# Les rudiments du language c#

Nouhaila Bensalah

Chercheuse en IA/NLP

nouhaila.bensalah@etu.fstm.ac.ma

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# Les variables

- Des blocs de mémoire.
- Permettent de stocker une ou plusieurs données.
- Peuvent avoir plusieurs valeurs différentes dans un programme.
- permettent d'effectuer:
  - Des calculs mathématiques.
  - Enregistrer l'âge du visiteur.
  - Comparer des valeurs

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# Opérations de lecture et écriture

Pour lire une chaîne saisie dans la console et l'enregistrer dans une variable:

```
string n = Console.ReadLine();
```

Pour afficher le contenu d'une variable dans la console:

```
Console.WriteLine("Chaine saisie: {0}", n);
```

Ou:

```
Console.WriteLine($"Chaine saisie: {n}");
```

## Remarques

- string est le type de la valeur saisie et sauvegardée dans la variable n.
- {} fait référence à la première variable située après le texte du message à afficher.
- $ permet de remplacer les variables situées entre {} par leurs valeurs respectives

# Les variables: Types et déclaration

## Déclarer une variable:

*type nomVariable*;

## Les principaux types de base du framework .NET sont:

| Type | Description |
|---|---|
| byte | Entier de 0 à 255 |
| short | Entier de -32768 à 32767 |
| int | Entier de -2147483648 à 2147483647 |
| long | Entier de -9223372036854775808 à 9223372036854775807 |
| float | Nombre simple précision de -3,402823e38 à 3,402823e38 |
| double | Nombre double précision de -1,79769313486232e308 à 1,79769313486232e308 |
| decimal | Nombre décimal convenant particulièrement aux calculs financiers (en raison de ses nombres significatifs après la virgule) |
| char | Représente un caractère |
| string | Une chaine de caractère |
| bool | Une valeur booléenne (vrai ou faux) |

## Déclarer une variable

```
int n;
```

## Déclarer et initialiser une variable

```
int n = 5;
```

## Ceci est?

```
byte n = 300;
Console.WriteLine(n);
```

## Exemples

1. `int x = 5/2;`

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# Opérations sur les variables

## Pour les variables numériques (int, float...)

- = → **affectation**
- + → **addition**
- - → **soustraction**
- * → **multiplication**
- / → **division**
- % → **reste de la division**

# Opérations sur les variables

## Exemples

```
int a = 5; int b = 2;
Console.WriteLine($"a + b");
Console.WriteLine($"a - b");
Console.WriteLine($"a * b");
Console.WriteLine($"a/b");
Console.WriteLine($"(double)a/b");
Console.WriteLine($"a%b");
```

# Opérations sur les variables

## Opérateurs particuliers

$i = i + 1 \rightarrow i + +;$

$i = i - 1 \rightarrow i - -;$

$i = i + 2 \rightarrow i+ = 2;$

$i = i - 2 \rightarrow i- = 2;$

# Opérations sur les variables

## Exemples

```
int i = 2;

int j = i++; //post-incrémentation


int i = 2;

int j = ++i; //pre-incrémentation
```

# Opérations sur les variables

## L'opérateur + pour concaténer les chaînes de caractères

```
string codePostal = "1500";

string ville = "Rabat";

string pays = "Maroc";

string adresse = codePostal + "" + ville + "" + pays;
```

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# Les opérateurs de comparaison

Une condition se construit grâce à des opérateurs de comparaison. Les plus courants sont:

| Opérateur | Description |
|---|---|
| == | Egalité |
| != | Différence |
| > | Supérieur à |
| < | Inférieur à |
| >= | Supérieur ou égal |
| <= | Inférieur ou égal |
| && | ET logique |
| \|\| | OU logique |
| ! | Négation |

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# If

Permet d'exécuter le code (...) si une condition est vraie

```
if (condition)
{
...
}
```

Exemple

```
decimal compteEnBanque = 300;

if (compteEnBanque >= 0)

    Console.WriteLine("Votre compte est créditeur");
```

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# If ... else

```
if (condition)
{...}
else
{...}
```

Exemple

```
decimal compteEnBanque = 300;

if (compteEnBanque >= 0)

    Console.WriteLine("Votre compte est créditeur");

else

    Console.WriteLine("Votre compte est débiteur");
```

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# If ... else if ... else

On peut enchaîner les conditions avec else if

```
if (condition1)
{
...
}
else if (condition2)
{
...
}
...
else
{
...
}
```

# If ... else if ... else

## Exemple

```
if (titre == "Mme")

    Console.WriteLine("Vous êtes une femme");

else if (titre == "Mlle")

    Console.WriteLine("Vous êtes une femme non mariée");

else if (titre == "M.")

    Console.WriteLine("Je n'ai pas pu déterminer votre civilité");

else

    Console.WriteLine("Vous êtes un homme");
```

# If ... else if ... else

## Exercice avec If ... else if ... else

- Écrire un code C# qui retourne le signe du résultat de la multiplication de deux nombres sans calculer leur produits.
- Les deux nombres sont des entiers différents de zéro.

