# Programmation Impérative – TP 2
# Tableaux, chaînes de caractères et pointeurs

**Responsable**
Philippe SWARTVAGHER
philippe.swartvagher@enseirb-matmeca.fr

**Intervenants**
Fadwa ABAKARIM
fadwa.abakarim@enseirb-matmeca.fr
Guillaume MERCIER
mercier@enseirb-matmeca.fr

2025 – 2026

Pour les exercices demandant d'écrire une fonction, on écrira également une fonction `main()`, pour s'assurer que le code écrit compile et que la fonction a le comportement attendu.

Sauf si c'est l'objectif explicite de la fonction, une fonction n'est pas censée *afficher* (par exemple, faire un appel à `printf()`) le résultat du traitement de la fonction. Si je fais appel à une fonction `multiplication()`, je veux récupérer le résultat de la multiplication comme valeur de retour de la fonction, je ne veux pas que `multiplication()` fasse elle-même un appel à `printf()` pour afficher le résultat.

*Une correction sera disponible en ligne une fois que tous les groupes ont réalisé le TP.*

## 1 Échange de valeurs

Écrire une fonction `swap` qui échange les valeurs de deux variables de type `int` passées en paramètres de la fonction.

Dessinez également ce qui se passe en mémoire au fur et à mesure de l'exécution du programme.

## 2 Retour sur la division euclidienne

Reprendre l'exercice sur la division euclidienne du TP précédent.

Écrire une fonction `division` qui à partir de deux entiers $a$ et $b$ permet d'obtenir le quotient et le reste de la division euclidienne de $a$ par $b$. On n'utilisera pas de structure.

# 3 Produit scalaire

Écrire une fonction `produit_scalaire` qui prend en paramètre deux tableaux de `int` de même taille et renvoie leur produit scalaire ($\sum_i x_i y_i$).

Les valeurs du tableau seront définies directement dans le code, et pas récupérées comme paramètres du programme.

# 4 Statistiques sur un tableau

Écrire une fonction `statistiques` qui prend en paramètre un tableau de `int` et permet d'obtenir le minimum, le maximum, la somme et la moyenne des valeurs dans le tableau. Le minimum, le maximum et la somme sont de type `int` et la moyenne de type `float`.

Les valeurs du tableau seront définies directement dans le code, et pas récupérées comme paramètres du programme.

On n'utilisera pas de structure.

Comment faire pour utiliser cette fonction sans avoir à fournir une variable pour chaque statistique (si on n'est intéressé que par le maximum, par exemple) ?

# 5 Palindrome

Écrire un programme qui détermine si la chaîne de caractères passée comme paramètre est un palindrome.

Exemples d'exécutions :

```
./palindrome kayak
'kayak' est un palindrome
./palindrome kayaks
'kayaks' n'est pas un palindrome
```

# 6 Code de César

Le chiffrement par décalage (aussi appelé code de César), consiste à décaler les lettres d'un texte d'un nombre fixe (qu'on appellera la *clé*). Par exemple, si la clé vaut 3, tous les `A` vont devenir des `D`, tous les `B` vont devenir des `E`, ... et tous les `Z` vont devenir des `C`.

Écrire un programme `code_cesar` qui prend comme arguments la clé et une chaîne de caractères à chiffrer et affiche la chaîne une fois chiffrée. On chiffrera uniquement les lettres. Tout autre caractère sera laissé intact.

Écrire ensuite un programme `decode_cesar` qui permet de déchiffrer une chaîne de caractères.

Exemples d'exécutions :

```
./code_cesar 3 "Vive le langage C !"
Ylyh oh odqjdjh F !
./decode_cesar 3 "Ylyh oh odqjdjh F !"
Vive le langage C !
./code_cesar 52 "Vive le langage C !"
Vive le langage C !
./decode_cesar 52 "Vive le langage C !"
Vive le langage C !
```

On affichera directement le résultat chiffré, sans passer par une chaîne de caractères qui stocke la chaîne chiffrée.

