Fiche savoirs sur les tableaux en C#

Présentation

En informatiques un **tableau** est un **regroupement** de **variables** de même type mises dans des **cases** contigües en mémoire centrale.
Voici un exemple de tableau de 4 notes en mémoire centrale :

Adresse	Valeur
1500	10
1501	09
1502	15
1503	12

En programmation, un nom est donné au tableau et on utilise un indice plutôt que les adresses mémoire :

Le tableau **notes** :

Indice	Valeur
0	10
1	09
2	15
3	12

Pour accéder à une variable du tableau, il faut :

- utiliser le **nom** du tableau ;
- suivi d'un indice de **case**.

Tableau statique et tableau dynamique

Un **tableau statique** a une taille **fixe** déterminée à la déclaration du tableau.

Un **tableau dynamique** a une taille qui est **définie lors de l'exécution du programme**. La taille du tableau est alors définie au moment de l'exécution du programme et en fonction du déroulement de ce programme. Cela permet d'avoir une **allocation dynamique de mémoire** en fonction des besoins.

Déclarer un tableau statique

Pour déclarer un tableau en C# il faut que

- le nombre de case soit **fixe** ;
- et définir **dès le départ le nombre de cases** ;
- précisez le **type** des données.

```
letype[] nomTableau = new letype[taille];
// exemple
int[] notes = new int[4];
```

Utiliser un tableau statique de données

Un tableau s'utilise comme une variable mais en précisant l'indice de la case concernée.

```
// exemple
notes[0] = 10;
notes[1] = 9;
notes[2] = 15;
notes[3] = 12;
```

Voici une manière de renseigner le tableau de notes avec les valeurs saisies par l'utilisateur :

```
// Déclaration du tableau
int[] notes = new int[4];
```

```
// Saisie des notes
for (int i = 0; i < 4; i++)
{
  Console.Write("Entrez une note  : ");
  notes[i] = int.Parse(Console.ReadLine());
}
```

Pour initialiser un tableau lors de sa déclaration, procédez de la manière suivante :

```
int[] monTableau = new int[] {1, 2, 3, 4, 5};
```

Exercice 1

- Créez un nouveau projet TableauExercice1.
- Le programme doit permettre de saisir 5 températures entières entre -20 et +40 (sans contrôler la saisie) ;
- Le programme doit ensuite afficher combien de températures sont négatives et combien de températires sont positives

Tableau dynamique

Un tableau dynamque en C# est une liste de données qui peut changer de taille pendant l'exécution du programme.

Une liste est une collection ordonnée d'éléments. Chaque élément a un index, et il est possible d'ajouter, d'insérer, de supprimer et de trouver des éléments à n'importe quel endroit dans la liste.

Pour utiliser les listes en C#, il faut inclure le namespace en haut de ton fichier

```
using System.Collections.Generic;
```

Déclaration d'une liste

La déclaration d'une liste en C# est similaire à celle d'un tableau, mais elle utilise la classe List<T>, où T est le type de données que la liste va stocker.

```
List<int> maListe = new List<int>();
```

Dans cet exemple, **<int>** est le type de la liste, **maListe** est le nom de la liste, et new List<int>() est l'instance de la liste.

Il n'est pas nécessaire de spécifier une taille lors de la déclaration d'une liste car les sont dynamiques et peuvent changer de taille.

Par défaut, une liste nouvellement créée ne contient aucun élément.

Pour déclarer et et initialiser une liste en même temps, tout comme un tableau :

```
List<int> maListe = new List<int> {1, 2, 3, 4, 5};
```

Ajouter un élément à la fin de la liste

La méthode **Add()** est utilisée pour ajouter un élément à la fin de la liste.

```
List<int> maListe = new List<int>(); // créer une nouvelle liste vide
maListe.Add(1); // ajouter 1 à la liste
maListe.Add(2); // ajouter 2 à la liste
```

Ajouter plusieurs éléments à la fin de la liste

La méthode **AddRange()** prend une collection d'éléments et les ajoute à la fin de la liste.

```
List<int> maListe = new List<int>(); // créer une nouvelle liste vide
maListe.AddRange(new int[] {1, 2, 3, 4, 5}); // ajouter plusieurs éléments à la liste
```

Après cette instruction, maListe contient les valeurs {1, 2, 3, 4, 5}.

