' && mot[strlen(mot) - 1] == 'a') {
        printf("Le mot appartient au langage L1.\n");
    } else {
        printf("Le mot n'appartient pas au langage L1.\n");
    }

    return 0;
}
```

Ce programme vérifie si le mot saisi par l'utilisateur commence et se termine par 'a'. S'il remplit cette condition, il affiche un message confirmant que le mot appartient à L1. Sinon, il affiche un message indiquant que le mot ne fait pas partie de L1.

I.6 L2 :

Pour le langage L2, qui consiste en des mots contenant autant de 'a' que de 'b', nous pouvons élaborer un programme semblable au précédent. Cependant, cette fois, nous devons compter le nombre de 'a' et de 'b' dans le mot, puis vérifier s'ils sont égaux.

Le Programme :

```c
#include <stdio.h>
#include <string.h>

int main() {
 char mot[100]; // Déclare une chaîne de caractères pour stocker le mot saisi par l'utilisateur

    // Demande à l'utilisateur de saisir un mot
    printf("Entrez un mot : ");

    // Utilisation de fgets pour lire l'entrée de manière sécurisée et éviter les débordements de tampon
    fgets(mot, sizeof(mot), stdin);

    // Supprime le caractère de nouvelle ligne (\n) ajouté par fgets s'il est présent
    if (strlen(mot) > 0 && mot[strlen(mot) - 1] == '\n') {
        mot[strlen(mot) - 1] = '\0';
    }

    int nb_a = 0; // Variable pour stocker le nombre de 'a' dans le mot
    int nb_b = 0; // Variable pour stocker le nombre de 'b' dans le mot

    // Parcourt chaque caractère du mot pour compter les 'a' et les 'b'
    for (int i = 0; mot[i] != '\0'; i++) {
        if (mot[i] == 'a') {
            nb_a++; // Incrémente le compteur si le caractère est 'a'
        } else if (mot[i] == 'b') {
            nb_b++; // Incrémente le compteur si le caractère est 'b'
        }
    }

    // Vérifie si le nombre de 'a' est égal au nombre de 'b'
    if (nb_a == nb_b) {
        printf("Le mot appartient au langage L.\n");
    } else {
        printf("Le mot n'appartient pas au langage L.\n");
    }

    return 0;
}
```

Ce programme vérifie si le mot saisi par l'utilisateur contient le même nombre de 'a' que de 'b'. Si tel est le cas, il affiche un message confirmant que le mot appartient à L2. Sinon, il affiche un message indiquant que le mot ne fait pas partie de L2.

I.7 L3 :

Pour le langage $L3$, défini comme le langage des mots de la forme $a*b*$, nous devons vérifier si le mot commence par une séquence de 'a' suivie d'une séquence de 'b', ou si le mot est vide (c'est-à-dire qu'il ne contient que 'a' ou seulement 'b').

Le Programme :

```c
#include <stdio.h>
#include <string.h>

int main() {
 char mot[100]; // Déclare une chaîne de caractères pour stocker le mot saisi par l'utilisateur

    // Demande à l'utilisateur de saisir un mot
    printf("Entrez un mot : ");

    // Utilisation de fgets pour lire l'entrée de manière sécurisée et éviter les débordements de tampon
    fgets(mot, sizeof(mot), stdin);

    // Supprime le caractère de nouvelle ligne (\n) ajouté par fgets s'il est présent
    if (strlen(mot) > 0 && mot[strlen(mot) - 1] == '\n') {
        mot[strlen(mot) - 1] = '\0';
    }

    int i = 0;

    // Parcours le mot jusqu'à ce que 'a' soit trouvé
    while (mot[i] == 'a') {
        i++;
    }

    // Continue à parcourir le mot jusqu'à ce que 'b' soit trouvé
    while (mot[i] == 'b') {
        i++;
    }

