

[I. 6](#_Toc160928624)

[I.1.1 Commentair : 6](#_Toc160928625)

[6](#_Toc160928626)

[I.1.1 Exécution : 8](#_Toc160928627)

[I.2 Mots Sélectionnés : 9](#_Toc160928628)

[1. Mot : "mascara" 9](#_Toc160928629)

[2. Mot : "informatique" 9](#_Toc160928630)

[3. Mot : "Theorie-De-Language" 9](#_Toc160928631)

[4. Mot : "I" 9](#_Toc160928632)

[I.3 Que Fait ce Programme? 9](#_Toc160928633)

[I.4 Modification : 11](#_Toc160928634)

[I.5 L1 : 12](#_Toc160928635)

[I.6 L2 : 14](#_Toc160928636)

[L2. 16](#_Toc160928637)

[I.7 L3 : 16](#_Toc160928638)

[I.8 L4 : 18](#_Toc160928639)

[I.9 Test : 20](#_Toc160928640)

[II. 22](#_Toc160928641)

[II.1 calculer la longueur du mot (u) : 22](#_Toc160928642)

[1. Mot : "if(x11) z=y1;" 24](#_Toc160928643)

[2. Mot : "if(x11==y1) z=z;" 24](#_Toc160928644)

[3. Mot : "x11==y1" 24](#_Toc160928645)

[4. Mot : "y1;" 24](#_Toc160928646)

[5. Mot : "z=z;" 24](#_Toc160928647)

[III. 25](#_Toc160928648)

[25](#_Toc160928649)

[III.1 Que fait ce programme ? 26](#_Toc160928650)

[III.2 Modification : 28](#_Toc160928652)

[IV. 29](#_Toc160928653)

[IV.1 Concaténation de mots : 29](#_Toc160928654)

[IV.2 Affichage des préfixes propres : 31](#_Toc160928655)

[IV.3 Affichage des suffixes propres : 33](#_Toc160928656)

[V. Programme Final : 35](#_Toc160928657)

# 

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

### Commentair :

### 



**Analyse du programme :**

* + 1. Problème à Résoudre :

##### Demande à l'utilisateur de saisir un mot.

##### Affichage de la longueur de ce mot.

* + 1. 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().

* + 1. 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.

* + 1. Tests Effectués :

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

* + 1. 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().

### 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 "babababababaaa", la sortie sera 14.

## Mots Sélectionnés :

#### Mot : "mascara"

##### La longueur du mot est 7

#### Mot : "informatique"

##### La longueur du mot est 11

#### Mot : "Theorie-De-Language"

##### La longueur du mot est 19

#### Mot : "i"

##### La longueur du mot est 1

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

## Modification :



## 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 :**

****

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.

## 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 :**

****

## 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.

## L3 :

Pour le langage *L*3​, 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 :**



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.

## 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 :**



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.

## 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')

* + 1. 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')

* + 1. 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')

* + 1. 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')

# 

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

## 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 :**



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 :

#### Mot : "if(x11) z=y1;"

##### Longueur du mot : 15

#### Mot : "if(x11==y1) z=z;"

##### Longueur du mot : 18

#### Mot : "x11==y1"

##### Longueur du mot : 8

#### Mot : "y1;"

##### Longueur du mot : 3

#### Mot : "z=z;"

##### Longueur du mot : 5

# 



**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.

## 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.

## Modification :

**Voici le programme modifié :**

****

# 

## **Concaténation de mots :**



**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 ".

## **Affichage des préfixes propres :**



**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"

## **Affichage des suffixes propres :**



**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"

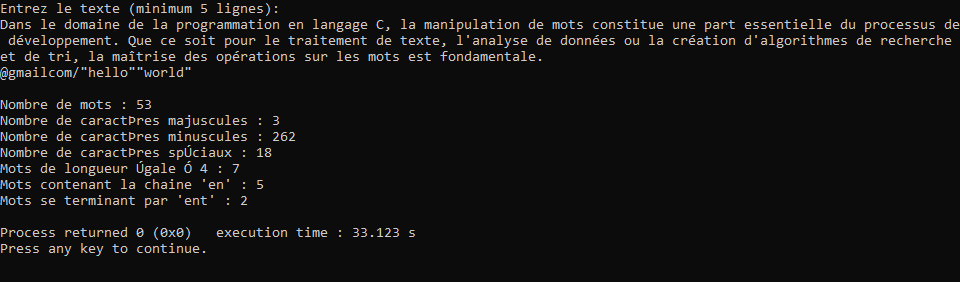
"e"

# Programme Final :

**Le Programme :**

****

****

**Resultat :**

****