La syntaxe de base du C#

Dans cet article, on présente les bases essentielles du langage C# dans un ordre logique, pour comprendre comment écrire et organiser un programme simple.

Ce qu'on va savoir faire

À la fin de cet article, tu seras capable d'écrire ce programme complet :

```
using System;

class Program
{
    static void Main()
    {
        Console.Write("Quel est ton prénom ? ");
        string prenom = Console.ReadLine();

        Console.Write("Quel est ton âge ? ");
        int age = int.Parse(Console.ReadLine());

        Console.WriteLine($"Bonjour {prenom} !");

        if (age >= 18)
        {
            Console.WriteLine("Tu es majeur(e).");
        }
        else
        {
            Console.WriteLine("Tu es mineur(e).");
        }

        Console.WriteLine("Comptons jusqu'à ton âge :");
        for (int i = 1; i <= age; i++)
        {
            Console.WriteLine(i);
        }
    }
}
```

Le bloc de base

En C# moderne, tu peux écrire ton code **directement**, sans structure autour :

```
// Mon premier programme
Console.WriteLine("Bonjour !");
```

C'est suffisant. Le compilateur s'occupe du reste dans les coulisses. C'est ce qu'on appelle des ***top-level statements***.

💡 Si tu croises cette structure dans des tutoriels ou anciens projets, ne sois pas surpris(e) :

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Bonjour !");
    }
}
```

C'est l'ancienne façon d'écrire — obligatoire avant 2020. Elle est toujours valide, et on la retrouvera naturellement quand on abordera la programmation orientée objet.

Nuance importante à connaître

Un seul fichier du projet peut utiliser les top-level statements. Dès qu'on a plusieurs fichiers/classes, la structure complète redevient nécessaire.

🧪 Exercice 1 — Premier programme

Objectif : créer et exécuter un programme minimal.

Consigne : écris un programme qui affiche ces deux lignes :---

🧪 Exercice 1 — Premier programme

Objectif : créer et exécuter un programme minimal.

Consigne : recopie la structure de base et ajoute cette ligne dans `Main()` :

```
Console.WriteLine("Mon premier programme C# !");
```

Résultat attendu :

```
Mon premier programme C# !
```

À retenir : `Console.WriteLine()` affiche une ligne dans la console.

---

2. Variables

Une **variable** sert à stocker une valeur pour la réutiliser.

```
string prenom = "Alice";
int age = 17;
```

Syntaxe :

```
type nom = valeur;
```

Les types les plus courants pour commencer :

Type	Contient	Exemple
`int`	un entier	`42`
`double`	un décimal	`3.14`
`string`	du texte	`"Bonjour"`
`bool`	vrai ou faux	`true / false`

⚠️ Un `string` se note avec des guillemets `"..."`, un `char` avec des apostrophes `'...'`.

🧪 Exercice 2 — Déclarer des variables

Objectif : déclarer et afficher plusieurs variables.

Consigne : déclare ces trois variables, puis affiche-les chacune sur une ligne séparée :

```
Prénom : Alice
Âge    : 17
Taille : 1.68
```

Hint : utilise `Console.WriteLine()` trois fois.

3. Afficher avec interpolation

La façon la plus lisible d'afficher du texte **et** des variables ensemble :

```
1  string prenom = "Alice";
2  int age = 17;
3
4  Console.WriteLine($"Bonjour {prenom}, tu as {age} ans.");
```

Résultat :

```
Bonjour Alice, tu as 17 ans.
```

💡 Le `$` devant les guillemets active l'interpolation. Les variables se placent entre `{ }`.

🧪 Exercice 3 — Interpolation

Objectif : utiliser l'interpolation de chaînes.

Consigne : à partir des variables `prenom`, `age` et `taille` de l'exercice précédent, affiche **une seule ligne** :

```
Alice a 17 ans et mesure 1.68 m.
```

---

4. Lire une entrée utilisateur

`Console.ReadLine()` lit ce que l'utilisateur tape :

```
1  Console.Write("Quel est ton prénom ? ");
2  string prenom = Console.ReadLine();
```

💡 `Console.Write()` n'ajoute pas de retour à la ligne — pratique pour mettre la réponse sur la même ligne que la question.

Convertir l'entrée

`ReadLine()` renvoie **toujours du texte**. Si tu attends un nombre, il faut convertir :