Insérer un élément à une position spécifique

Pour insérer un élément à une position spécifique de la liste, plutôt qu'à la fin, on utilise la méthode **Insert()** qui prend deux paramètres :

- l'index à laquelle l'élément doit être inséré
- et l'élément à insérer :

```
List<int> maListe = new List<int>() {1, 2, 3, 4, 5}; // créer et initialiser une liste
maListe.Insert(2, 10); // insérer 10 à l'index 2
```

Supprimer un élément de la liste

La méthode **Remove()** permet de supprimer des éléments d'une liste.

- Tutoriel de Microsoft Learn : https://learn.microsoft.com/fr-fr/dotnet/csharp/tour-of-csharp/tutorials/list-collection
- Apprendre C# : https://learn.microsoft.com/fr-fr/shows/csharp-for-beginners/

Exercice 2

- Créez un nouveau projet TableauExercice2.
- Reprenez le code de l'exercice 1 et utilisez un tableau dynamique en lieu et place du tableau statique

Rechercher dans un tableau

Il est parfois nécessaire de trier les tableaux ou de faire des recherches dans les tableaux.

Pour information, voici les vidéos sur 3 solutions possible de tris, en C# :

- Tri par sélection : https://youtu.be/0Z8rYUjFrG4
- Tri par insertion : https://youtu.be/dSN7mtGOeZc
- Tri par bulle : https://youtu.be/kUP8kb9t50Q

Recherche séquentielle dans un tableau non trié

La recherche se fait

- à partir de la première case du tableau,
- en comparant chaque case avec la valeur cherchée,
- et la recherche s'arrête dès que la valeur est trouvée, ou en fin de tableau si la valeur n'est pas présente.

C'est la méthode de recherche la plus simple et qui fonctionne que le tableau soit trié ou non. Cependant, cette méthode est peu performante dans le cas d'un tableau trié.

Le nombre d'itérations n'est pas fixe.

- au minimum : 1 fois (si la valeur se trouve dans la 1re case)
- au maximum : n fois (si la valeur n'est pas trouvée)

Exercice 2

- Créez un nouveau projet TableauExercice2.
- Le programme doit permettre de saisir un tableau de températures de n cases (il faut saisir la taille du tableau au préalable) puis de saisir une valeur à chercher.
- La recherche se fait séquentiellement.
- Un message doit préciser si la valeur a été trouvée ou non.
- Dans le cas où elle a été trouvée, il faut préciser la position dans le vecteur. On ne s'inquiètera pas du problème lié à une saisie erronée au niveau du type d'une variable.

Exercice 3

- Créez une nouveau projet à partir de ce dépôt Git https://github.com/ctecher/TableauExercice3.git
- Le programme doit permettre de saisir le nombre de notes à gérer ;
- Puis vous permettre des saisir les notes en les rangeant dans un tableau à une dimension pouvant contenir le nombre de notes voulus.
- Le programme doit alors calculer la moyenne des notes et l'afficher :
  - **sans faire appel à une fonction** ;
  - puis en faisant **appel à une fonction** qui reçoit en paramètre le tableau de notes et qui retourne la moyenne.

En C#, pour obtenir la taille d'un tableau, il faut utiliser **Length** sur le tableau. Par exemple, si le tableau s'appelle notes, alors notes.Length donne sa taille.

Les tableaux à deux dimensions

- Les tableaux qui ont été présentés étaient à une dimension.
- Il est possible de déclarer des tableaux à plusieurs dimensions.
- Les tableaux à 2 dimensions peuvent aussi être appelés **matrices**.

Déclaration et utilisation d'un tableau à 2 dimensions

```
// Syntaxe de la déclaration
letype[,] nomTableau = new letype[nbColonnes,nbLignes];

// Syntaxe d'accès à une case
nomTableau[indiceColonne, indiceLigne]

// Exemple de déclaration
int[,] distance = new int[10,20];

/* Affichage de la case d'indices (i, j)
* avec i forcément entre 0 et 9
* avec j forcément entre 0 et 19 */
Console.WriteLine(distance[i, j]);

// boucles pour remplir chaque case de la matrice
for (int x = 0; x < 10; x++)
{
  for (int y = 0; y < 20; y++)
  {
    distance[x, y] = int.Parse(Console.ReadLine());
  }
}
```

Pour parcourir une matrice complète, il faut 2 boucles imbriquées.