Exercices sur les Chaînes de Caractères en C

Challenge 1 : Saisie et Affichage Simple

Créez un programme qui lit une chaîne saisie par l'utilisateur et l'affiche.

Fonction recommandée : `fgets()` pour une saisie sécurisée

Objectif : Maîtriser la saisie et l'affichage de chaînes de caractères.

Challenge 2 : Calculateur de Longueur (Sans strlen)

Développez un programme qui calcule la longueur d'une chaîne sans utiliser `strlen()`.

Méthode : Parcourir caractère par caractère jusqu'au `'\0'`

Objectif : Comprendre la structure interne des chaînes de caractères.

Challenge 3 : Concaténation de Chaînes

Créez un programme qui combine deux chaînes en une seule.

Approches :

- Manuelle : Copier la première, puis ajouter la seconde
- Avec `strcat()` pour comparaison

Objectif : Manipuler et assembler des chaînes de caractères.

Challenge 4 : Comparateur de Chaînes

Développez un programme qui vérifie si deux chaînes sont identiques.

Méthode : Comparaison caractère par caractère ou utilisation de `strcmp()`

Objectif : Implémenter des algorithmes de comparaison de chaînes.

Challenge 5 : Inverseur de Chaîne

Créez un programme qui inverse l'ordre des caractères d'une chaîne.

Exemple : "abcd" → "dcba"

Méthode : Échanger les caractères symétriquement (premier avec dernier, etc.)

Objectif : Manipuler les indices et modifier les chaînes en place.

Challenge 6 : Compteur d'Occurrences

Développez un programme qui compte les occurrences d'un caractère dans une chaîne.

Processus :

1. Saisir la chaîne principale
2. Saisir le caractère à rechercher
3. Parcourir et compter les correspondances

Objectif : Implémenter des algorithmes de recherche et de comptage.

Challenge 7 : Conversion en Majuscules

Créez un programme qui convertit tous les caractères en majuscules.

Méthode :

- Utiliser `toupper()` de `<ctype.h>`
- Ou conversion manuelle : `c - 'a' + 'A'` pour les minuscules

Objectif : Manipuler les codes ASCII et transformer les caractères.

Challenge 8 : Conversion en Minuscules

Développez un programme qui convertit tous les caractères en minuscules.

Méthode :

- Utiliser `tolower()` de `<ctype.h>`
- Ou conversion manuelle : `c - 'A' + 'a'` pour les majuscules

Objectif : Renforcer la transformation de caractères.

Challenge 9 : Suppresseur d'Espaces

Créez un programme qui supprime tous les espaces d'une chaîne.

Algorithme :

1. Parcourir la chaîne originale
2. Copier seulement les caractères non-espaces
3. Terminer par `'\0'`

Objectif : Filtrer et réorganiser le contenu des chaînes.

Challenge 10 : Recherche de Sous-Chaîne

Développez un programme qui vérifie si une sous-chaîne existe dans une chaîne principale.

Approches :

- Manuelle : Double boucle pour comparer
- Utiliser `strstr()` pour comparaison

Objectif : Implémenter des algorithmes de recherche de motifs.

Concepts Clés des Chaînes

Représentation en Mémoire

- Tableau de caractères terminé par `'\0'`
- Déclaration : `char str[100]` ou `char str[] = "texte"`

Fonctions Essentielles

#include <string.h>
strlen() // Longueur
strcpy() // Copie
strcat() // Concaténation
strcmp() // Comparaison
strstr() // Recherche de sous-chaîne

Saisie Sécurisée

char buffer[100];
fgets(buffer, sizeof(buffer), stdin);
// Supprimer le '\n' si nécessaire
buffer[strcspn(buffer, "\n")] = '\0';

Bonnes Pratiques

- Toujours vérifier la taille des buffers
- Initialiser les chaînes avant utilisation
- Gérer le caractère de fin `'\0'`
- Utiliser `fgets()` plutôt que `gets()`

Extensions Possibles

- Validation de formats (email, téléphone)
- Tokenisation de chaînes
- Expressions régulières simples
- Chiffrement/déchiffrement de texte

Challenge 11 : Le Rédacteur de Chaîne "En Place"

Objectif Principal : Créer une fonction de recherche et remplacement qui modifie une chaîne de caractères directement dans le tableau qui la contient, sans allouer de mémoire supplémentaire ou utiliser un second tableau.

La Mission

Vous devez implémenter une fonction rechercher_remplacer avec le prototype suivant :

```
void rechercher_remplacer(char source[], const char cible[], const char remplacement[]);
```

Comportement attendu :

- La fonction doit parcourir la chaîne source.
- Elle doit trouver **toutes** les occurrences de la sous-chaîne cible.
- Pour chaque occurrence trouvée, elle doit la remplacer par la chaîne remplacement.
- Toute la modification doit se faire **"en place"**, c'est-à-dire en modifiant directement le contenu du tableau source.

Contraintes :

1. **Pas de nouveau tableau :** Vous ne pouvez pas déclarer un tableau auxiliaire pour construire le résultat. Toute la magie doit opérer à l'intérieur du tableau source.
2. **Buffer suffisant :** Vous pouvez supposer que le tableau source est initialement déclaré avec une taille suffisante pour contenir la chaîne finale, même si elle s'allonge.

Exemple d'Utilisation

Votre fonction doit fonctionner avec le code main suivant :

```c
#include <stdio.h>

// --- Placez votre fonction rechercher_remplacer() ici ---

int main() {
    // Cas 1: Le remplacement allonge la chaîne
    char phrase1[256] = "Je vois un chat sur le tapis du chat.";
    printf("Original : %s\n", phrase1);
    rechercher_remplacer(phrase1, "chat", "grand lion");
    printf("Modifie  : %s\n\n", phrase1);

    // Cas 2: Le remplacement raccourcit la chaîne
    char phrase2[256] = "Le programmeur programme des programmes.";
    printf("Original : %s\n", phrase2);
    rechercher_remplacer(phrase2, "programme", "code");
    printf("Modifie  : %s\n\n", phrase2);

    // Cas 3: La cible n'est pas trouvée
    char phrase3[256] = "Ceci est un test simple.";
    printf("Original : %s\n", phrase3);
    rechercher_remplacer(phrase3, "difficile", "facile");
    printf("Modifie  : %s\n\n", phrase3);

    return 0;
}
```

Sortie Attendue

Original : Je vois un chat sur le tapis du chat.
Modifie : Je vois un grand lion sur le tapis du grand lion.

Original : Le programmeur programme des programmes.
Modifie : Le codeur code des codes.

Original : Ceci est un test simple.
Modifie : Ceci est un test simple.

Conseil Stratégique : Pour gérer le cas où la chaîne s'allonge, réfléchissez à parcourir et modifier la chaîne source en partant de la **fin** plutôt que du début. Cela vous évitera d'écraser des données non traitées.