[] : Tableau

Définition

Un tableau est une collection d'entitées du même type. Chaque entité est accessible via un index.

Un tableau peut être visualisé comme un coffre qui contient d'autres coffres.

Comment déclarer un tableau

Pour déclarer un tableau, nous utiliserons la syntaxe suivante:

```
type[] nomTableau = new type[taille];
```

2 nouveaux éléments ici :

📌 `[]`

📌 `new`

`taille` correspond au nombre d'éléments dans mon tableau

Exemple de déclaration de tableaux:

```
int[] tableau_int = new int[10]; // tableau de 10 entiers

string[] tableau_string = new string[12]; // tableau de 12 string
```

Remplir un tableau

Pour remplir un tableau, nous utiliserons la syntaxe suivante:

```
nomTableau[index] = valeur;
```

Le premier index de mon tableau est 0.

Exemple de remplissage de tableau:

```
int[] tableau_int = new int[4]; // tableau de 4 entiers

tableau_int[0] = 3; //index = 0, valeur = 3;
tableau_int[1] = 5;
tableau_int[2] = 7;
tableau_int[3] = 9;
```

Initialisation lors de la déclaration

Vous pouvez éviter l'expression `new` et le type de tableau lorsque vous initialisez un tableau lors de la déclaration, comme indiqué dans le code suivant.

```
int[] array2 = { 1, 3, 5, 7, 9 };
string[] weekDays2 = { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" };
```

Exercice :

📌 Créer un tableau de string `mois` pour y mettre tous les mois de l'année et instanciez le.

Solution :

```
string[] mois = {"janvier", "fevrier", "mars", "avril", "mai", "juin", "juillet", "aout", "septembre", "octobre", "novembre", "decembre" };

--- reponce alternative

string[] mois = new string [12];
mois[0] = "janvier";
.
.
.
mois[11] = "decembre";
```

Accéder à la valeur dans le tableau

Pour accéder à la valeur dans le tableau, nous utiliserons la syntaxe suivante:

```
nomTableau[index]
```

!! Le premier index de mon tableau est 0.

Exemple d'accès à la valeur dans un tableau:

```
int[] tableau_int = new int[4]; // tableau de 4 entiers

tableau_int[0] = 3; //index = 0, valeur = 3;
tableau_int[1] = 5;
tableau_int[2] = 7;
tableau_int[3] = 9;

Console.WriteLine(tableau_int[0]); // affichera 3
Console.WriteLine(tableau_int[1]); // affichera 5
Console.WriteLine(tableau_int[2]); // affichera 7
Console.WriteLine(tableau_int[3]); // affichera 9
```

Exercice

Avec l'exercice précédent,
Créer un tableau de string `mois_avec_e`, qui contiendra tous les mois qui possèdent un `e` dans leurs nom.
Celui-ci sera initialiser seulement à l'aide d'index et le tableau de `mois`.

Solution :

```
string[] mois = {"janvier", "fevrier", "mars", "avril", "mai", "juin", "juillet", "aout", "septembre", "octobre", "novembre", "decembre" };

string[] mois_avec_e = new string [7];
mois_avec_e[0] = mois[0];
mois_avec_e[1] = mois[1];
mois_avec_e[2] = mois[6];
mois_avec_e[3] = mois[8];
mois_avec_e[4] = mois[9];
mois_avec_e[5] = mois[10];
mois_avec_e[6] = mois[11];
```

Méthode de base

Un tableau en C# est un object qui possède des centaines de propriétés déjà intégrées.
Pour les tableaux, la principale et la plus utilisée est :

📌 `Length` : retourne la taille du tableau

Principalement utilisée pour parcourir votre tableau.

Exemple

```
int[] tableau_int = new int[4]; // tableau de 4 entiers

tableau_int[0] = 3; // index commence a 0
tableau_int[1] = 5;
tableau_int[2] = 7;
tableau_int[3] = 9;

Console.WriteLine(tableau_int.Length); // affichera 4
```

Logique

Condition

Une condition est une expression qui retourne un `booleen (true ou false)`.

En fonction de la valeur de la condition, un bloc de code sera exécuté (`true`) ou non (`false`).

Tous les opérateurs de comparaison peuvent être utilisé dans les conditions

Opérateur de comparaison	Signification
==	égal à
!=	différent de
>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à

If

La condition la plus simple est le `if` qui permet d'éxécuter un bloc de code si la condition est vraie.

```
if (condition 1)
{
// code a executer si la condition 1 est vrai
}
```

Exercice :

créer deux variable `int` solde_camille et solde_bob
initialiser les tel que :
solde_camille = 1111;
solde_bob = 289;
créer une condition `if` si solde_camille est supérieur au solde_bob et y mettre le code suivant

```
Console.WriteLine("camille a un solde superieur a bob " + solde_camille);
```

Exécuter le code.

