

# Programmation séquentielle

## Série 6 - Anagrammes et palindromes

18.10.2022

### Buts

- Utilisation des arguments d'un programme.
- Manipulation de chaînes de caractères.
- Modularisation.

### Introduction

Deux mots sont des anagrammes l'un de l'autre quand ils contiennent les mêmes lettres mais dans un ordre différent. Par exemple, tutut et tuttu sont des anagrammes.

Un mot est un palindrome si il se lit de la même façon de gauche à droite que de droite à gauche. Par exemple, rotor et kayak sont des palindromes.

Dans cette série d'exercice, vous écrirez une fonction qui vérifie si deux mots sont des anagrammes et une fonction qui vérifie si deux mots sont des palindromes.

### Énoncé

#### Anagrammes

Ecrire une fonction

```
void sort(char str[]);
```

qui trie une chaîne de caractères par ordre croissant. Attention, le tri ne doit pas considérer les caractères au delà du caractère `\0` qui indique une fin de chaîne de caractère. Vous pouvez utiliser l'algorithme du tri par sélection.

Utilisez cette fonction pour écrire une fonction

```
bool isanagram(char str1[], char str2[]);
```

qui teste si deux chaînes de caractères sont des anagrammes et retourne vrai si c'est le cas, faux sinon. Elle utilisera le principe suivant :

```
tri(mot1);
tri(mot2);
if egalite(mot1, mot2) {
    // anagrammes
} else {
    // pas anagrammes
}
```

## Palindrome

Ecrire une fonction

```
void revert(char str1[], char str2[]);
```

qui recopie la chaîne `str1` dans `str2` en inversant l'ordre des lettres. Il faut que `str2` soit assez grand pour contenir tous les caractères ainsi que le caractère de fin de chaîne `\0`.

Utilisez cette fonction pour écrire une fonction

```
bool ispalindrome(char str[]);
```

qui teste si une chaîne de caractères est un palindrome et retourne vrai si c'est le cas, faux sinon. Elle fonctionnera selon le principe suivant :

```
mot_tmp = chaine de taille de mot
revert(mot, mot_tmp)
retourn egalite(mot, mot_tmp)
```

Vous devrez utiliser la fonction `strlen` pour déterminer la taille d'une chaîne de caractère et `strcmp` pour comparer deux chaînes de caractère.

## Programme principal

Avec les fonctions que vous avez développées, écrivez un programme principal nommé `teststr` qui permet de tester si deux chaînes sont des anagrammes ou si une chaîne est un palindrome. Il fonctionnera de la façon suivante :

```
$ ./teststr 1 tutut tuttu
tutut et tuttu sont des anagrammes
$ ./teststr 1 salut salu
salut et salu ne sont pas des anagrammes
$ ./teststr 2 salut
salut n'est pas un palindrome
$ ./teststr 2 tuttu
tuttu est un palindrome
$ ./teststr 3
```

```
erreur, utilisation :  
./teststr 1 mot1 mot2  
ou  
./teststr 2 mot  
$
```

Vous devez convertir le premier argument en nombre entier pour déterminer quelle opération effectuer. Le programme principal retournera `EXIT_SUCCESS` en cas de succès ou `EXIT_FAILURE` en cas d'échec.

## Partie bonus

Cette partie n'est pas obligatoire. Vous pouvez la faire si vous souhaitez aller plus loin.

### Variante pour le palindrome

Ecrivez une variante de votre fonction `ispalindrome` qui fonctionne selon le principe suivant :

```
while (first_index < last_index {  
    if (mot[first_index] != mot [last_index]) {  
        return false;  
    }  
    first_index +=  
    last_index -=  
}  
return true
```