



Variable





Qu'est-ce qu'une variable ?

Une variable est une zone mémoire dans laquelle on peut stocker une valeur.

Elle est composée de 3 éléments :

- Un nom
- Un type
- Une valeur



Déclaration d'une variable

Pour déclarer une variable on associe un type à un nom, avant d'associer celui-ci à une valeur. Pour être capable de bien associer une variable à une valeur, il faut déjà comprendre les types.

Exemple : `string name = "Victor"`

- ➔ `string` : le type
- ➔ `name` : ma variable
- ➔ `"Victor"` : la valeur attribuée à ma variable



Type

Qu'est-ce qu'un type ?

Les types permettent de classer les variables, et de les utiliser plus facilement.

Il existe plusieurs types de variables :

- 📌 `int` : nombre entier
- 📌 `float` : nombre à virgule
- 📌 `string` : chaîne de caractères
- 📌 `bool` : booléen (vrai ou faux)



Pourquoi utiliser des types ?

Par exemple, si on a une variable de type `int`, on pourra l'utiliser pour stocker des nombres entiers.

Si on a une variable de type `string`, on pourra l'utiliser pour stocker des chaînes de caractères.

Si on demande à l'utilisateur d'entrer un texte, on va stocker cette valeur dans une variable de type `string` et non `int`.



Pourquoi utiliser des types ?

Les types permettent de ne pas faire de bêtise, ainsi que de rajouter des interactions entre les variables.

Exemple :

On ne peut pas additionner une chaîne de caractères avec un nombre.



Int

Le type `int` est le type pour les nombres entiers.

Un nombre entier est un nombre qui ne contient pas de virgule.

Il peut aussi être négatif.

Exemple : `125` , `0` , `-12`



Float

Le type `float` est le type pour les nombres à virgule.

Un nombre à virgule est un nombre qui contient une virgule.

Il peut aussi être négatif.

Exemple : `1.025`, `0.0`, `-12.256`



String

Le type `string` est le type pour les chaînes de caractères.

Une chaîne de caractères est une suite de caractères qui forme un mot ou une phrase.

Exemple : `"hello world"`, `"hello"`, `"world"`



Bool

Le type `bool` est le type pour les valeurs booléennes.

Une valeur booléenne est une valeur qui est vraie ou fausse.

Exemple : `true`, `false`



Exemple de déclaration

```
int nom_de_variable = 12;  
float variable_floating_point = 2.0f;  
string what_ever_name = "bonjour les amiches";  
bool name_true_false = true;
```

Il faut bien comprendre ici que le nom des variables est libre de choix.

Mais pour autant il ne faut pas mettre n'importe quoi, car celle ci participe à la clarté de votre code.



Choisir ses noms de variable

Vous demandez à un utilisateur de mettre le nom d'une voiture en input.

Vous stockerez cette valeur dans une string et vous la nommerez en rapport avec l'objet de votre demande.

```
string marque_voiture;  
string car_brand;  
string brand_car;  
string car_name;
```



Exercice sur les variables

- 📌 Créer une variable `age` de type `int` et lui assigner une valeur.
- 📌 Créer une variable `name` de type `string` et lui assigner une valeur.
- 📌 Créer une variable `isAdult` de type `bool` et lui assigner une valeur.
- 📌 Créer une variable `height` de type `float` et lui assigner une valeur.
- 📌 Afficher la valeur de chaque variable avec la fonction
`Console.WriteLine(nom_de_variable)`



Exercice solution

```
int age = 20;  
string name = "Jean";  
bool isAdult = true;  
float height = 1.83f;  
  
Console.WriteLine(age);  
Console.WriteLine(name);  
Console.WriteLine(isAdult);  
Console.WriteLine(height);
```



Exercice 1

Vous possédez un café et vous voulez automatiser la visualisation de certains chiffres.

Ainsi vous souhaitez écrire un programme et récolter des informations.

