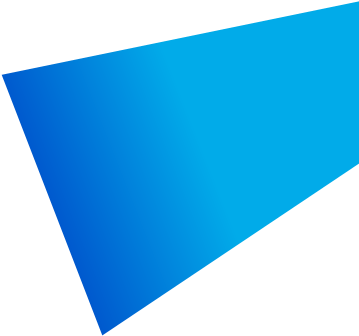
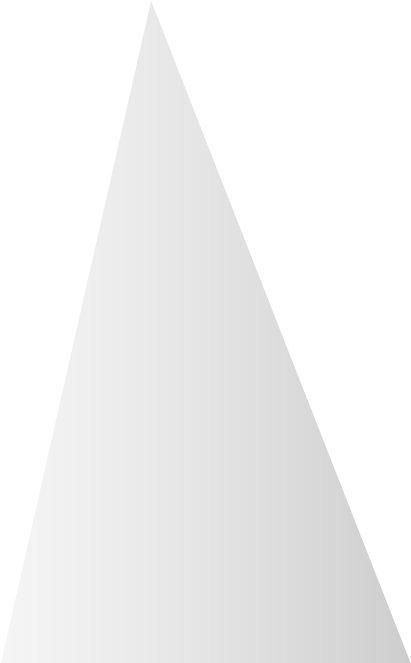
# Rapport



# de TP

Manipulation

De mots

en C

Module : Théorie de Language

Etudient : Hassi Imad-Eddine

Group  : G3

[Rapport 1](#_Toc160927870)

[de TP 1](#_Toc160927871)

[I. EXO1 : 5](#_Toc160927872)

[I.1.a Commentair : 5](#_Toc160927873)

[5](#_Toc160927874)

[I.1.b Exécution : 6](#_Toc160927875)

[I.2 Mots Sélectionnés : 8](#_Toc160927876)

[I.3 Que Fait ce Programme? 8](#_Toc160927877)

[I.4 Modification : 10](#_Toc160927878)

[I.5 L1 : 11](#_Toc160927879)

[I.6 L2 : 13](#_Toc160927880)

[I.7 L3 : 15](#_Toc160927881)

[I.8 L4 : 17](#_Toc160927882)

[I.9 Test : 19](#_Toc160927883)

[II. EXO2 : 21](#_Toc160927884)

[II.1 calculer la longueur du mot u="if(x11==y1) z=y1;" : 21](#_Toc160927885)

[III. EXO3 : Commenter, Compiler , Exécuter 24](#_Toc160927886)

[III.1 Que fait ce programme ? 25](#_Toc160927887)

[III.2 Modification : 27](#_Toc160927888)

[IV. EXO4 : 29](#_Toc160927889)

[IV.1 Concaténation de mots : 29](#_Toc160927890)

[IV.2 Affichage des préfixes propres : 31](#_Toc160927891)

[IV.3 Affichage des suffixes propres : 33](#_Toc160927892)

[V. EXO5 : 35](#_Toc160927893)

# EXO1 :

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

### Commentair :

[**Le Programme :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/I.1.a)



**Analyse du programme :**

* + 1. Compréhension du problème :

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

* + 1. 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.

* + 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. 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.

* + 1. 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().

### Exécution :

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

## Mots Sélectionnés :

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

## 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 :

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

## Modification :

[**Le Programme :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/I.4)



## 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 :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/I.5)

****

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​.

## 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 :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/I.6)

****

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

## 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 :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/I.7)



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​.

## L4 :

Pour le langage L4​, défini comme le langage des mots de la forme *anbn*, 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 :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/I.8)



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​.

## 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')

* + 1. 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')

* + 1. 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')

* + 1. Test pour le langage L4(mots de la forme anbn) :

##### 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')

# EXO2 :

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

## 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 :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/II.1)



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

# EXO3 :Commenter, Compiler , Exécuter

**[Le Programme :](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/III.%20EXO3)**



**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.

## Que fait ce programme ?

Ce programme effectue les opérations suivantes :

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

#### Lit le mot saisi par l'utilisateur.

#### Calcule la longueur du mot à l'aide de la fonction strlen() de la bibliothèque string.h.

#### Parcourt le mot de la fin vers le début à l'aide d'une boucle for.

#### 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.

#### Après avoir affiché le mot en ordre inverse, affiche une nouvelle ligne.

#### 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.

## 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 :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/III.2)

****

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.