```
1  Console.Write("Quel est ton âge ? ");
2  int age = int.Parse(Console.ReadLine());
```

🧪 Exercice 4 — Saisie utilisateur

Objectif : lire des données saisies et les réafficher.

Consigne : écris un programme qui demande le prénom et l'année de naissance, puis affiche :

```
Bonjour Alice ! Tu es né(e) en 2007.
```

---

5. Opérations sur les nombres

```
1  int a = 10;
2  int b = 3;
3
4  Console.WriteLine(a + b);   // 13
5  Console.WriteLine(a - b);   // 7
6  Console.WriteLine(a * b);   // 30
7  Console.WriteLine(a / b);   // 3  (division entière !)
8  Console.WriteLine(a % b);   // 1  (reste de la division)
```

⚠️ La division entre deux `int` donne un `int` : `10 / 3 = 3`, pas `3.33`.
Pour un résultat décimal, utilise `double` : `10.0 / 3 = 3.333...`

Raccourcis utiles :

```
1  x += 5;   // x = x + 5
2  x -= 2;
```

```
x *= 3;
x /= 2;
x++;      // x = x + 1
x--;      // x = x - 1
```

🧪 Exercice 5 — Calculatrice simple

Objectif : combiner saisie, conversion et opérations.

Consigne : demande deux entiers à l'utilisateur, puis affiche :

```
Somme      : 13
Différence : 7
Produit    : 30
Reste      : 1
```

6. Les conditions

if / else

```
if (age >= 18)
{
    Console.WriteLine("Majeur");
}
else
{
    Console.WriteLine("Mineur");
}
```

if / else if / else

```
if (age < 12)
{
    Console.WriteLine("Enfant");
}
else if (age < 18)
{
    Console.WriteLine("Adolescent");
}
else
{
    Console.WriteLine("Adulte");
}
```

Opérateurs de comparaison :

Opérateur	Signification
`==`	égal à
`!=`	différent de
`>`	strictement plus grand
`<`	strictement plus petit
`>=`	plus grand ou égal
`<=`	plus petit ou égal

🧪 Exercice 6 — Conditions

Objectif : utiliser `if / else if / else`.

Consigne : demande une note sur 20, puis affiche :

```
18 → Excellent !
14 → Bien
10 → Suffisant
8  → Insuffisant
```

Définis toi-même les seuils.

7. Les boucles

Boucle `while`

Répète **tant que** la condition est vraie :

```
1  int i = 0;
2
3  while (i < 5)
4  {
5      Console.WriteLine(i);
6      i++;
7  }
```

Boucle `for`

Quand on connaît le nombre de répétitions à l'avance :

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

Structure d'un `for` :

```
for ( initialisation ; condition ; mise à jour )
```

🧪 Exercice 7 — Boucles

Objectif : utiliser une boucle `for`.

Consigne : demande un nombre `n` à l'utilisateur et affiche sa **table de multiplication** de 1 à 10 :

```
3 x 1  =  3
3 x 2  =  6
3 x 3  =  9
...
3 x 10 = 30
```

---

8. Exercices de synthèse

Ces exercices combinent plusieurs notions.

🏁 Synthèse 1 — Le programme d'accueil

Reproduis le programme présenté en introduction (sans regarder !).
Il doit demander prénom + âge, afficher un message personnalisé, indiquer si l'utilisateur est majeur ou non, et compter jusqu'à son âge.

🏁 Synthèse 2 — Pair ou impair

Consigne : demande 5 nombres à l'utilisateur (avec une boucle `for`), et pour chacun indique s'il est pair ou impair.

Hint : `nombre % 2 == 0` → pair.

Résultat attendu :

```
Nombre 1 : 7 → impair
Nombre 2 : 4 → pair
...
```

🏁 Synthèse 3 — Devine le nombre

Consigne : le programme choisit le nombre 42. L'utilisateur doit le deviner.
Tant qu'il n'a pas trouvé, le programme dit *"trop grand"* ou *"trop petit"*.
Quand il trouve, il affiche le nombre d'essais.

Hint : utilise une boucle `while` et un compteur.

Annexe — Référence rapide

```
// Afficher
Console.WriteLine("texte");          // avec retour à la ligne
Console.Write("texte");              // sans retour à la ligne
Console.WriteLine($"Bonjour {nom}"); // avec variable

// Lire
string texte = Console.ReadLine();

// Convertir
int n    = int.Parse(Console.ReadLine());
double d = double.Parse(Console.ReadLine());

// Commentaires
// ligne unique
/* plusieurs
   lignes */
```