(Suite à la prochaine page)



Exercice 1

Pour cela vous demandez au serveur du jour de rentrer :

- 📌 Le nombre de cafés servis
- 📌 La quantité de graine (en grammes) utilisées
- 📌 Le nombres de clients servis
- 📌 Le nom du serveur

Créez des variables adaptées à chacune de ces informations et nommez-les judicieusement.



Exercice 2

Bob, Richard et Mark sont pêcheurs sur le même bateau.

Ils font la compétition du nombre de poissons péchés et du poids total péché.

Le capitaine leur propose de créer un petit programme dans lequel chacun entrera leur nom, suivi du nombre de poissons péchés aujourd'hui ainsi que le poids total de leurs prise du jour.

Créez deux variables pour chacun d'eux et nommez-les.



Comment utiliser ma variable

Pour utiliser une variable, c'est à dire accéder à la valeur stockée dans celle-ci, il suffit de l'appeler par son nom.

Pour autant, nous ne pouvons pas redéfinir son type après déclaration.

```
int nom_de_variable;  
  
nom_de_variable = 153;  
  
Console.WriteLine(nom_de_variable); // affichera 153  
// nom_de_variable = "bonjour"    --> erreur car "bonjour" est une string et non un int
```



Attribution

L'opérateur `=` est un opérateur d'assignation.

Cet opérateur permet d'affecter une valeur à une variable.

C'est à dire de lui donner une valeur ou changer sa valeur.



Attribution

Il est possible de redéfinir la valeur d'une variable en lui attribuant une nouvelle valeur.

```
int nom_de_variable;  
  
nom_de_variable = 153;  
Console.WriteLine(nom_de_variable); // affichera 153  
  
nom_de_variable = 12;  
Console.WriteLine(nom_de_variable); // affichera 12
```



Les opérateurs

Les opérateurs sont des symboles qui permettent d'effectuer des opérations sur des variables ou des valeurs.

Nous avons vu précédemment l'opérateur d'assignation `=`.

Mais il en existe pour faire des opérations arithmétiques, pour comparer des variables, pour en incrémenter et pour concaténer.



Les opérations arithmétiques





Addition

Pour additionner deux variables, nous utiliserons l'opérateur `+`.

```
int a = 3;  
int b = 5;  
  
int c = a + b;  
Console.WriteLine(c); // affichera 8
```



Soustraction

Pour soustraire deux variables, nous utiliserons l'opérateur `-`.

```
int a = 3;  
int b = 5;  
  
int c = a - b;  
Console.WriteLine(c); // affichera -2
```



Multiplication

Pour multiplier deux variables, nous utiliserons l'opérateur `*`.

```
int a = 3;  
int b = 5;  
  
int c = a * b;  
Console.WriteLine(c); // affichera 15
```



Division

Pour diviser deux variables, nous utiliserons l'opérateur `/`.

```
int a = 3;  
int b = 5;  
  
int c = a / b;  
Console.WriteLine(c); // affichera 0
```



Modulo

Pour calculer le reste d'une division entre deux variables, nous utiliserons l'opérateur `%`.

```
int a = 31;  
int b = 5;  
  
int c = a % b;  
Console.WriteLine(c); // affichera 1
```



Exercice

Créer une variable :

- 📌 `nbr_pizza` (entier) qui sera égale à 5.
- 📌 `part_de_pizza` (entier) qui sera égale à 6.
- 📌 `nbr_part_de_pizza` qui sera égale au `nbr_pizza` multiplier par `part_de_pizza`.



Exercice

Créer une variable :

- `nbr_personne` qui sera égale à 9.
- `part_de_pizza_par_personne`, qui sera égale au `nbr_part_de_pizza` divisées par le `nbr_personne`.
- `part_de_pizza_restante` qui sera égale au nombre de parts restantes après distribution (utilisez modulo)



Solution

```
int pizza = 5;  
  
int part_de_pizza = 6;  
  
int nbr_part_de_pizza = pizza * part_de_pizza;  
  
int nbr_personne = 9;  
  
int part_de_pizza_par_personne = nbr_part_de_pizza / nbr_personne;  
  
int part_de_pizza_restante = nbr_part_de_pizza % nbr_personne;
```



Opérations de comparaisons

La comparaison de variables est au cœur des boucles logiques et conditionnelles, elle vous permettra de comparer deux variables.

