CENTRE DE RÉADAPTATION MULHOUSE

Rééducation et Formation Professionnelle

Les tableaux statiques en C#

Apprendre à initialiser, parcourir et modifier un tableau statique

CONTENU

Les tableaux à une dimension 1

Introduction 1

Parcourir un tableau à une dimension 1

Rappel : Les boucles For 1

La boucle for pour parcourir un tableau 2

Les tableaux à plusieurs dimensions 3

Jagged array 4

Sources 5

PBourgeois@arfp.asso.fr

LES TABLEAUX A UNE DIMENSION

INTRODUCTION

Les tableaux sont des structures de données qui permettent de stocker plusieurs variables de même type. Il est obligatoire de spécifier le type des éléments d'un tableau et cela se fait dès la déclaration de votre tableau. Par exemple, si je désire créer un tableau qui contiendra des nombres entiers, je vais le déclarer de la manière suivante :

```
int[] monTableau ;
```

Ces tableaux sont dits statiques. Cela signifie qu'ils auront une capacité de stockage limitée qu'il faudra définir à l'initialisation du tableau. Il existe plusieurs façons d'initialiser un tableau :

```
// Déclaration d'un tableau à une dimension pouvant contenir 5 nombres entiers
int[] monTableau = new int[5] ;

// Déclaration et ajout de 5 valeurs
int[]  secondTableau = new int[] {0, 27, 30, 16, 4} ;

// Syntaxe alternative plus rapide à écrire
int[]  troisiemeTableau = {5, 4, 3, 1, 0} ;
```

Les éléments mis dans un tableau occupent une place qui leur est attribuée. Ces places sont limitées par la taille de votre tableau et elles sont caractérisées par un indice qui vous permet d'y accéder. Les indices d'un tableau en C# commencent à une valeur de 0 et vont jusqu'à la taille maximale de votre tableau – 1. Ces indices permettent donc de récupérer la valeur d'un élément d'un tableau mais aussi de modifier cette valeur.

```
// Récupérer la valeur du second élément de secondTableau et l'afficher
Console.WriteLine($"La deuxième valeur de secondTableau est : {secondTableau[1]}") ;

// Changer la valeur du dernier élément de troisiemeTableau par 14
troisiemeTableau[troisiemeTableau.Length - 1] = 14;
```

PARCOURIR UN TABLEAU A UNE DIMENSION

Parcourir un tableau est chose aisée lorsque l'on a bien compris le fonctionnement des boucles For.

RAPPEL : LES BOUCLES FOR

Une boucle est un bloc d'instructions qui va se répéter plusieurs fois selon une condition que l'on donnera. Il existe plusieurs formes de boucles. Dans le cas d'utilisation de tableaux, on utilise le plus souvent les boucles for.

L'instruction for est composée de trois partie :

- L'initialiseur : `int index = 0` qui déclare qu'index est la variable de boucle et ici on l'initialise à 0.
- La condition : `index < 10` tant que cette condition est vraie, les instructions à l'intérieur de la boucle vont se répéter.
- L'itérateur : `index++` montre comment est modifié la variable de boucle à chaque fois que toutes les instructions à l'intérieur de la boucle sont exécutées.

```
// La boucle suivante affichera 12 8 4 0 -4 sur des lignes différentes
for (int i = 12 ; i > -7 ; i = i - 4)
{
   Console.WriteLine(i) ;
}
```

Les points-virgules (;) sont OBLIGATOIRES dans la déclaration des boucles for pour bien délimiter l'initialiseur, la condition et l'itérateur.

LA BOUCLE FOR POUR PARCOURIR UN TABLEAU

Si l'on désire parcourir un tableau en utilisant chacun de ses éléments, il suffit de se rappeler de deux choses. L'indice auquel est le premier élément d'un tableau ; en C# c'est l'indice 0. L'indice du dernier élément d'un tableau ; en C# c'est l'indice égale à la taille du tableau – 1 soit monTableau.Length – 1.

```
string[] maFamille = {"Georges", "Stephan", "Henry", "Corentin", "Steeve"} ;
// Afficher les membres de ma famille sur une ligne
for (int i = 0 ; i < maFamille.Length ; i++)
{
   Console.Write(maFamille[i] + " ") ;
}
```

On pourrait également modifier les valeurs de notre tableau :

```
string[] maFamille = {"Georges", "Stephan", "Henry", "Corentin", "Steeve"} ;
// Remplacer les 'e' dans les prénoms de ma famille par des "i" et affiche le résultat
for (int i = 0 ; i < maFamille.Length ; i++)
{
   maFamille[i] = maFamille[i].Replace("e", "i");
   Console.Write(maFamille[i] + " ") ;
}
```

