

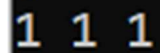
SEQUENCE 5– TABLEAUX
SEANCE 4 – TRIS DE TABLEAUX**OBJECTIFS**

Savoir-faire quelques algos classiques de tris sur un tableau

EXO 1 : LECTURE DE CODE

Lisez ce code et observez le résultat obtenu. Que pouvez-vous en déduire sur la nature des tableaux ? Sont ils des types référents (passés par défaut par référence) ou des types valeurs comme les types primitifs : int, char... (passés par valeur)

```
int[ ] tab = new int[3] {0,0,0};  
Program.PlusUn(tab);  
Console.WriteLine($"{tab[0]} {tab[1]} {tab[2]}");
```



```
private static void PlusUn(int[] tableau)  
{  
    for (int i = 0; i < tableau.Length; i++)  
        tableau[i] = tableau[i] + 1;  
}
```

Conclusion : un tableau est un type référent. Lorsqu'un tableau est passé en paramètre, son contenu n'est pas recopié : c'est sa référence mémoire qui est recopiée.

On peut modifier le contenu d'un tableau passé en paramètre dans une méthode (sans mettre de mot clef ref !)

Par contre, si on veut modifier le tableau dans sa globalité en recréant une autre zone mémoire, il faut utiliser le mot clef ref. Exemple : si on veut ajouter une case, il faut refaire un autre tableau plus grand et tout recopier. Remarque : c'est assez rare car dans ce cas, on préférera l'utilisation de listes.

EXO 2 : TRIS

1. Au sein du répertoire « **Sequence_5_Tableaux** », créez un projet « **Exo_AlgosTris** » type console dans une solution, nommée « **Seance4_AlgosTris** ».
2. Copiez-collez le code suivant :

```
class Program
{
    static void Main(string[] args)
    {
        int[] tab = new int[] {5, 10, 4, 2, 8, 9 };
        Program.Affiche(tab);
        // ICI, programmez le tri bulle
        // ...

        Program.Affiche(tab);
    }

    public static void Affiche(int[] tab)
    {
        for (int i = 0; i < tab.Length; i++)
            Console.Write(tab[i] + " ");
        Console.WriteLine();
    }
}
```

3. Complétez le code ci-dessus afin de faire un tri bulle expliqué ci-après. Pour cela, décomposez en suivant les étapes suivantes :
 - a) Commencez au sein du main par la 1ere boucle (Etape 1) : faites remonter la plus grande valeur dans la dernière case. Testez, affichez.
 - b) Puis faites l'autre boucle qui répète le processus jusqu'à ce que le tableau soit trié.
 - c) Transformez votre code en méthode TriBulle pour délocaliser le code du main.
 - d) Testez à nouveau

Le tri bulle :

Le principe du tri bulle est de faire remonter les plus « grosses bulles ». Ainsi les éléments les plus grands sont remontés. L'algorithme s'arrête dès lors qu'il n'y a plus eu de permutation.

Ex : tableau de nombres « 5 10 4 2 8 9 » ; pour chaque étape, les éléments comparés sont en gras.

Étape 1.

- 1.1. (5 10 4 2 8 9) (5 10 4 2 8 9) Pas d'échange, car $5 < 10$.
- 1.2. (5 10 4 2 8 9) (5 4 10 2 8 9). Échange, car $10 > 4$.
- 1.3. (5 4 10 2 8 9) (5 4 2 10 8 9). Échange, car $10 > 2$.
- 1.4. (5 4 2 10 8 9) (5 4 2 8 10 9). Échange, car $10 > 8$.
- 1.5. (5 4 2 8 10 9) (5 4 2 8 9 10). Échange, car $10 > 9$.

À la fin de cette étape, un nombre est à sa place définitive, le plus grand : 10.

Étape 2.

- 2.1. (5 4 2 8 9 10) (4 5 2 8 9 10). Échange, car $5 > 4$.
- 2.2. (4 5 2 8 9 10) (4 2 5 8 9 10). Échange, car $5 > 2$.
- 2.3. (4 2 5 8 9 10) (4 2 5 8 9 10). Pas d'échange.
- 2.4. (4 2 5 8 9 10) (4 2 5 8 9 10). Pas d'échange.

9 et 10 ne sont pas comparés puisqu'on sait que le 10 est déjà à sa place définitive.

Par hasard, tous les nombres sont déjà triés, mais cela n'est pas encore détecté par l'algorithme.

Étape 3.

- 3.1. (4 2 5 8 9 10) (2 4 5 8 9 10). Échange, car $4 > 2$.
- 3.2. (2 4 5 8 9 10) (2 4 5 8 9 10). Pas d'échange.
- 3.3. (2 4 5 8 9 10) (2 4 5 8 9 10). Pas d'échange.

Les deux derniers nombres sont exclus des comparaisons, puisqu'on sait qu'ils sont déjà à leur place définitive.

Étape 4.

- 4.1. (2 4 5 8 9 10) Pas d'échange

Puisqu'il n'y a eu aucun échange durant cette étape 4, le tri optimisé se termine.

ou encore : https://www.youtube.com/watch?v=MtcrEhrt_K0
ou : <https://www.youtube.com/watch?v=lv3vgjM8Pv4>

4. Complétez le main afin de faire une méthode **triSelection**, tri expliqué ci-après.

Le tri par sélection :

Le principe du tri par sélection/échange (ou *tri par extraction*) est de parcourir le tableau pour trouver l'emplacement de la valeur minimale puis d'échanger cette valeur avec la 1^{er} case, puis de repartir du second élément et d'aller chercher le plus petit élément du tableau pour le mettre en second, etc...

Étape 1.

- 1.1. (**5** 9 1 4 8) par défaut, min est 5 à la case 0 puis compare 9 à 5 : le min reste en case 0
- 1.2. (**5** 9 **1** 4 8) compare 1 à 5 : le min est en case 2
- 1.3. (5 9 **1** **4** 8) compare 4 à 1 : le min reste en case 2
- 1.4. (5 9 **1** 4 **8**) compare 8 à 1 : le min reste en case 2
- 1.5. (1 9 5 4 8) échange la valeur contenue dans la case 0 avec la case contenant le min

Etape 2.

- 2.1 (1 **9** 5 4 8) par défaut min est 9 à la case 1 puis compare 5 à 9 : le min est en case 2
- 2.2 (1 9 **5** 4 8) compare 4 à 5 : le min est en case 3
- 1.1 (1 9 5 **4** 8) compare 8 à 4 : le min reste en case 3
- 2.2 (1 **4** 5 9 8) échange la valeur contenue dans la case 1 avec la case contenant le min

Etape 3.

- 3.1 (1 4 **5** 9 8) par défaut min est 5 à la case 2 puis compare 9 à 5 : le min reste en case 2
- 3.2 (1 4 **5** 9 8) compare 8 à 5 : le min reste en case 2
- 3.2 (1 4 **5** 9 8) pas d'échange

Etape 4.

- 4.1 (1 4 5 **9** 8) par défaut min est 9 à la case 3 puis compare 8 à 9 : le min est en case 4
- 4.2 (1 4 5 **8** **9**) échange la valeur contenue dans la case 3 avec la case contenant le min

TOUS LES TRIS

Observez sur ce site, qu'il existe encore bien d'autres algorithmes de tri Et lancez la démo, vous pourrez constater qu'ils n'ont pas la même efficacité !

<http://lwh.free.fr/pages/algo/tri/tri.htm>

POUR LES PLUS RAPIDES

Choisissez le tri que vous désirez programmer .