    // Vérifie si nous avons atteint la fin du mot
    if (mot[i] == '\0') {
        printf("Le mot appartient au langage L3.\n");
    } else {
        printf("Le mot n'appartient pas au langage L3.\n");
    }

    return 0;
}
```

Ce programme examine si le mot saisi par l'utilisateur débute par une séquence de 'a' suivie d'une séquence de 'b'. Si c'est le cas, il affiche un message confirmant que le mot appartient à L3. Sinon, il affiche un message indiquant que le mot ne fait pas partie de L3.

I.8 **L4 :**

Pour le langage L4, qui consiste en des mots de la forme anbn, nous devons vérifier si le mot commence par une séquence de 'a' suivie d'une séquence de 'b' comportant le même nombre de caractères que la séquence de 'a'.

Le Programme :

```c
#include <stdio.h>
#include <string.h>

int main() {
 char mot[100]; // Déclare une chaîne de caractères pour stocker le mot
                           saisi par l'utilisateur

        // Demande à l'utilisateur de saisir un mot
    printf("Entrez un mot : ");

       // Utilisation de fgets pour lire l'entrée de manière sécurisée et
                      éviter les débordements de tampon
    fgets(mot, sizeof(mot), stdin);

       // Supprime le caractère de nouvelle ligne (\n) ajouté par fgets
                              s'il est présent
    if (strlen(mot) > 0 && mot[strlen(mot) - 1] == '\n') {
        mot[strlen(mot) - 1] = '\0';
    }

    int i = 0;
    int nb_a = 0;
    int nb_b = 0;

      // Compte le nombre de 'a' dans le mot
    while (mot[i] == 'a') {
        nb_a++;
        i++;
    }

      // Compte le nombre de 'b' dans le mot
    while (mot[i] == 'b') {
        nb_b++;
        i++;
    }

       // Vérifie si nous avons atteint la fin du mot et si le nombre de
                        'a' est égal au nombre de 'b'
    if (mot[i] == '\0' && nb_a > 0 && nb_b > 0 && nb_a == nb_b) {
        printf("Le mot appartient au langage L4.\n");
    } else {
        printf("Le mot n'appartient pas au langage L4.\n");
    }

    return 0;
}
```

Ce programme examine si le mot saisi par l'utilisateur commence par une séquence de 'a' suivie d'une séquence de 'b' comportant le même nombre de caractères que la séquence de 'a'. S'il satisfait à cette condition, il affiche un message confirmant que le mot appartient à L4. Sinon, il affiche un message indiquant que le mot ne fait pas partie de L4.

I.9 **Test :**

Pour évaluer chacun des programmes, nous allons exécuter les codes en utilisant des mots appropriés afin de vérifier si les programmes détectent correctement si les mots appartiennent aux langages correspondants.

1) Test pour le langage L1:

- Mot : "asmaa" (commence et se termine par 'a')
- Mot : "houda" (ne commence pas par 'a')
- Mot : "apple" (commence mais ne se termine pas par 'a')

2) Test pour le langage:

- Mot : "abab" (contient autant de 'a' que de 'b')
- Mot : "aaabbb" (contient plus de 'a' que de 'b')
- Mot : "bbbaaa" (contient plus de 'b' que de 'a')

3) Test pour le langage L3:

- Mot : "aabb" (commence par 'a' et se termine par 'b')
- Mot : "aabbaaa" (commence par 'a' mais ne se termine pas par 'b')
- Mot : "bb" (ne commence pas par 'a')

4) Test pour le langage L4:

- Mot : "aabb" (contient le même nombre de 'a' que de 'b')
- Mot : "aaabbb" (contient plus de 'a' que de 'b')
- Mot : "abba" (ne contient pas le même nombre de 'a' que de 'b')

II.

Considérons un langage défini sur l'alphabet B={if,x11,y1,z,(,),=,;}

II.1 calculer la longueur du mot (u) :

Pour élaborer un programme en langage C qui calcule la longueur du mot u="if(x11==y1) z=y1;", nous suivrons une méthode similaire à celle du programme précédent. Nous ferons usage de la fonction 'strlen()' de la bibliothèque string.h afin de déterminer la longueur de la chaîne de caractères.

Le Programme :