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# switch

L'instruction switch peut être utilisée lorsqu'une variable peut prendre beaucoup de valeurs.

```
switch (expression){

    case x :

        ...

        break;

    case y :

        ...

        break;
    default :

        ...

        break;
}
```

# switch

## Exemple

```
int nombre = 10;
switch (nombre) {
    case 1 :
        Console.WriteLine("Un");
        break;
    case 2 :
        Console.WriteLine("Deux");
        break;
    case 3 :
        Console.WriteLine("Trois");
        break;
    default :
        Console.WriteLine("Autre");
        break;
}
```

## Remarques

- Le bloc défaut peut apparaître à n'importe quelle position dans switch. Quelle que soit sa position, il est toujours évalué en dernier, une fois que tous les blocs case ont été évaluées.
- En l'absence d'un bloc default et si aucun bloc case n'est exécuté, le bloc switch sera traversé sans être exécuté.
- break permet de quitter switch.
- Même dans bloc default, il faut placer un break.

## Exercice avec switch

- Écrire un code C# qui demande à l'utilisateur de saisir une chaîne de caractères (un mois).
- Affiche le nombre de jours de ce mois (28, 30 ou 31).

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# La boucle while

Boucle while : à chaque itération on teste si la condition est vraie avant d'accéder aux traitements

```
while (condition)
{
    ...
}
```

## Exemple

```
int i = 0;
while(i < 5){
    Console.WriteLine(i);
    i++;
}
```

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# La boucle do ... while

La Boucle do ... while exécute le bloc au moins une fois ensuite elle vérifie la condition

```
do {
    ...
}
while(condition);
```

## Exemple

```
int i = 0;
do{
    Console.WriteLine(i);
    i++;
}
while(i < 5);
```

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# La boucle for

Elle permet de répéter un bout de code tant qu'une condition est vraie.

```
for(initialisation; condition; incrémentation){
    ...
}
```

## Exemple

```
for(int i = 0; i < 5; i++) {
    Console.WriteLine(i);
}
```

## Exercice

Écrire un code C# qui permet d'afficher les nombres pairs compris entre 0 et 10.

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# Les tableaux

- Une armoire dans laquelle on range des variables.
  - De même type.
  - le nombre de variables est fixé à la déclaration.
  - Chaque variable est posée sur une étagère.
  - On utilise le nom de l'armoire et on indique l'indice de l'étagère où est stockée la variable afin d'accéder à la variable qui est posée sur une étagère.

# Les tableaux

## Déclaration

$type[]\ nomTableau = new\ type[nbrElement];$

## Exemple

$int[]\ tab = new\ int[6];$

## Remarques

- Tous les éléments du tableau sont initialises à 0.
- $tab[i]$: permet d'accéder à l'élément d'indice $i$ du tableau.
- Le premier élément d'un tableau est d'indice 0.
- On ne peut dépasser la taille initiale d'un tableau ni changer le type déclaré.

# Les tableaux

## Déclaration et initialisation

- $int\ []\ tab = new\ int\ []\ \{3, 5, 4, 9, 7, 1\};$
- $int\ []\ tab = \{3, 5, 4, 9, 7, 1\};$

$tab[6] = 10;$ $\rightarrow$ IndexOutOfRangeException

## Parcourir un tableau avec un for

```
for (int i = 0; i < tab.Length; i++) {

    Console.WriteLine(tab[i]);
}
```

## Parcourir un tableau avec un foreach

```
foreach (int n in tab) {

    Console.WriteLine(n);
}
```

## Déclaration d'un tableau à deux dimensions

*type*[,] *nomTableau* = *new type*[*nbrlignes*, *nbrcolonnes*];

## Déclaration et initialisation

```
int [,] tab2dim =

{

    {1, 2},

     {3, 4}

};
```

## Parcourir un tableau à deux dimensions avec un for

```
for (int i = 0; i < 2; i++)

    for (int j = 0; j < 2; j++)

        Console.WriteLine(tab[i,j]);
```

