

## Série II

BONNAZ Aymeric

3 août 2025

Cette série d'exercices appartient au corpus testant vos connaissances et votre compréhension des concepts étudiés lors de l'initiation **C# INTM Strasbourg**.  
Les réponses écrites doivent être données sous leurs questions.

Nom apprenant : \_\_\_\_\_

Date de réalisation : \_\_\_\_\_

**Exercice I — Atelier autour des tableaux**

Afin d'apprendre à manipuler les tableaux, cet exercice a pour objectif de développer des solutions à des problèmes simples.

**Somme :**

La première de ces fonctionnalités concerne le calcul de la somme des éléments d'un tableau d'entiers. La signature de la méthode obtenue est la suivante :

```
int : SumTab(int[] tab);
```

Si le résultat est indéfini, on renvoie la valeur -1.

1. (2 points) Implémenter la méthode **SumTab**.

```
Somme des éléments d'un tableau :  
tab : [-1, 4, 7, 12, -6, 5]  
somme : 21
```

FIGURE 1 – Somme de tous les éléments d'un tableau

**Opération sur un tableau :**

La seconde de ces fonctionnalités concerne les trois opérations élémentaires suivantes : l'addition, la soustraction et la multiplication. L'objectif est de les appliquer sur chacun des éléments d'un tableau. Celles-ci sont associées aux opérateurs  $\{+, -, *\}$ . Prenons **tab** un tableau d'entiers, **b** un entier et un opérateur **ope** de la liste précédente.

**b** est le second opérande de l'opération élémentaire appliquée à chaque élément.

La signature de la méthode obtenue est la suivante : 

```
int[] : OpeTab(int[] tab, char ope, int b);
```

Si l'opérateur n'existe pas ou que le tableau ne contient pas d'éléments, on renvoie un tableau vide.

2. (2 points) Implémenter la méthode **OpeTab**.

```
Opération sur un tableau :  
tab : [-1, 4, 7, 12, -6, 5]  
ope : * 2  
res : [-2, 8, 14, 24, -12, 10, -4, 16]
```

FIGURE 2 – Multiplication par 2 des éléments d'un tableau

**Concaténation :**

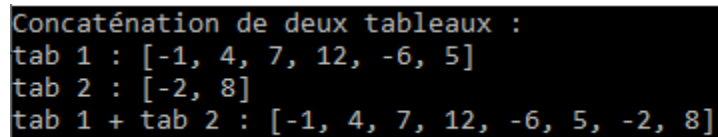
Pour finir, il est intéressant de mettre en place la concaténation de deux tableaux.

La signature de la méthode obtenue est la suivante :

```
int[] : ConcatTab(int[] tab1, int[] tab2);
```

Le résultat attendu est le tableau issu de la concaténation des deux tableaux. Les tableaux vides sont acceptés comme arguments de la fonction.

3. (1 point) Implémenter la méthode **ConcatTab**.



```
Concaténation de deux tableaux :  
tab 1 : [-1, 4, 7, 12, -6, 5]  
tab 2 : [-2, 8]  
tab 1 + tab 2 : [-1, 4, 7, 12, -6, 5, -2, 8]
```

FIGURE 3 – Concaténation de deux tableaux

---

**Exercice II — Morpion**

Le **morpion** se joue à deux joueurs au tour par tour et consiste pour chacun des participants à former le premier un alignement sur une grille. Cet exercice se concentre sur la version classique à 9 cases et a pour objectif de déterminer le vainqueur d'une partie à partir d'une grille préalablement remplie.

Quelques précisions attenantes au déroulement d'une partie :

- Au début, la grille est vide.
- Le joueur 1 dispose des **X**, le joueur 2 des **O**.
- A chaque tour, un joueur trace son symbole dans l'une des cases vides restantes.
- Lorsqu'un alignement est réalisé par l'un des joueurs ou que la grille est pleine, la partie se termine.