```c
#include <stdio.h>
#include <string.h>

int main() {
    char mot[] = "if(x11==y1) z=y1;"; // Déclare le mot u

    // Calcul de la longueur du mot
    int longueur = strlen(mot);

    // Affichage de la longueur du mot
    printf("La longueur du mot u est : %d\n", longueur);

    return 0;
}
```

Ce programme ressemble à celui précédemment décrit. Il détermine la longueur du mot u="if(x11==y1) z=y1;" en utilisant la fonction 'strlen()' de la bibliothèque string.h, puis affiche la longueur du mot sur la sortie standard.

Voici les résultats de l'exécution du programme avec les mots spécifiés :

1. Mot : "if(x11) z=y1;"
   - Longueur du mot : 15
2. Mot : "if(x11==y1) z=z;"
   - Longueur du mot : 18
3. Mot : "x11==y1"
   - Longueur du mot : 8
4. Mot : "y1;"
   - Longueur du mot : 3
5. Mot : "z=z;"
   - Longueur du mot : 5

III.

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main() {
          char mot[100]; // Déclare un tableau de caractères pour
                  stocker le mot saisi par l'utilisateur

    printf("Entrer un mot: "); // Invite l'utilisateur à saisir un mot
    gets(mot); // Lit le mot saisi par l'utilisateur

    int n = strlen(mot); // Calcule la longueur du mot

    // Parcourt le mot de la fin vers le début et affiche chaque
                    caractère
    for (int i = 0; i < n; i++)
        printf("%c", mot[n - i - 1]);

    printf("\n"); // Affiche une nouvelle ligne

   system("PAUSE"); // Met en pause le programme (peut ne pas fonctionner
                           sur tous les systèmes)

    return 0;
}
```

Dans le programme :

- - Nous déclarons un tableau de caractères nommé 'mot' pour stocker le mot saisi par l'utilisateur.
- - L'utilisateur est invité à saisir un mot à l'aide de l'instruction ‘printf()’.
- - Nous utilisons la fonction ‘gets()’ pour lire la chaîne de caractères saisie par l'utilisateur et la stocker dans le tableau ‘mot’.
- - La longueur du mot est calculée à l'aide de la fonction ‘strlen()’ et est stockée dans la variable 'n'.
- - Une boucle ‘for’ est utilisée pour parcourir le mot de la fin vers le début, affichant chaque caractère à l'aide de l'instruction ‘printf()’.
- - Enfin, la commande ‘system("PAUSE")’ est utilisée pour suspendre l'exécution du programme jusqu'à ce qu'une touche soit pressée.

III.1 Que fait ce programme ?

Le programme fonctionne de la manière suivante : Il invite l'utilisateur à saisir un mot. Une fois que l'utilisateur a saisi le mot, le programme le lit. En utilisant la fonction strlen() de la bibliothèque string.h, il calcule la longueur du mot. Ensuite, à l'aide d'une boucle for, le programme parcourt le mot de la fin vers le début.

À chaque itération de la boucle, le programme affiche le caractère correspondant à la position n - i - 1, où n est la longueur du mot et i est l'indice de la boucle. Une fois que tous les caractères ont été affichés en ordre inverse, le programme ajoute une nouvelle ligne. Enfin, le programme utilise la fonction system("PAUSE") pour mettre en pause l'exécution, une pratique courante sur les systèmes Windows pour éviter la fermeture immédiate de la fenêtre de la console.

En résumé, ce programme prend un mot en entrée, inverse l'ordre des caractères de ce mot, puis affiche le mot inversé.

