# Norme

# Pourquoi une norme ?

Tout developpeur a son propre style de code, et c'est bien normal.
Mais il est important de respecter une norme pour que tout le monde puisse comprendre le code que vous avez écrit.
Tout le monde c'est vous aussi dans 6 mois.

# Pourquoi une norme ?

Passer de :

```
void fonction1(int[] tab) { for (int i = 0; i < tab.Length; i++){  Console.Write(tab[i] + " ");}Console.WriteLine();}
```

```
void printTab(int[] tab)
{
    for (int i = 0; i < tab.Length; i++)
    {
        Console.Write(tab[i] + " ");
    }
    Console.WriteLine();
}
```

# Travailler correctement

Pour le moment on a vu principalement des exercices individuels de petite taille.
La realité est bien différente, vous allez travailler sur des projets de plus en plus gros, et vous allez devoir.
Vos fonctions vont devenir plus complexe. Etre appelé par d'autres fonctions, et vous allez devoir appeler d'autres fonctions.

# Travailler correctement

Tous ca pour dire que vous allez passer beaucoup de temps à lire du code, et à le modifier.
Si vous n'avez pas une norme, vous allez perdre beaucoup de temps à comprendre ce que fait le code, pour le modifier.
De meme pour le nomage des variables, des fonctions, des classes, etc.
Pour eviter de perdre du temps, de l'energie sur le long terme il est important de respecter une norme.
Une norme qui va rendre votre travail propre et lisible.

# Travail en équipe

Il en va de meme dans un projet en équipe.
Si vous n'avez pas une norme, vous allez perdre beaucoup de temps à comprendre ce que fait le code, et à le modifier.
Vos collègues vont perdre du temps à comprendre votre code, et à le modifier.
Et la productivite va chuter. De meme que pour les echanges entre collaborateurs, qui vont etre plus longs et plus compliqués.

# Reprendre un projet

Un exemple frappant est celui de reprendre un projet que vous avez fait il y a 6 mois.
Ouvrez votre fichier et essayer de dire ce que fait votre code.

```
void fonction1(int[] tab) { for (int i = 0; i < tab.Length; i++){  Console.Write(tab[i] + " ");}Console.WriteLine();}
```

```
void printTab(int[] tab)
{
    for (int i = 0; i < tab.Length; i++)
    {
        Console.Write(tab[i] + " ");
    }
    Console.WriteLine();
}
```

# Lire un code

Lire un code c'est comme lire un livre.
Si le livre est bien écrit, vous allez comprendre ce que dit l'auteur.
Si le livre est mal écrit, vous allez perdre du temps à comprendre ce que dit l'auteur.
C'est pareil pour un code.
L'objectif c'est de transmettre, et de se greffer sur le travail des autres.
Et sans norme, ca devient vite compliqué.

# Norme de base

# Le Nomage

- 📌 Pas de trais d'union dans les noms de variables, de fonctions, de classes, etc.
  cela peut empecher la compilation dans certains cas.
  Privilegier les underscore. _
- 📌 Pas d'accents ni d'espace dans les noms de variables, de fonctions, de classes, etc.
  On evite les caracteres speciaux aussi.
- 📌 Variable private commence par un underscore _.

```
public int ma_variable = 0;
private int _element = 0;
```

# La declaration

- 📌 Une variable par ligne, et on instancie la variable au moment de sa declaration.
  Toutes les variables sont en debut de fonction, classes, etc.
- 📌 Separation des declaration de variables et de la logique de la fonction par un saut de ligne.

```
int a, b, c;
if (a !=0) {
        b =1;
        c= 2;
}
```

```
int a = 0;
int b = 0;
int c = 0;

if (a != 0)
{
        b = 1;
        c = 2;
}
```

# Les opérateurs

📌 Les opérateurs sont séparés par des espaces avant et apres celui-ci.

```
int a=0;
```

```
int a = 0;
```

# Les opérateurs

📌 Les parenthèses sont directement collées aux variables, mais separees des conditions.

```
if(a==0  )
```

```
if (a == 0)
```

# Les opérateurs

📌 Pareil pour les `[]`

```
int[] tab = new int[ 10 ];
```

```
int[] tab = new int[10];
```

# Lisibilité

- 📌 Aerer le code, et separer les blocs de code par des sauts de ligne.
- 📌 Eviter les lignes trop longues, vous pouvez mettre un retour chariot pour separer les lignes (80 caracteres par ligne).

```
if (a == b && c == d && e == f && g == h && i == j && k == l && m == n && o == p && q == r && s == t && u == v && w == x && y == z)
```

```
if (a == b && c == d && e == f && g == h && i == j &&
        k == l && m == n && o == p && q == r && s == t &&
        u == v && w == x && y == z)
```

# Lisibilité

📌 À chaque fois qu'un nouveau bloc d'instructions est débuté, et qu'il est subordonné à une ligne précédente (if, while, etc.), alors il faut le mettre en retrait par rapport à la ligne précédente.

```
if (a == b)
{
        c = d; // retrait d'une tabulation (4 espaces)
        if (e == f)
        {
                g = h; // retrait d'une tabulation (4 espaces)
        }
}
```

# Lisibilité

📌 Les accolades sont toujours sur la meme ligne que la condition ou une apres et l'autre sur la meme ligne, mais pas l'un puis l'autre.

```
if (condition) {
        // code
}
else
{
        // code
}
```

```
if (condition)
{
        // code
}
else
{
        // code
}
```

```
if (condition) {
        // code
}
else {
        // code
}
```

# Lisibilité

📌 Les variables constantes devronts etre mis en constantes (const) lors de l'instanciations.

```
const double PI = 3.1416; // le nombre pi mis en constante
double rayonSphere = 0.0; // rayon de la sphère saisi
double volumeSphere = 0.0; // volume de la sphère calculé

Console.Out.WriteLine("Entrez le rayon de la sphère.");
rayonSphere = double.Parse(Console.In.ReadLine());
volumeSphere = 4 * PI * rayonSphere * rayonSphere * rayonSphere / 3;
```

# Les conventions de nommage

- 📌 Les noms des methodes sont generalement compose d'un seul verbe et au moins un nom.
- 📌 Pas d'espace, on separe les mots par des underscore `_` ou une majuscule au mot suivant.

```
void printTab(int[] tab);
int do_sum(int a, int b);
```

# La convention imposee par l'employeur

Evidemment c'est conventions scientifique sont des conventions generales, mais il est possible que votre employeur impose une convention particuliere.
Il faut donc se renseigner sur les conventions de nommage de votre entreprise. (qui vous seront probablement fournis dans un document de norme de codage en entree entreprise.)