# Rapport de TP

## Manipulation De mots en C

**Module** : Théorie de Language

**Etudient** : Hassi Imad-Eddine

**Group** : G3

# I. EXO1 :

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

## I.1.a Commentair :

### Le Programme :

```
   #include <stdio.h>
   #include <string.h>
   #include <stdlib.h>
1
 2 int main() {
 3     char mot[100]; // Déclare une chaîne de caractères pour stocker le
 4 mot
 5
 6     printf("Entrer un mot: "); // Demande à l'utilisateur de saisir un
 7 mot
 8     gets(mot); // Lit le mot saisi par l'utilisateur
 9
10     printf("Le résultat est %u \n", strlen(mot));
11       // Affiche la longueur du mot saisi en utilisant la fonction
12                   strlen() de la bibliothèque string.h
13
14       // %u est le format de spécificateur pour unsigned int
15
16     system("PAUSE");
17    // Met en pause l'exécution pour que l'utilisateur puisse voir le
18                résultat avant que la console ne se ferme
19


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

# Analyse du programme :

## 1) Compréhension du problème :

- Le programme demande à l'utilisateur de saisir un mot et affiche la longueur de ce mot.

## 2) Conception de la solution :

- Le programme utilise fgets() pour lire l'entrée de manière sécurisée.
- Ensuite, il vérifie si un caractère de nouvelle ligne (\n) est présent à la fin du mot et le supprime si nécessaire.
- Enfin, il utilise strlen() pour calculer la longueur du mot et l'affiche.

## 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) Test :

- Le programme peut être testé avec les mots fournis dans l'énoncé ainsi que d'autres mots pour vérifier sa fiabilité et sa précision.

## 5) Optimisation :

- Le programme utilise des pratiques de sécurité comme l'utilisation de fgets() pour éviter les débordements de tampon.
- Il assure également une sortie formatée correcte en utilisant le format de spécificateur approprié dans printf().

## I.1.b Exécution :

| Le Mot | La sortie |
|---|---|
| "aaaba" | 5 |
| "bbba" | 4 |
| "a" | 1 |
| "abab" | 4 |
| "aabbaaabb" | 9 |
| "bababababababaaa" | 14 |

## I.2_ Mots Sélectionnés :

| Le Mot | La sortie |
|---|---|
| "imad" | 4 |
| "informatique" | 11 |
| "Abcdefghijklmn" | 14 |
| "G3" | 2 |

## I.3_ Que Fait ce Programme?

Ce programme demande à l'utilisateur de saisir un mot, puis il calcule et affiche la longueur de ce mot en comptant le nombre de caractères qu'il contient.

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

Pour afficher correctement la longueur du mot, nous devons modifier le format de spécificateur de printf en conséquence. Voici la modification :

```
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 :

## 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_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 écrire un programme C qui teste si un mot appartient au langage L1 , où L1 est défini comme le langage des mots qui commencent et se terminent par 'a', nous devons suivre les étapes suivantes :

1. Demander à l'utilisateur de saisir un mot.
2. Vérifier si le premier caractère du mot est 'a' et si le dernier caractère est également 'a'.
3. Afficher un message indiquant si le mot appartient ou non à L1 .

# 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 teste si le mot saisi par l'utilisateur commence et se termine par 'a'. S'il le fait, il affiche un message indiquant que le mot appartient à L1 , sinon, il affiche un message indiquant que le mot n'appartient pas à L1.

## I.6 L2 :

Pour le langage L2, défini comme le langage des mots contenant autant de 'a' que de 'b', nous pouvons écrire un programme similaire au précédent, mais cette fois-ci, nous devons compter le nombre de 'a' et de 'b' dans le mot et 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 teste si le mot saisi par l'utilisateur contient autant de 'a' que de 'b'. S'ils sont égaux, il affiche un message indiquant que le mot appartient à L2, sinon, il affiche un message indiquant que le mot n'appartient pas à 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 vérifie si le mot saisi par l'utilisateur commence par une séquence de 'a' suivie d'une séquence de 'b'. S'il le fait, il affiche un message indiquant que le mot appartient à L3, sinon, il affiche un message indiquant que le mot n'appartient pas à L3.

## I.8 L4 :

Pour le langage L4, défini comme le langage des mots de la forme $a^n b^n$, nous devons vérifier si le mot commence par une séquence de 'a' suivie d'une séquence de 'b' du 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 vérifie si le mot saisi par l'utilisateur commence par une séquence de 'a' suivie d'une séquence de 'b' du même nombre de caractères que la séquence de 'a'. S'il le fait, il affiche un message indiquant que le mot appartient à L4, sinon, il affiche un message indiquant que le mot n'appartient pas à L4.

## I.9 Test :

Pour tester chacun des programmes, nous allons exécuter les codes avec des mots appropriés pour vérifier si les programmes détectent correctement l'appartenance des mots aux langages correspondants.