III.2 **Modification :**

Voici le programme modifié :

```c
#include <stdio.h>
#include <string.h>

int main() {
    char mot[100]; // Déclare un tableau de caractères pour stocker le mot saisi par l'utilisateur

    printf("Entrez un mot : "); // Invite l'utilisateur à saisir un mot
    fgets(mot, sizeof(mot), stdin); // Lit le mot saisi par l'utilisateur

    // Supprime le caractère de nouvelle ligne (\n) ajouté par fgets s'il est présent
    if (strlen(mot) > 0 && mot[strlen(mot) - 1] == '\n') {
        mot[strlen(mot) - 1] = '\0';
    }

    int longueur = strlen(mot);
    int estPalindrome = 1; // Variable pour indiquer si le mot est un palindrome

    // Parcourt le mot jusqu'à la moitié de sa longueur
    for (int i = 0; i < longueur / 2; i++) {
        // Compare les caractères du début et de la fin du mot
        if (mot[i] != mot[longueur - i - 1]) {
            estPalindrome = 0; // Si les caractères sont différents, le mot n'est pas un palindrome
            break;
        }
    }

    // Affiche le résultat
    if (estPalindrome) {
        printf("Le mot est un palindrome.\n");
    } else {
        printf("Le mot n'est pas un palindrome.\n");
    }

    return 0;
}
```

IV.

IV.1 Concaténation de mots :

```c
#include <stdio.h>
#include <string.h>

int main() {
    char mot1[100], mot2[100];

    // Demande à l'utilisateur d'entrer le premier mot
    printf("Entrez le premier mot : ");
    fgets(mot1, sizeof(mot1), stdin);
    if (strlen(mot1) > 0 && mot1[strlen(mot1) - 1] == '\n') {
        mot1[strlen(mot1) - 1] = '\0'; // Supprime le caractère de nouvelle ligne
    }

    // Demande à l'utilisateur d'entrer le deuxième mot
    printf("Entrez le deuxième mot : ");
    fgets(mot2, sizeof(mot2), stdin);
    if (strlen(mot2) > 0 && mot2[strlen(mot2) - 1] == '\n') {
        mot2[strlen(mot2) - 1] = '\0'; // Supprime le caractère de nouvelle ligne
    }

    // Concatène les deux mots et affiche le résultat
    strcat(mot1, mot2);
    printf("La concaténation des deux mots est : %s\n", mot1);

    return 0;
}
```

Test du programme de concaténation de mots :

Entrons deux mots et vérifions si le programme concatène correctement les deux mots.

Exemple :

Mot 1 : "Theorie "
Mot 2 : "de Language"

Résultat attendu :

La concaténation des deux mots devrait donner :
" Theorie de Language ".

IV.2 **Affichage des préfixes propres :**

```c
#include <stdio.h>
#include <string.h>

int main() {
    char mot[100];

    // Demande à l'utilisateur d'entrer un mot
    printf("Entrez un mot : ");
    fgets(mot, sizeof(mot), stdin);
    if (strlen(mot) > 0 && mot[strlen(mot) - 1] == '\n') {
mot[strlen(mot) - 1] = '\0'; // Supprime le caractère de nouvelle ligne
    }

    // Affiche tous les préfixes propres du mot
    printf("Les préfixes propres du mot sont :\n");
    for (int i = 0; i < strlen(mot); i++) {
 printf("%.*s\n", i, mot); // Utilise le format %.*s pour afficher les
                           premiers i caractères du mot
    }

    return 0;
}
```

Test du programme d'affichage des préfixes propres :

Entrons un mot et observons comment le programme affiche ses préfixes propres.

Exemple :

Mot : "Programming"

Résultat attendu :

Les préfixes propres du mot "Stambuli" devraient être :

- "S"
- "St"
- "Sta"
- "Stam"
- "Stamb"
- "Stambu"
- "Stambul"
- "Stambuli"

IV.3 **Affichage des suffixes propres :**

```c
#include <stdio.h>
#include <string.h>

int main() {
    char mot[100];

    // Demande à l'utilisateur d'entrer un mot
    printf("Entrez un mot : ");
    fgets(mot, sizeof(mot), stdin);
    if (strlen(mot) > 0 && mot[strlen(mot) - 1] == '\n') {
 mot[strlen(mot) - 1] = '\0'; // Supprime le caractère de nouvelle ligne
    }

    // Affiche tous les suffixes propres du mot
    printf("Les suffixes propres du mot sont :\n");
    for (int i = 1; i < strlen(mot); i++) {
printf("%s\n", &mot[i]); // Affiche le suffixe à partir de la position i
    }

    return 0;
}
```

Test du programme d'affichage des suffixes propres :

Entrons un mot et vérifions comment le programme affiche ses suffixes propres.

