C#

Tableaux et collections

DANS **LA PLUPART** DES LOGICIELS,
ON MANIPULE DES DONNÉES EN
NOMBRE IMPORTANT
PAR EXEMPLE, POUR FAIRE DES
TRAITEMENTS DE MASSE

LES **TABLEAUX** ET LES
COLLECTIONS SONT ALORS
DES **OUTILS INDISPENSABLES**

ON RESSENT LE BESOIN D'UTILISER DES TABLEAUX OU DES COLLECTIONS, LORSQUE L'ON COMMENCE À NOMMER CES VARIABLES AINSI :

`Contact1, Contact2, Contact3, …`

DÉVELOPPER AINSI VOUS CONDAMNE AUX TRAITEMENTS LIMITÉS, AUX TRAITEMENTS UNITAIRES OU PEU OPTIMISÉS.

QU'EST-CE QU'UN TABLEAU ?

Un tableau est une variable capable de contenir plusieurs valeurs de même type

Dans le cas des tableaux à une dimension, on distingue deux choses importantes :

L'**indice**, aussi nommé **index** ou **position**	La **valeur**

COMMENT CRÉER UN TABLEAU ?

- L'EXTRAIT DE CODE SUIVANT MONTRE COMMENT CRÉER UN TABLEAU DE TAILLE « 3 », CONTENANT DES NOMS (DES CONTACTS).
- REMARQUEZ QUE LE PREMIER CONTACT EST LE NUMÉRO 0. LE DERNIER EST DONC LE NUMÉRO 2.
- LES TABLEAUX SONT TRÈS PRATIQUES À UTILISER DANS LES BOUCLES.

Les crochets indiquent qu'il s'agit d'un tableau

Spécifie la taille maximale du tableau : 3 éléments

```
string[] contacts = new string[3];

contacts[0] = "Jean";
contacts[1] = "Michel";
contacts[2] = "Jean-Michel";

MessageBox.Show("Les contacts sont : " + string.Join(", ", contacts));
```

Affecte les valeurs au tableau

Met bout à bout les valeurs du tableau

Indice	Valeur
0	Jean
1	Michel
2	Jean-Michel

```
// La propriété "Length" donne la "taille" du tableau
MessageBox.Show("Nombre de contact : " + contacts.Length.ToString());

contacts[4] = "Daniel"; // Ne fonctionne pas à cause de la taille de trois

Array.Resize(ref contacts, contacts.Length + 1); // Augmente la taille de un

contacts[4] = "Daniel"; // Fonctionne, maintenant

Array.Sort(contacts); // Trie le tableau

Array.Clear(contacts, 0, contacts.Length); // Purge le tableau
```

QUELQUES ASTUCES À CONNAITRE SUR LES TABLEAUX

EN C# IL EXISTE AUSSI LES **COLLECTIONS**, QUE L'ON APPELLE LES **« `List` »**

ELLES SONT UN PEU MOINS PERFORMANTES, ET ELLES PRENNENT PLUS DE MÉMOIRE :

MAIS ELLES SONT UN PEU PLUS SOUPLES : LEUR TAILLE N'A PAS BESOIN D'ÊTRE FIXÉE À L'AVANCE

```
List<string> contacts = new List<string>(); // Nul besoin de préciser de taille

contacts.Add("Jean"); // Pour ajouter des valeurs : on utilise la méthode Add()
contacts.Add("Michel");
contacts.Add("Jean-Michel");

MessageBox.Show("Les contacts sont : " + string.Join(", ", contacts));

// La propriété "Count" donne la "taille" de la collection
MessageBox.Show("Nombre de contact : " + contacts.Count.ToString());

contacts.Add("Daniel"); // L'ajout d'un 4ème contact se fait de la même manière
contacts[4] = "Jérôme"; // L'édition d'un contact existant se fait ainsi.

contacts.Sort(); // Trie la collection

contacts.Clear(); // Purge la collection
```

UTILISATION DES COLLECTIONS DANS LE CODE

```
string[,] tableauContacts = new string[3,3];
List<List<string>> collectionContacts = new List<List<string>>();

tableauContacts[0, 0] = "Michel";
tableauContacts[0, 1] = "Durand";
tableauContacts[0, 2] = "michel.durand@gmail.com";

collectionContacts.Add(new List<string>());
collectionContacts[0].Add("Michel");
collectionContacts[0].Add("Durand");
collectionContacts[0].Add("michel.durand@gmail.com");
```

POUR GÉRER TOUS LES CAS :

IL EST POSSIBLE DE DÉCLARER DES TABLEAUX À PLUSIEURS DIMENSIONS,

ET DES COLLECTIONS DE COLLECTIONS