Solution :

```
int solde_camille = 1111;
int solde_bob = 289;

if (solde_camille > solde_bob) {

Console.WriteLine("camille a un solde superieur a bob " + solde_camille);
}
```

Else

La condition `else` permet d'exécuter un bloc de code si la condition est fausse.

```
if (condition 1)
{
// code à éxécuter si la condition 1 est vraie
}
else
{
// sinon, code à éxécuter si la condition 1 est fausse
}
```

Exercice :

Reprendre le code précédent et ajouter au scope de else :

```
Console.WriteLine("Bob a un solde superieur a bob " + solde_bob);
```

Exécuter le code.

Changer le solde de bob a 2222;

Exécuter le code.
Quel que chose a-t-il changé dans l'affichage du terminal?

Else if

La condition `else if` permet d'exécuter un bloc de code si la condition est fausse et que l'on souhaite vérifier une autre condition.

```
if (condition 1)
{
// code à éxécuter si la condition 1 est vraie
}
else if (condition 2)
{
// sinon, code à éxécuter si la condition 1 est fausse et que condition 2 est vraie
}
else
{
// code à éxécuter si la condition 1 est fausse et que condition 2 est fausse
}
```

Exemple

```
int a = 5;
int b = 10;

if (a > b)
{
Console.WriteLine("a est supérieur à b");
}
else if (a < b)
{
Console.WriteLine("a est inférieur à b");
}
else
{
Console.WriteLine("a est égal à b");
}
```

Exercice

```
Changer int a = 120;
Exécuter le code.
Changer int a = 120;
Changer int b = 120;
Exécuter le code.
```

Exercice

```
int ton_age = mettre ton age;
int nombre_denfant = mettre ton nombre d enfant;

1 Créer une condition pour vérifier si tu es majeur.
Elle affichera \(avec Console.WriteLine\) si tu es majeur, ou si tu es mineur

2 Créer une ou plusieurs conditions pour vérifier si \:
tu n as pas d enfant,       // afficher je n'est pas d'enfant
3 tu as moins de 3 enfants mais tu as des enfants,   // afficher j'ai nbr d enfant
4 tu as 4 ou moins mais plus que 2,   // j'ai plus que 2 enfants mais ce n'est pas une famille nombreuse
5 tu as plus que 4 enfants, // j'ai une famille nombreuse
```

```
int ton_age = mettre ton age;
int nombre_denfant = mettre ton nombre d enfant;

if (ton_age >= 18)
{
Console.WriteLine("Tu es majeur");
}
else
{
Console.WriteLine("Tu es mineur");
}
if (nombre_denfant == 0) // condition 1
{
Console.WriteLine("Je n'ai pas d'enfant");
}
else if (nombre_denfant < 3) // si condition 1 fausse, condition 2
{
// autre possibilité, nbr_denfant <= 2
Console.WriteLine("J'ai " + nombre_denfant + " enfant");
}
```

Réponse

```
else if (nombre_denfant <= 4) // condition 3
{
Console.WriteLine("J'ai plus que 2 enfants mais ce n'est pas une famille nombreuse");
}
else // condition 4 , si aucune des conditons n'étaient bonne
{
Console.WriteLine("J'ai une famille nombreuse");
}
```

Opérateur Booleen

Les opérateurs booleen permettent de combiner des conditions.

Opérateur	Signification
&&	ET
||	OU
!	NON

ET

L'opérateur ET permet d'exécuter un bloc de code si les deux conditions sont vraies.

```
if (condition 1 && condition 2)
{
// code à executer si les deux conditions sont vraies
}
```

OU

L'opérateur OU permet d'exécuter un bloc de code si une des deux conditions est vraie.

```
if (condition 1 || condition 2)
{
// code à executer si une des deux conditions sont vraies
}
```

NON

L'opérateur NON permet d'inverser la valeur de la condition.

```
if (!condition)
{
// code à executer si la condition est fausse
}
```

Exercice

```
int nbr_pizza = 14;
int nbr_couvert = 30;
int nbr_personne = 12;

1 créer une condition qui verifie :
- tout le monde aura 2 couverts au moins
        ET
- tout le monde aura 1 pizza au moins

2 créer une condition qui verifie :
- tout le monde aura 2 couvert au moins
        OU
- tout le monde aura 1 pizza au moins
```

Priorité des parenthèses

Les opérateurs booleen sont évalués de gauche à droite dans une condition.

Les parenthèses `()` fonctionnent en informatique comme en mathématiques
La priorité de la condition se dirige dans la parenthèse en 1er.

Table de vérité

A	B	A && B
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux

A	B	A || B
Vrai	Vrai	Vrai
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

A	!A
Vrai	Faux
Faux	Vrai

Exercice

Reprendre l'exercice précédent en ajoutant la dernière condition :

```
3 Créer une condition qui vérifie :
- tout le monde aura 2 couverts au moins
        ET
- tout le monde aura 1 pizza au moins
        ET
                - il y a plus de 20 couverts
                        OU
                - il y a plus de 10 personnes
```