

C# : tableaux à plusieurs dimensions

- Niveau requis :
débutant, avisé
- Commentaires : *Contexte d'utilisation du sujet du tuto.*

Introduction

<http://msdn.microsoft.com/fr-fr/library/2yd9wwz4.aspx>

<http://msdn.microsoft.com/fr-fr/library/2s05feca.aspx>

Schémas de tableau à plusieurs dimensions

- Allocation d'un tableau de tableau de 5 lignes

```
int [][] tableauDeTableau; // référence sur tableau à deux dimensions
new tabDeTab = new int [5][]; // On alloue 5 lignes, on ne sait pas combien
de cases (ou colonnes)

// pour le "tableau horizontal"

0| | | |
1| | | | | |
2| |
3| | | | |
4| | | // Ici la cinquième ligne, l'indice commence à
zéro
```

- Référencer le nombre de colonne d'une ligne

```
int [][] tableauDeTableau; // référence sur tableau à deux dimensions
new tabDeTab[0] = new int [3]; // La première ligne référence un tableau de
trois entiers

/*****
0 1 2 3 // indice commence à zéro !
0| | | |
etc... ?
*****/
```

- Accès à une case de tableau à deux dimensions :

```
int [][] tableauDeTableau; // référence sur tableau à deux dimensions
new tabDeTab = new int [5][]; // On alloue 5 lignes
tabDeTab [3][2] = 1515;

0 1 2 3
```

```
0|   |   |   |
1|   |   |   |   |   |
2|   |   |1515|   |   |
3|   |   |   |   |   |
4|   |   |   |   |   |
```

Tableaux multidimensionnel et en escalier

Rappels

```
/******  
 * Rappel tableau à une dimension  
******/  
  
// Déclarer un tableau à une dimension  
int[] array1 = new int[5];  
  
// Déclarer et définir les valeurs des éléments du tableau  
int[] array2 = new int[] { 1, 3, 5, 7, 9 };  
  
// Autre syntaxe  
int[] array3 = { 1, 2, 3, 4, 5, 6 };  
  
/******  
 * Tableau à deux dimensions  
******/  
  
// Déclarer un tableau à deux dimensions (appelée aussi matrice)  
int[,] multiDimensionalArray1 = new int[2, 3]; // 2 lignes, 3  
colonnes  
  
// Déclarer et définir les valeurs des éléments du tableau  
int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 } };  
  
/******  
 * Tableau en escalier  
******/  
  
// Déclarer un tableau en escalier (a jagged array)  
int[][] jaggedArray = new int[6][];  
  
// Définir les valeurs de la première matrice dans la structure du  
tableau en escalier  
jaggedArray[0] = new int[4] { 1, 2, 3, 4 };
```

Remplir un tableau de string à deux dimensions et l'afficher

```

/*****
****
 * Les tableaux multidimensionnels sont appelés aussi des matrices
 * ici on montre différente manière d'afficher une matrice implémentée
 * de valeurs dans le programme principal.
****
**/

static void Main(string[] args) // début du programme principal
{
    //tableau multidimensions de char
    string[,] tabString = new string[3, 2] { {"un", "deux"},
                                             {"trois", "quatre"},
                                             {"cinq", "six"} };

    // Autre écriture ne sera pas afficher, présenter pour syntaxe :
    string[,] array2Db = new string[,] { { "one", "two" }, {
"three", "four" },
                                         { "five", "six" } };

Console.WriteLine("//////////////////////////////////////////
//");
    Console.WriteLine("Affichage des éléments des tableaux de string
et de int :");
Console.WriteLine("//////////////////////////////////////////
//");

    Console.WriteLine();
    Console.WriteLine("..... AFFICHAGE n° 1 de TabString
SANS BOUCLE.....");
    Console.WriteLine();
    Console.WriteLine("première ligne, tab[0 ,0] et tab[0, 1] => " +
tabString[0, 0] + " et " + tabString[0, 1]); //un et deux
    Console.WriteLine();
    Console.WriteLine("deuxième ligne, tab[1, 0] et tab[1, 1] => "
+ tabString[1, 0] + " et " + tabString[1, 1]); //trois et quatre
    Console.WriteLine();
    Console.WriteLine("troisième ligne, tab[2, 0] et tab[2, 1] => "
+ tabString[2, 0] + " et " + tabString[2, 1]); //cinq et six
    Console.WriteLine();

    int i, j; //indices vertical et horizontal du tabString
    Console.WriteLine("..... AFFICHAGE n° 2 de TabString AVEC
UNE BOUCLE (tab de string).....");
    for (i = 0; i < 1; i++)
    {
        for (j = 0; j < 2; j++)
        {
            Console.WriteLine(" | "+ tabString[i, j] +" | ");
        }
    }
}