Le retour d'une comparaison est une variable `bool` true ou false en fonction de la vérité de comparaison.



Egalité

Pour comparer si deux variables sont égales, nous utiliserons l'opérateur `==`.

```
int a = 3;  
int b = 5;  
  
bool c = a == b;  
Console.WriteLine(c); // affichera false
```



Supérieur

Pour comparer si une variable est supérieure à une autre, nous utiliserons l'opérateur `>`.

```
int a = 3;  
int b = 5;  
  
bool c = a > b;  
Console.WriteLine(c); // affichera false
```



Inférieur

Pour comparer si une variable est inférieur à une autre, nous utiliserons l'opérateur <.

```
int a = 3;  
int b = 5;  
  
bool c = a < b;  
Console.WriteLine(c); // affichera true
```



Supérieur ou égal

Pour comparer si une variable est supérieure ou égale à une autre, nous utiliserons l'opérateur `>=`.

```
int a = 3;  
int b = 5;  
  
bool c = a >= b;  
Console.WriteLine(c); // affichera false
```



Inférieur ou égal

Pour comparer si une variable est inférieure ou égale à une autre, nous utiliserons l'opérateur `<=`.

```
int a = 3;  
int b = 5;  
  
bool c = a <= b;  
Console.WriteLine(c); // affichera true
```



Exercice

Créer 4 variables, `a`, `b`, `c` et `d` de type `float` tel que :

- 📌 `a` = 1.5
- 📌 `b` = 5
- 📌 `c` = 1.5
- 📌 `e` = 0.5



Exercice

En utilisant l'opérateur `>` et 2 des 4 variables (`a`, `b`, `c`, `d`)

Créer 2 variables de type `bool` tel que :

- 📌 `f` = true
- 📌 `g` = false



Exercice

En utilisant l'opérateur < et 2 des 4 variables (a , b , c , d)

Créer 2 variables de type bool tel que :

- 📌 h = true
- 📌 i = false



Exercice

En utilisant l'opérateur `==` et 2 des 4 variables (`a`, `b`, `c`, `d`)

Créer 2 variables de type `bool` tel que :

- 📌 `j` = true
- 📌 `k` = false



Solution :

```
float a = 1.5f;  
float b = 5f;  
float c = 1.5f;  
float d = 0.5f;  
  
bool f = b > a;  
bool g = d > a;  
  
bool h = c < b;  
bool i = c < d;  
  
bool j = a == c;  
bool k = d == c;
```



Opérateurs d'incrémantation

En informatique, l'incrémantation est l'opération qui consiste à ajouter 1 à un compteur.

Pour incrémenter une variable de 1, nous utiliserons l'opérateur `++`.

```
int a = 3;  
  
a++;  
Console.WriteLine(a); // affichera 4
```



Opérateurs de décrémentation

L'opération inverse, la décrémentation, consiste à retirer 1 au compteur.

Pour décrémenter une variable de 1, nous utiliserons l'opérateur `--`.

```
int a = 3;  
  
a--;  
Console.WriteLine(a); // affichera 2
```



Opérateur incrémantation de n

Pour incrémenter une variable de n, nous utiliserons l'opérateur `+=`.

```
int a = 3;  
  
a += 5;  
Console.WriteLine(a); // affichera 8
```



Opérateur décrémentation de n

Pour décrémenter une variable de n, nous utiliserons l'opérateur `-=`.

```
int a = 3;  
  
a -= 5;  
Console.WriteLine(a); // affichera -2
```



Exercice

Créer deux variables `int`: `salaire` et `solde_banque`

- 📌 Initialiser `salaire` à 1280 et `solde_banque` à 329.
- 📌 Incrémenter `solde_banque` de `salaire`