Généralement, un tableau comme maFamille se fera à partir d'une saisie d'un utilisateur. On peut demander à notre utilisateur d'entrer les noms des membres de sa famille espacés par un espace. La chaine de caractères ainsi récupérée peut être découpée à partir des caractères d'espace et stocker les noms séparément dans notre tableau :

```
string[] familleUtilisateur;
Console.Write("Entrez les noms des membres de votre famille séparés par un espace :") ;
familleUtilisateur = Console.ReadLine().Split(' ') ;
for (int i = 0 ; i < familleUtilisateur.Length ; i++)
{
   Console.WriteLine(familleUtilisateur[i]) ;
}
```

On pourrait voir un tableau à une seule dimension comme un tableau dont chaque valeur occupe une colonne dans une ligne :

maFamille contient les éléments Georges, Stephan, Henry, Corentin, Steeve, ...

Mais il est également possible d'ajouter une dimension à un tableau pour avoir des lignes en plus (par exemple une ligne pour ma belle-famille). En ajoutant une troisième dimension, on aurait une notion de profondeur en plus. On peut ainsi ajouter n dimensions à nos tableaux si besoin

LES TABLEAUX A PLUSIEURS DIMENSIONS

Un tableau multidimensionnel se déclare et s'initialise de différentes façons :

```
// Déclaration d'un tableau à 2 dimensions
int[ , ] maMatrice2D = new int[3, 4];

// Déclaration d'un tableau à 3 dimensions avec valeurs
string[ , , ] maMatrice3D = new string[4, 2, 3] {
                                    {{"a", "b", "c"}, {"d", "e", "f"}},
                                    {{"g", "h", "i"}, {"j", "k", "l"}},
                                    {{"m", "n", "o"}, {"p", "q", "r"}},
                                    {{"s", "t", "u"}, {"v", "w", "x"}}
                                };
```

Si on souhaite directement mettre des valeurs dans un tableau multidimensionnel, on peut simplifier l'expression :

```
float[ , ] zelda = {{12.4f, 6.8f}, {0.2f, 14f}};
```

On pourra par la suite boucler sur chaque élément d'un tableau multidimensionnel avec des boucle imbriquées :

```
// En reprenant maMatrice3D précédemment construite :
for (int i = 0 ; i < maMatrice3D.GetLength(0) ; i++)
{
   for (int j = 0 ; j < maMatrice3D.GetLength(1) ; j++)
   {
      for (int mk = 0 ; k < maMatrice3D.GetLength(2) ; k++)
      {
         Console.Write(maMatrice3D[i, j, k] + " ") ;
      }
      Console.Write(Environment.NewLine);
   }
```

```
Console.Write(Environment.NewLine);
}
```

monTableau.GetLength(n) permet de récupérer la taille de la dimension n de mon tableau : 0 pour la taille de la première dimension (le nombre de lignes), 1 pour la seconde dimension (nombre de colonnes), ...

Pour le moment, les tableaux multidimensionnels que nous avons vus sont tous de la même taille. Si un tableau est initialisé avec 4 lignes, 3 colonnes et 2 de profondeur : chacune des 4 lignes aura 3 colonnes et ce sur les 2 espaces de profondeur du tableau.

Mais il est possible de faire en sorte que leur taille varie en utilisant ce qu'on appelle des jagged arrays. Ces tableaux sont en fait des tableaux de tableaux.

JAGGED ARRAY

Un tableau jagged est un tableau en escalier. Vous comprendrez cette dénomination en voyant le résultat du code suivant :

```
// Déclaration d'un tableau de tableau avec 3 lignes
int[][] tableauEnEscalier = new int [3][] ;

// Création des tableaux dans notre tableau
tableauEnEscalier[0] = new int[1] {1} ;
tableauEnEscalier[1] = new int[3] {1, 2, 1} ;
tableauEnEscalier[2] = new int[4] {1, 3, 3, 1} ;

// Parcourir les tableaux pour afficher leur contenu
for (int i = 0 ; i < tableauEnEscalier.Length ; i++)
{
   for (int j = 0 ; j < tableauEnEscalier[i].Length ; i++)
   {
      Console.Write(tableauEnEscalier[i][j] + " ") ;
   }
   Console.Write(Environment.NewLine) ;
}

/* Affichage attendu :
1
1 2 1
1 3 3 1
*/
```

SOURCES

Les tableaux :

Guide microsoft sur les tableaux en C# : contient la majorité de ce cours

La classe Array : comprend de nombreuses méthodes utiles à appliquer sur vos tableaux

Les boucles for :

boucle for : pour un meilleur rappel

boucle foreach Bien que non utilisés dans ce cours, bons à connaître.

--- FIN DU DOCUMENT ---