Exemple :

Mot : "informatique"

Résultat attendu :

Les suffixes propres du mot "informatique" devraient être :
"informatique"
"nformatique"
"formatique"
"ormatique"
"rmatique"
"matique"
"atique"
"tique"
"ique"
"que"
"ue"
"e"

V. Programme Final :

Le Programme :

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>

#define MAX_LINES 5
#define MAX_LENGTH 100

int main() {
    char texte[MAX_LINES][MAX_LENGTH]; // Tableau pour stocker le texte
char mot[MAX_LENGTH]; // Variable pour stocker chaque mot temporairement

    int majuscules = 0, minuscules = 0, speciaux = 0;
    int mots_total = 0, mots_longueur_4 = 0, mots_contenant_en = 0,
mots_terminant_ent = 0;

    // Saisie du texte ligne par ligne
    printf("Entrez le texte (minimum 5 lignes):\n");
    for (int i = 0; i < MAX_LINES; i++) {
        fgets(texte[i], sizeof(texte[i]), stdin);
    }

    // Parcours du texte ligne par ligne
    for (int i = 0; i < MAX_LINES; i++) {
 char *token = strtok(texte[i], " \n"); // Tokenisation du texte en mots

        // Parcours des mots dans chaque ligne
        while (token != NULL) {
            mots_total++; // Incrémente le nombre total de mots

  // Comptage des caractères majuscules, minuscules et spéciaux dans le
                                    mot
```

```c

 for (int j = 0; j < strlen(token); j++) {
                if (isupper(token[j]))
                    majuscules++;
                else if (islower(token[j]))
                    minuscules++;
                else
                    speciaux++;
            }

            // Vérification de la longueur du mot
            if (strlen(token) == 4)
                mots_longueur_4++;

            // Vérification si le mot contient "en"
            if (strstr(token, "en") != NULL)
                mots_contenant_en++;

            // Vérification si le mot se termine par "ent"
            if (strlen(token) >= 3 && strcmp(&token[strlen(token) - 3],
"ent") == 0)
                mots_terminant_ent++;

            token = strtok(NULL, " \n"); // Passage au mot suivant
        }
    }

    // Affichage des résultats
    printf("\nNombre de mots : %d\n", mots_total);
    printf("Nombre de caractères majuscules : %d\n", majuscules);
    printf("Nombre de caractères minuscules : %d\n", minuscules);
    printf("Nombre de caractères spéciaux : %d\n", speciaux);
    printf("Mots de longueur égale à 4 : %d\n", mots_longueur_4);
    printf("Mots contenant la chaine 'en' : %d\n", mots_contenant_en);
    printf("Mots se terminant par 'ent' : %d\n", mots_terminant_ent);

    return 0;
}
```

Resultat :

```
Entrez le texte (minimum 5 lignes):
Dans le domaine de la programmation en langage C, la manipulation de mots constitue une part essentielle du processus de
 développement. Que ce soit pour le traitement de texte, l'analyse de données ou la création d'algorithmes de recherche
et de tri, la maîtrise des opérations sur les mots est fondamentale.
@gmailcom/"hello""world"

Nombre de mots : 53
Nombre de caractPres majuscules : 3
Nombre de caractPres minuscules : 262
Nombre de caractPres spÛciaux : 18
Mots de longueur Úgale Ó 4 : 7
Mots contenant la chaine 'en' : 5
Mots se terminant par 'ent' : 2

Process returned 0 (0x0)   execution time : 33.123 s
Press any key to continue.
```

MERCI