```

```
    }
}
Console.WriteLine();

for ( i = 1 ; i < 2; i++)
{
    for (j = 0; j < 2; j++)
    {
        Console.WriteLine(" | " + tabString[i, j] + " | ");
    }
}
Console.WriteLine();

for (i = 2; i < 3; i++)
{
    for (j = 0; j < 2; j++)
    {
        Console.WriteLine(" | " + tabString[i, j] + " | ");
    }
}
Console.WriteLine();

//Tableau multidimensions de int :
int[,] tabInt = new int[4, 2] { { 1, 2 }, { 3, 4 }, { 5, 6 }, {
7, 8 } };

// autre écriture ne sera pas afficher, présenter pour syntaxe :
int[,] array2D = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7,
8 } };

Console.WriteLine("..... AFFICHAGE n°1 DU TABLEAU DE INT
(avec une boucle).....");

Console.WriteLine(" | "+tabInt[0,0]+ " | " +tabInt[0,1]);
Console.WriteLine(" | " +tabInt[1,0]+ " | " +tabInt[1,1]);
Console.WriteLine(" | " +tabInt[2,0]+ " | " +tabInt[2,1]);
Console.WriteLine(" | " +tabInt[3,0]+ " | " +tabInt[3,1]);
Console.WriteLine();

Console.WriteLine("..... AFFICHAGE n°2 DU TABLEAU DE INT
(avec une boucle).....");
int l, c; //indice vertical et horizontal de tabInt
for (l = 0; l < 4; l++)
{
    for (c = 0; c < 2; c++)
    {
        Console.WriteLine(" | " + tabInt[l, c] + " | ");
    }
}
}
```

```

        System.Console.WriteLine("Appuyer sur une touche pour quitter le
programme.");
        System.Console.ReadKey();
    }

```

Afficher le contenu d'un tableau en escalier

```

class TableauEnEscalier1
{
    static void Main(string[] args)
    {
        int[][] numbers = new int[2][] { new int[] { 1, 2, 3 }, new
int[] {4, 5, 6, 7, 8, 9 } };

        Console.WriteLine("{ " + numbers[0][0] + " , " + numbers[0][1] + " ,
" + numbers[0][2] + " }");
        Console.WriteLine();
        Console.WriteLine("{ " + numbers[1][0] + " , " + numbers[1][1] +
" , " + numbers[1][2] + " , " + numbers[1][3] + " , " + numbers[1][4] + " , " +
numbers[1][5] + " }");
        Console.WriteLine();
    }
}

```

Remplir et afficher une matrice avec une boucle

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

/*****
 * Les tableaux multidimensionnels sont appelés aussi des matrices
 * ici on se sert d'une fonction pour remplir et pour afficher la matrice
 *****/

namespace Structuré
{
    class RemplirEtAfficherMatrice
    {
        public const int L = 10; // nombre max de lignes
        public const int C = 15; // nombre max de colonnes
    }
}

```

```
static void Main(string[] args) // début du programme principal
{
    do // Début du "do ...while" proposant à l'utilisateur de recommencer le programme
    {
        int[,] Ma_matrice;
        // matrice = new int [ L , C ]; // Allocation d'espace en mémoire (pas encore l'affectation des cases : Mais c'est la fonction qui le fait et qui remplit les cases
        Ma_matrice = SaisirMatriceEntiers(); // appel de la fonction qui remplit la matrise "Ma_matrice"

        AfficherMatriceEntiers(Ma_matrice); // appel de la fonction qui permet d'afficher "Ma_matrice"

        Console.WriteLine();
    } while(veutContinuer("Voulez-vous remplir et afficher une autre matrice ? (o/n)"));

    Console.WriteLine("Tapez sur Entree pour Terminer");
    Console.ReadLine(); // fin du programme principal
}

/*****Remarques*****/
* toutes les fonctions ci-dessous pourraient être placées dans un autre fichier
* par exemple, une class Mes fonctions
* dans ce cas, il faudrait placer le nom du main suivi d'un point devant le nom de la fonction lors de son appel
* par exemple class MesFonctions
* appel Ma_matrice = MesFonctions.SaisirMatriceEntiers();
* C'est ce qui est fait pour l'appel de la fonction "Lire.Caractere" utilisée dans les fonctions ci-dessous.*****/