**Partie de Morpion**

X	O	X
	X	X
O	O	O

FIGURE 4 – Exemple de partie de Morpion, le joueur 2 (**O**) remporte la partie.

1. (1 point) Pour commencer, citez plusieurs structures de données permettant de représenter la grille du morpion ? Justifiez-votre choix.

---

---

---

**Affichage de la grille :**

La première étape de cet exercice consiste en l'affichage d'une grille de **morpion**. Chaque case de la grille peut contenir, soit l'un des deux symboles **X** ou **O**, ou **\_** pour représenter une case vide. Chaque case est séparée d'un espace pour favoriser la lecture de l'état de la grille.

```
Affichage grille de Morpion :  
X O X  
  X X  
O O O
```

FIGURE 5 – Affichage d'une grille.

2. (2 points) Implémenter la méthode `void : DisplayMorpion(typeGrille grille);`.

### Vérification de la grille :

Le paragraphe suivant présente les contraintes pour la vérification de la grille de **morpion**.

Une grille est fournie en entrée et la fonction à développer doit rendre le statut de la partie. Voici les valeurs possibles :

- **1** ou **2**, si le vainqueur est respectivement le premier ou le second joueur.
- **0**, si aucun joueur n'a réussi à produire un alignement sur une grille pleine.
- **-1**, si la partie n'est pas terminée.

Un alignement valide consiste à remplir une colonne, une ligne ou une diagonale de **3** cases avec le même symbole.

3. (3 points) Implémenter la méthode `int : CheckMorpion(typeGrille grille);`.
-

**Exercice III — Recherche d'un élément**

La recherche d'une valeur particulière dans un ensemble de mesures est un problème classique en informatique. Dans le cadre de cet exercice, les entrées sont un tableau d'entiers à une dimension et une valeur recherchée, la sortie est l'indice correspondant à l'emplacement de la valeur dans ce tableau. La figure (6) présente deux résultats de recherche.

Quelques précisions attenantes aux modalités de résolution du problème :

- Pour un tableau de taille N, les indices sont compris entre 0 et N - 1 inclus.
- Si la valeur est non trouvée, l'indice renvoyé est -1.
- Si le tableau cible est vide, l'indice renvoyé est également -1.

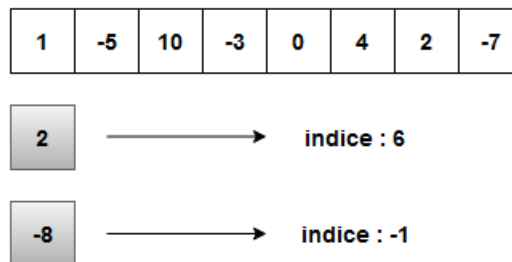


FIGURE 6 – Recherche d'éléments dans le tableau  $A = [1, -5, 10, 3, 0, 4, 2, -7]$ .

**Recherche linéaire :**

La recherche linéaire consiste à lire de gauche à droite les éléments du tableau jusqu'à trouver (ou ne pas trouver) la valeur souhaitée.

1. (2 points) Implémenter la méthode `int : LinearSearch(int[] tableau, int valeur) ;`.
2. (1 point) **Dans le pire cas**, combien d'éléments doivent être lus dans la méthode précédente ?

**Recherche dichotomique :**

Supposons à présent que le tableau d'entrée est **trié**, la recherche dichotomique consiste à comparer la valeur souhaitée avec l'élément du milieu.

- Si cet élément est égal à la valeur, on retourne son indice.
  - Si cet élément est inférieur à la valeur, on recommence avec le sous-tableau de droite.
  - Si cet élément est supérieur à la valeur, on recommence avec le sous-tableau de gauche.
3. (2 points) Implémenter la méthode `int : BinarySearch(int[] tableau, int valeur) ;`.
  4. (1 point) **Dans le pire cas**, combien d'éléments doivent être lus dans la méthode précédente ?
-