## 7 Implémenter la fonction `strchr`

Écrire une fonction `my_strchr`, qui, comme `strchr`, renvoie un pointeur sur la première occurrence du caractère `c` dans la chaîne `s` ; `c` et `s` étant des paramètres de cette fonction. La fonction renvoie `NULL` si le caractère n'est pas présent dans la chaîne.

## 8 Implémenter la fonction `strcpy`

Écrire une fonction `my_strcpy`, qui, comme `strcpy`, copie une chaîne de caractères vers une autre.

## 9 Implémenter la fonction `strcat`

Écrire une fonction `my_strcat`, qui, comme `strcat`, concatène une chaîne de caractère avec une autre.

## 10 Suppression des occurrences d'un caractère

Écrire une fonction `remove_char` qui supprime d'une chaîne de caractères toutes les occurrences du caractère passé en paramètre.

## 11 Tri d'un tableau

Écrire une fonction `sort`, qui trie un tableau sans utiliser de tableau supplémentaire : après l'appel à `sort()`, le tableau passé en paramètre est trié.

On pourra implémenter le tri par sélection. Pourquoi cet algorithme n'est pas efficace ?

## 12 Nombres d'occurrences de lettres

Écrire un programme qui affiche le nombre de lettres dans la chaîne de caractères passée en paramètre. On ne distinguera pas les majuscules des minuscules et on ignorera tous les autres caractères possibles.

Exemples d'exécutions :

```
./lettres kayak
a: 2
k: 2
y: 1
./lettres "22 ! v'la les flics !"
a: 1
c: 1
e: 1
f: 1
i: 1
l: 3
s: 2
v: 1
```

## 13 Implémenter la fonction `atoi`

Écrire une fonction `my_atoi`, qui, comme `atoi`, convertit un nombre dans une chaîne de caractères en nombre de type `int`.

**Bonus 1** Gérer les nombres négatifs.

**Bonus 2** Écrire la fonction `my_atof` qui fonctionne pour les nombres flottants.

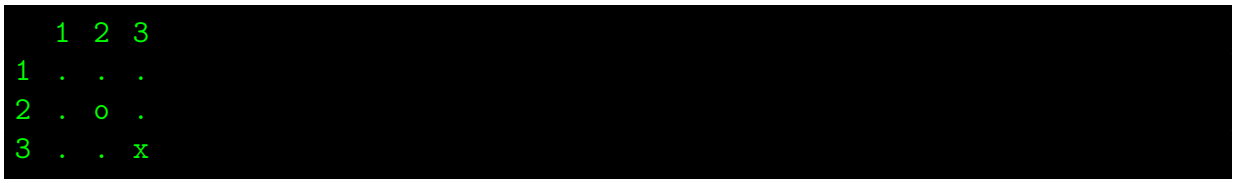
## 14 Ajout de la médiane aux statistiques

Maintenant qu'on a une fonction qui trie les tableaux, compléter la fonction `statistiques` pour qu'elle donne aussi la médiane des valeurs dans le tableau.

**Bonus** Comment faire pour ne pas modifier le tableau passé en paramètre ?

## 15 Bonus : morpion

Faire un programme qui permet à deux joueurs de jouer au morpion. La boucle de jeu consistera à afficher l'état du plateau, par exemple :




puis demander au joueur dont c'est le tour où il souhaite placer un de ses pions. La partie est terminée lorsqu'un joueur parvient à aligner trois pions lui appartenant (auquel cas il a gagné) ou bien si tout le plateau est occupé sans que trois pions appartenant au même joueur ne soient alignés (auquel cas aucun joueur n'a gagné).

**Bonus** Faire en sorte qu'un joueur seul puisse jouer contre l'ordinateur. L'ordinateur pourra jouer selon différentes stratégies, par exemple (du plus simple au plus complexe) : poser son pion sur la première case vide, poser son pion sur une case vide choisie aléatoirement, poser son pion de façon à empêcher l'adversaire de gagner, ...

**Variante** De la même façon, on pourra implémenter un Puissance 4.

*Terminé ?*

*Essayez d'implémenter en C les programmes nécessaires pour résoudre les défis de l'Advent of Code, dont certaines annales se trouvent à l'adresse* `https://gitlab.com/phsw/adventofcode`.