Exercice

Créer deux nouvelles variables `int : depense_course` et `loyer`

- 📌 Initialiser `depense_course` à 450
- 📌 Initialiser `loyer` à 600
- 📌 Décrémenter `solde_banque` de `depense_course`
- 📌 Décrémenter `solde_banque` de `loyer`



Exercice

Vous avez une augmentation de salaire de 150

📌 Incrémenter votre salaire de 150



Solution :

```
int salaire = 1280;
int solde_banque = 329;

solde_banque += salaire;

int depense_course = 450;
int loyer = 600;

solde_banque -= depense_course;
solde_banque -= loyer;

salaire += 150;
```



Opérateurs de concatenation

Pour concaténer deux chaînes de caractères, nous utiliserons l'opérateur `+`.

```
string a = "Hello";
string b = "World";

string c = a + b;
Console.WriteLine(c); // affichera HelloWorld
```



Scope

Le Scope | La portée





Qu'est-ce que le scope

Le scope est la portée d'une variable, portée géographique : c'est là où la variable est visible dans le code.

Les variables déclarées dans une :

- 📍 Boucle ne peuvent être utilisées que dans cette boucle.
- 📍 Fonction ne peuvent être utilisées que dans cette fonction.



Exemple:

```
int a = 3;  
{  
int b = 5;  
Console.WriteLine(a); // affichera 3  
Console.WriteLine(b); // affichera 5  
}  
Console.WriteLine(b); // affichera une erreur
```

Ici, **b** est déclaré dans des accolades **{ }**, et n'est donc accessible qu'au sein de ces **{ }**.



Variable globale

Une variable globale est une variable déclarée en dehors de tout scope.

Elle est donc accessible partout dans le code.

Dans l'exemple ci-dessus, `a` est une variable globale car elle est déclarée en dehors de tout scope (`{ }`), et est accessible partout (toujours ci-dessus nous avons accès à `a` dans le même scope où est déclaré `b`)



Variable locale

Une variable locale est une variable déclarée dans un scope, et qui ne peut être utilisée que dans ce scope.

b est une variable dite locale à son scope.

```
int a = 3;
{
    int b = 5;
    Console.WriteLine(a); // affichera 3
    Console.WriteLine(b); // affichera 5
}
Console.WriteLine(b); // affichera une erreur
```



Scope dans un scope

```
{      // debut du premier scope
    int a = 3;
    Console.WriteLine(a);
    {          // debut du scope 2
        int b = 4;
        Console.WriteLine(a);
        Console.WriteLine(b);
        {          // debut du scope 3
            int c = 6;
            Console.WriteLine(a);
            Console.WriteLine(b);
            Console.WriteLine(c);

        }      // fin du scope 3
                // variable c n'est plus accessible ici !!
    }    // fin du scope 2
            // variable b n'est plus accessible ici !!
}      // fin du premier scope
```



Scope dans un scope

Une variable est définie dans son scope comme vu précédemment et ce scope possédera toutes les entités présente dans celle-ci.

C'est pour ça que le programme ci-dessus fonctionne et affiche bien les résultats attendus



Exercice

Créer 3 variables globales telles que :

- 📌 `jean` une string initialisée à "Jean"
- 📌 `ville` une string initialisée à "Paris"
- 📌 `annee` un int initialisée à 1998



Exercice

Créer un scope dans lequel seront instanciés :

- annee2 un int égal à 1999
- lea une string égale à "Lea"

Créer un second scope dans le premier et instanciez :

- fact_check un booléen qui vérifiera `annee < anne2` ;
- Une string copie_ville qui sera égale à ville



Solution :

```
string jean = "Jean";
string ville = "Paris";
int annee = 1998;
{
    int annee2 = 1999;
    string lea = "Lea";
    {
        bool fact_check = annee < annee2;
        string copie_ville = ville;
    }
}
```