## Parcourir un tableau à deux dimensions avec un foreach

```
foreach (int n in tab2dim) {

    Console.Write(n);
}
```

# Outline


1. Les variables
   - Opérations de lecture et écriture
   - Opérations sur les variables
2. Les instructions conditionnelles
   - Les opérateurs de comparaison
   - If
   - If ... else
   - If ... else if ... else
   - switch
3. Structures itératives
   - La boucle while
   - La boucle do ... while
   - La boucle for
4. Les tableaux
5. Les méthodes

# Les méthodes

Le but de la méthode est de factoriser du code afin d'éviter d'avoir à répéter sans arrêt le même code.

## Exemple1

```
public static void AffichageBienvenueRhizlane()

{

    Console.WriteLine("Bonjour Rhizlane ");

    Console.WriteLine("Bienvenue dans le monde merveilleux du C#");

}
```

## Exemple2

```
public static void DireBonjour(string prenom, int age)
{
    Console.WriteLine("Bonjour " +prenom);
    Console.WriteLine("Vous avez " +age +"ans");
    Console.WriteLine("Bienvenue dans le monde merveilleux du C#");
}
```

## Exemple3

```
public static double Somme(int n)

{

    double somme = 0;

    for(int i = 0; i <= n; i++) {

      somme+ = i;

    }

    return somme;
}
```

## Remarques

- Une méthode regroupe un ensemble d'instructions pouvant prendre des paramètres et pouvant renvoyer une valeur.
- Les paramètres d'une méthode doivent être utilisés avec le bon type.
- Une méthode qui ne renvoie rien est préfixée du mot-clé void.
- Le point d'entrée d'un programme est la méthode statique Main().
- Le mot-clé return permet de renvoyer une valeur du type de retour de la méthode, à l'appelant de cette méthode.

# Paramètres de méthode

## La différence entre **ref**, **in** et **out**

- **ref:** permet à une méthode d'utiliser et modifier "la valeur originale" d'une variable passée en paramètre (modification facultative).
- **out:** oblige une méthode à modifier la valeur d'une variable passée en paramètre.
- **in:** permet à une méthode d'utiliser "la valeur originale" d'une variable passée en paramètre mais sans pouvoir la modifier.

# Paramètres de méthode

## Exemple1

```
public static void SansModification(in int i)
{
    i--;
}
```

La modification d'un paramètre précédé par **in** n'est pas autorisée → **Error**

## Exemple2

```
public static void Permutation(int a, int b)
{
    int aux = a;
    a = b;
    b = aux;
}
```

## Dans Main:

```
int n = 2;
int m = 5;
Permutation(n, m);
Console.WriteLine($"Après permutation, n = n");
Console.WriteLine($"Après permutation, m = m");
```

## Exemple2: Que s'est-il passé ?

- Par défaut, les types simples sont passés par valeur.
- C'est à dire la méthode appelée (ici Permutation) travaille seulement sur une copie de la variable.
- La méthode appelante conserve donc la valeur originale de la variable.

## Exemple2: Solution

```
public static void Permutation(ref int a, ref int b)
{
    int aux = a;
    a = b;
    b = aux;
}
```

## Dans Main:

```
int n = 2;
int m = 5;
Permutation(ref n, ref m);
Console.WriteLine($"Après permutation, n = n");
Console.WriteLine($"Après permutation, m = m");
```

# Paramètres de méthode

Une méthode ne peut retourner qu'une seule valeur. Si on veut retourner plusieurs valeurs?

## Exemple3

```
public static void FindMinMax(int i, int j, out int max, out int min)
{
    max = i > j ?i : j;
    min = i < j ?i : j;
}
```

## Dans Main:

```
int x;
int y;
FindMinMax(2, 3, out x, out y);
Console.WriteLine($"Le max de 2 et 3 est : x");
Console.WriteLine($"Le min de 2 et 3 est : y");
```

# Paramètres de méthode

Fonction FindMax qui retourne la valeur maximale quel que soit le nombre de paramètres passés.

## Exemple4

```
public static int FindMaxFromNValue(params int[] list)
{
    int max = list[0];
    for(int i = 1; i < list.Length; i++)
    {
        if(list[i] > max)
        max = list[i];
    }
    return max;
}
```

**Dans Main:**

$int\ h = FindMaxFromNValue(2, 8, 5);$