# EXO4 :

## **Concaténation de mots :**

[**Le Programme :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/IV.%20EXO4)



**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" |

## **Affichage des préfixes propres :**

[**Le Programme :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/IV.2)



**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"

## **Affichage des suffixes propres :**

[**Le Programme :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/IV.3)



**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"

# EXO5 :

[**Le Programme :**](https://github.com/imad-eddine04/Teorie-de-Language/tree/b5bf970c999a4b5dfe148e2b09c64b22b886e974/Teorie%20de%20Language/TPTHL/TP1/codes/V.%20EXO5)

****

****

**Resultat :**

exmpl :

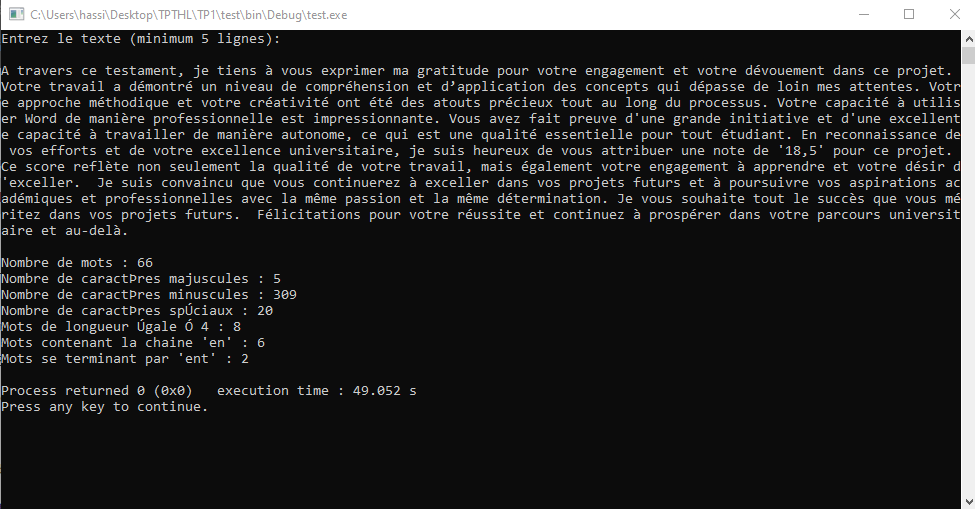
A travers ce testament, je tiens à vous exprimer ma gratitude pour votre engagement et votre dévouement dans ce projet. Votre travail a démontré un niveau de compréhension et d’application des concepts qui dépasse de loin mes attentes. Votre approche méthodique et votre créativité ont été des atouts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

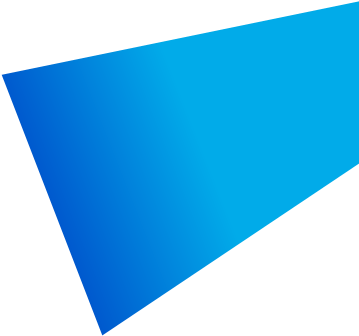
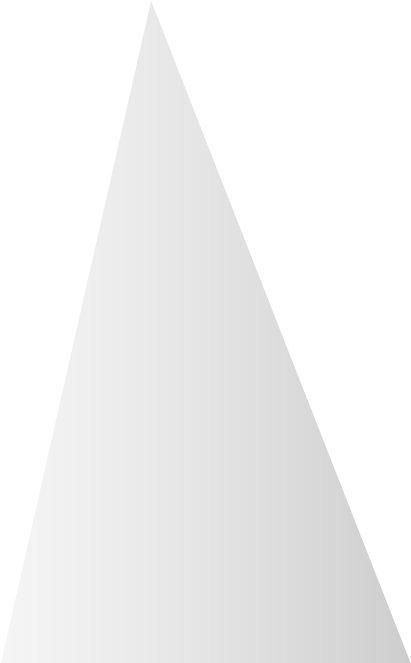
Je suis convaincu que vous continuerez à exceller dans vos projets futurs et à poursuivre vos aspirations académiques et professionnelles avec la même passion et la même détermination. Je vous souhaite tout le succès que vous méritez dans vos projets futurs.

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.**