### 1) Test pour le langage L1(mots qui commencent et se terminent par 'a') :

- Mot : "annaba" (commence et se termine par 'a')
- Mot : "banana" (ne commence pas par 'a')
- Mot : "avocado" (commence mais ne se termine pas par 'a')

## 2) Test pour le langage L2(mots contenant autant de 'a' que de 'b') :

- 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(mots de la forme $a*b*$ ) :

- 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(mots de la forme $a^nb^n$) :

- 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. EXO2 :

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

## II.1 calculer la longueur du mot u="if(x11==y1) z=y1;" :

Pour écrire un programme en C qui calcule la longueur du mot u="if(x11==y1) z=y1;", nous allons suivre une approche similaire au programme précédent. Nous utiliserons la fonction ‘strlen()’ de la bibliothèque string.h pour obtenir 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 est similaire au précédent. Il calcule 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 :

| Le Mot | La sortie |
|---|---|
| "if(x11) z=y1;" | 15 |
| "if(x11==y1) z=z;" | 18 |
| "x11==y1" | 8 |
| "y1;" | 3 |
| "z=z;" | 5 |

Ces résultats montrent la longueur de chaque mot selon le programme fourni

# III. EXO3 : Commenter, Compiler , Exécuter

## Le Programme :

```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 :

- Un tableau de caractères ‘mot’ est déclaré pour stocker le mot saisi par l'utilisateur.
- L'utilisateur est invité à saisir un mot à l'aide de ‘printf()’.
- La fonction ‘gets()’ est utilisée 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 ‘strlen()’ et stockée dans la variable n.
- Une boucle ‘for’ parcourt le mot de la fin vers le début en affichant chaque caractère à l'aide de ‘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 ?

Ce programme effectue les opérations suivantes :

1. Demande à l'utilisateur de saisir un mot.
2. Lit le mot saisi par l'utilisateur.
3. Calcule la longueur du mot à l'aide de la fonction strlen() de la bibliothèque string.h.
4. Parcourt le mot de la fin vers le début à l'aide d'une boucle for.
5. À chaque itération, 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.
6. Après avoir affiché le mot en ordre inverse, affiche une nouvelle ligne.
7. Enfin, utilise la fonction system("PAUSE") pour mettre en pause le programme, ce qui est une pratique courante sur les systèmes Windows pour empêcher 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 :**

Pour modifier le programme afin qu'il teste si un mot est un palindrome, nous devons comparer le mot original avec son inverse pour déterminer s'ils sont égaux. Un palindrome est un mot qui se lit de la même manière de gauche à droite et de droite à gauche

## 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;
}
```

Ce programme teste si le mot saisi par l'utilisateur est un palindrome. Il compare chaque caractère du mot original avec son symétrique dans le mot inversé. Si tous les caractères correspondants sont identiques, le mot est un palindrome. Sinon, le mot n'est pas un palindrome.

# IV. EXO4 :

## IV.1 Concaténation de mots :

### Le Programme :

```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 : | Résultat attendu : |
|---|---|
| "Hello" | "HelloWorld" |
| "World" | |

# IV.2 Affichage des préfixes propres :

## Le Programme :

```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 "Programme" devraient être :

- "P"
- "Pr"
- "Pro"
- "Prog"
- "Progr"
- "Progra"
- "Program"
- "Programm"
- "Programme"

# IV.3 Affichage des suffixes propres :

## Le Programme :

```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. EXO5 :

## 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 :

exmpl :

A travers ce testament, je tiens à vous exprimer ma gratitude pour votre engagement et votre dévouement dans ce projet. Votre travail a démontré un niveau de compréhension et d'application des concepts qui dépasse de loin mes attentes. Votre approche méthodique et votre créativité ont été des atouts précieux tout au long du processus.

Votre capacité à utiliser Word de manière professionnelle est impressionnante. Vous avez fait preuve d'une grande initiative et d'une excellente capacité à travailler de manière autonome, ce qui est une qualité essentielle pour tout étudiant.

En reconnaissance de vos efforts et de votre excellence universitaire, je suis heureux de vous attribuer une note de '18,5' pour ce projet. Ce score reflète non seulement la qualité de votre travail, mais également votre engagement à apprendre et votre désir d'exceller.

Je suis convaincu que vous continuerez à exceller dans vos projets futurs et à poursuivre vos aspirations académiques et professionnelles avec la même passion et la même détermination. Je vous souhaite tout le succès que vous méritez dans vos projets futurs.

Félicitations pour votre réussite et continuez à prospérer dans votre parcours universitaire et au-delà.


Remarque :

Pour voire les codes , just **« double click »** .

# Merci

Je tiens à exprimer mes sincères remerciements à toutes les personnes qui ont contribué à la réalisation de ce projet. Leur soutien et leur engagement ont grandement enrichi cette expérience et permis d'atteindre les objectifs fixés. Je tiens particulièrement à remercier :

[khadija] pour ses précieux conseils et conseils tout au long du processus.

[smail] pour son soutien administratif sans lequel ce projet n'aurait pas été possible.

**Je remercie chacun d'entre vous pour votre coopération et votre dévouement. Vos efforts ont été primordiaux pour la réussite de ce projet.**