// Fonction qui permet de recommencer le programme
private static bool veutContinuer(string invite)
{
    char cara;
    do
    {
        cara = Char.ToLower(Lire.Caractere(invite));
        if (((cara != 'n') && (cara != 'o')))
            //if (!( (cara == 'n') || (cara == 'o') )) // expression identique selon règle de De Morgan
        {
            Console.WriteLine("Répondez oui ou non (o/n)");
        }
    } while (!((cara == 'o') || (cara == 'n')));
}
```

```
        return ((cara == 'o'));
    }

    //fonction qui permet de remplir une matrice
    private static int[,] SaisirMatriceEntiers()
    {
        int nbLig, nbCol; // dimensions utiles
        int[,] matrice;

        // Saisie de la taille utile de la matrice
        do // Nombre de lignes
        {
            nbLig = Lire.Entier("Indiquez le nombre de lignes de la
matrice (entre 1 et " + L + ") ");
            // Arrêt quand la valeur saisie est acceptable (càd entre 1
et MAX_NB_LIG incluses).
        } while ((nbLig < 1) || (L < nbLig));

        do // Nombre de colonnes
        {
            nbCol = Lire.Entier("Indiquez le nombre de colonnes de la
matrice (entre 1 et " + C + ") ");
            // Arrêt quand la valeur saisie est acceptable (càd entre 1
et MAX_NB_COL incluses).
        } while ((nbCol < 1) || (C < nbCol));

        // Allocation d'espace pour la matrice à saisir :
        matrice = new int[nbLig, nbCol];

        // Saisie de la matrice
        for (int i = 0; i < nbLig; i++) // Pour chaque ligne
        {
            for (int j = 0; j < nbCol; j++) // Pour chaque colonne
dans la ligne
            // Arrêt avant de dépasser la taille max d'une ligne (càd le
nombre de colonnes).
            {
                matrice[i, j] = Lire.Entier("Matrice [" + i + ", " + j +
"] =");
            }
        }
        return matrice;
    }

    // Fonction qui permet d'afficher une matrice :
    private static void AfficherMatriceEntiers(int[,] matrice)
    {
        int nbLig = matrice.GetLength(0); // nb lignes de la matrice
: "GetLength(0)" longueur de la première dimension, la première c'est 0
        int nbCol = matrice.GetLength(1); // nb colonnes :
"GetLength(1)" : GetLength(1) longueur de la deuxième dimension, la deuxième
c'est 1
    }
```

```
        for (int i = 0; i < nbLig; i++) // Pour chaque ligne
        {
            Console.WriteLine(); //saut de ligne avant chaque ligne
de la matrice

            for (int j = 0; j < nbCol; j++) // Pour chaque colonne
dans la ligne
            {
                Console.Write(matrice[i, j] + " | ");
            }
        }
        Console.WriteLine(); //saut de ligne après la matrice
    }
}
```

remplir une matrice et l'afficher inversée

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Structuré
{
    class TransposerMatrice
    {
        public const char RetourAlaLigne = '\u000A'; // caractère de
retour à la ligne
        public const int L = 10; // nombre max de lignes
        public const int C = 15; // nombre max de colonnes

        static void Main(string[] args)
        {
            // Déclarations
            int[,] matrice; // Déclaration d'une matrice (tableau
à 2 dimensions)

            do // jusqu'à ce que l'utilisateur décide de ne pas
continuer...
            {
                // Saisir la matrice d'entiers A
                matrice = SaisirMatriceEntiers();
            }
        }
    }
}
```

```

        // Afficher la matrice saisie
        Console.WriteLine(RetourAlaLigne + "La matrice saisie
est : " + RetourAlaLigne);
        AfficherMatriceEntiers(matrice);

        // Transposition et affichage
        Console.WriteLine(RetourAlaLigne + "La matrice
transposée à partir de la matrice saisie est : " + RetourAlaLigne);
        AfficherMatriceEntiers(TransposeMatrice(matrice));

    } while (veutContinuer("Voulez-vous continuer ? \" oui :
o\"; \"Non : n?\""));

    Console.WriteLine("Tapez entrée pour terminer !");
    Console.ReadLine(); // pour voir le résultat
}
/*****
*****
    * Fonction qui permet de transposer une matrice
    */

    private static int[,] TransposeMatrice(int[,] matrice) //
Pareil que "public static void TransposeMatrice(int[,] matrice, out int[,]
matriceTransposée"
    {
        int nbLig = matrice.GetLength(0); // nb lignes de la
matrice donnée
        int nbCol = matrice.GetLength(1); // nb colonnes
        int[,] matriceT; // matrice transposée

        // Allocation d'espace pour la matrice transposée :
        // les lignes de la matrice donnent les colonnes de sa
transposée
        // les colonnes de la matrice donnent les lignes de sa
transposée
        matriceT = new int[nbCol, nbLig];

        for (int i = 0; i < nbLig; i++) // Pour chaque ligne de la
matrice donnée
        {
            for (int j = 0; j < nbCol; j++) // Pour chaque colonne
dans cette ligne
            {
                matriceT[j, i] = matrice[i, j];
            }
        }
        return matriceT;
    }

/*****
*****

```

```
// Fonction qui permet de recommencer un programme***/

private static bool veutContinuer(String invite)
{
    char cara;
    do
    {
        cara = Char.ToLower(Lire.Caractere(invite));
        //if (!( (cara == 'n') || (cara == 'o') )) // selon De
Morgan

        if (((cara != 'n') && (cara != 'o')))
        {
            Console.WriteLine("Répondez oui ou non (o/n)");
        }

    } while (!( (cara == 'o') || (cara == 'n') ));
    return ((cara == 'o'));
}

/*****
*****
    * Fonction qui permet de remplir une matrice
    */
private static int[,] SaisirMatriceEntiers()
{
    int nbLig, nbCol; // dimensions utiles
    int[,] matrice;

    // Saisie de la taille utile de la matrice
    do // Nombre de lignes
    {
        nbLig = Lire.Entier("Indiquez le nombre de lignes de la
matrice (entre 1 et " + L + ") ");
        // Arrêt quand la valeur saisie est acceptable (càd
entre 1 et MAX_NB_LIG incluses).
    } while ((nbLig < 1) || (L < nbLig));

    do // Nombre de colonnes
    {
        nbCol = Lire.Entier("Indiquez le nombre de colonnes de
la matrice (entre 1 et " + C + ") ");
        // Arrêt quand la valeur saisie est acceptable (càd
entre 1 et MAX_NB_COL incluses).
    } while ((nbCol < 1) || (C < nbCol));

    // Allocation d'espace pour la matrice à saisir :
    matrice = new int[nbLig, nbCol];

    // Saisie de la matrice
```

```

        for (int i = 0; i < nbLig; i++) // Pour chaque ligne
        {
            for (int j = 0; j < nbCol; j++) // Pour chaque colonne
dans la ligne
                // Arrêt avant de dépasser la taille max d'une ligne
                (càd le nombre de colonnes).
                {
                    matrice[i, j] = Lire.Entier("Matrice [" + i + ", " +
j + "] =");
                }
            }
        }
        return matrice;
    }

/*****
*****
    * Fonction qui permet d'afficher une matrice
    */
    private static void AfficherMatriceEntiers(int[,] matrice)
    {
        int nbLig = matrice.GetLength(0); // nb lignes de la
matrice : "GetLength(0)" longueur de la première dimension, la première
c'est 0
        int nbCol = matrice.GetLength(1); // nb colonnes :
"GetLength(1)" : GetLength(1) longueur de la deuxième dimension, la deuxième
c'est 1

        for (int i = 0; i < nbLig; i++) // Pour chaque ligne
        {
            Console.WriteLine(); //saut de ligne avant chaque
ligne de la matrice

            for (int j = 0; j < nbCol; j++) // Pour chaque colonne
dans la ligne
            {
                Console.Write(matrice[i, j] + " | ");
            }
        }
        Console.WriteLine(); //saut de ligne après la matrice
    }
}
}
}

```

Utilisation

Last update: 21/12/2014 20:04 utilisateurs:hyathie:tutos:c-sharp-tableaux_plusieurs-dimension http://debian-facile.org/utilisateurs:hyathie:tutos:c-sharp-tableaux_plusieurs-dimension

From: <http://debian-facile.org/> - Documentation - Wiki

Permanent link: http://debian-facile.org/utilisateurs:hyathie:tutos:c-sharp-tableaux_plusieurs-dimension

Last update: **21/12/2014 20:04**

