

# BASES DU C#

---

Et utilisation de visual studio

# Le C# : C Sharp

- Développé par Microsoft pour le framework .net :
  - Librairie de fonctions très nombreuses
  - Ensemble de règles (pattern) pour développer une application
- Langage objet très proche du java et du C++
- Langage à typage fort

# Outil de « dév » : Visual Studio

- Visual studio community : ensemble complet d'outils pour développer :
  - Des applications consoles
  - Des applications fenêtrées
  - Des applications WEB
  - Des services WEB XML
  - Des applications mobiles

Offres logiciels Microsoft

Offres logiciels VMware

Autres offres logiciels

## Bureau et mobile (5)



### Développement .NET Desktop

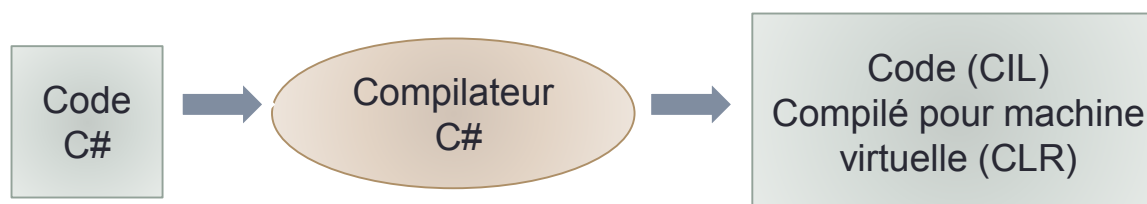
Générez des applications WPF, WinForms et Xamarin en C#, Visual Basic et F# à l'aide de Visual Studio



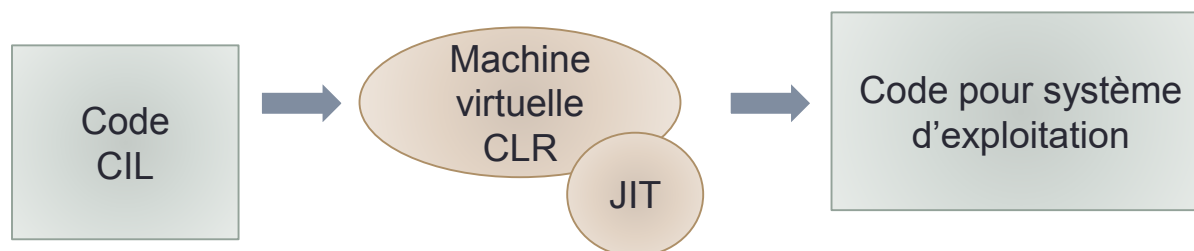
Pour l'installer chez vous ! Suivez le lien « offres logiciels Microsoft » sur l'intranet info

# C# : langage compilé puis interprété

- **Phase de compilation:** code compilé et traduit en langage commun (CIL) pour la machine virtuelle CLR.



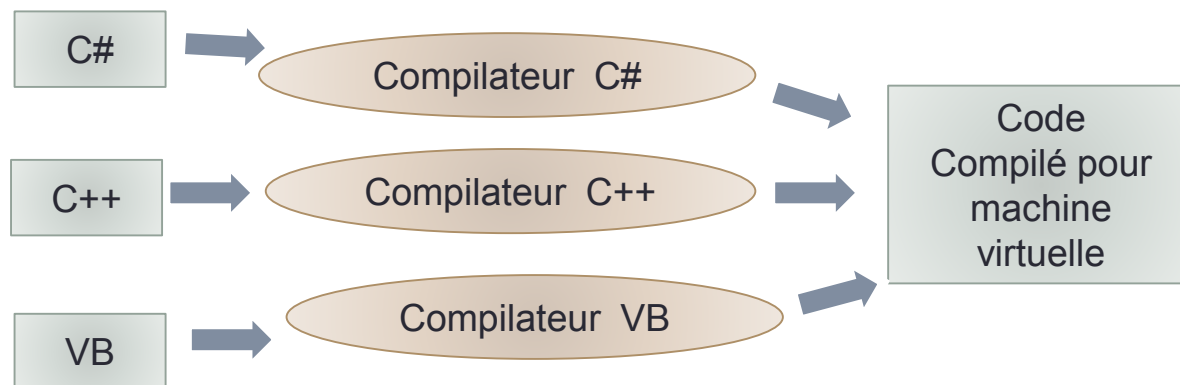
- **Phase d'interprétation :** code traduit à la volé par le compilateur JIT (Just In Time) de la machine virtuelle pour l'OS sur lequel elle est installée



Optimisation de la mémoire : garbage collector  
Accélération de l'exécution du code en réutilisant  
dès que possible le cache

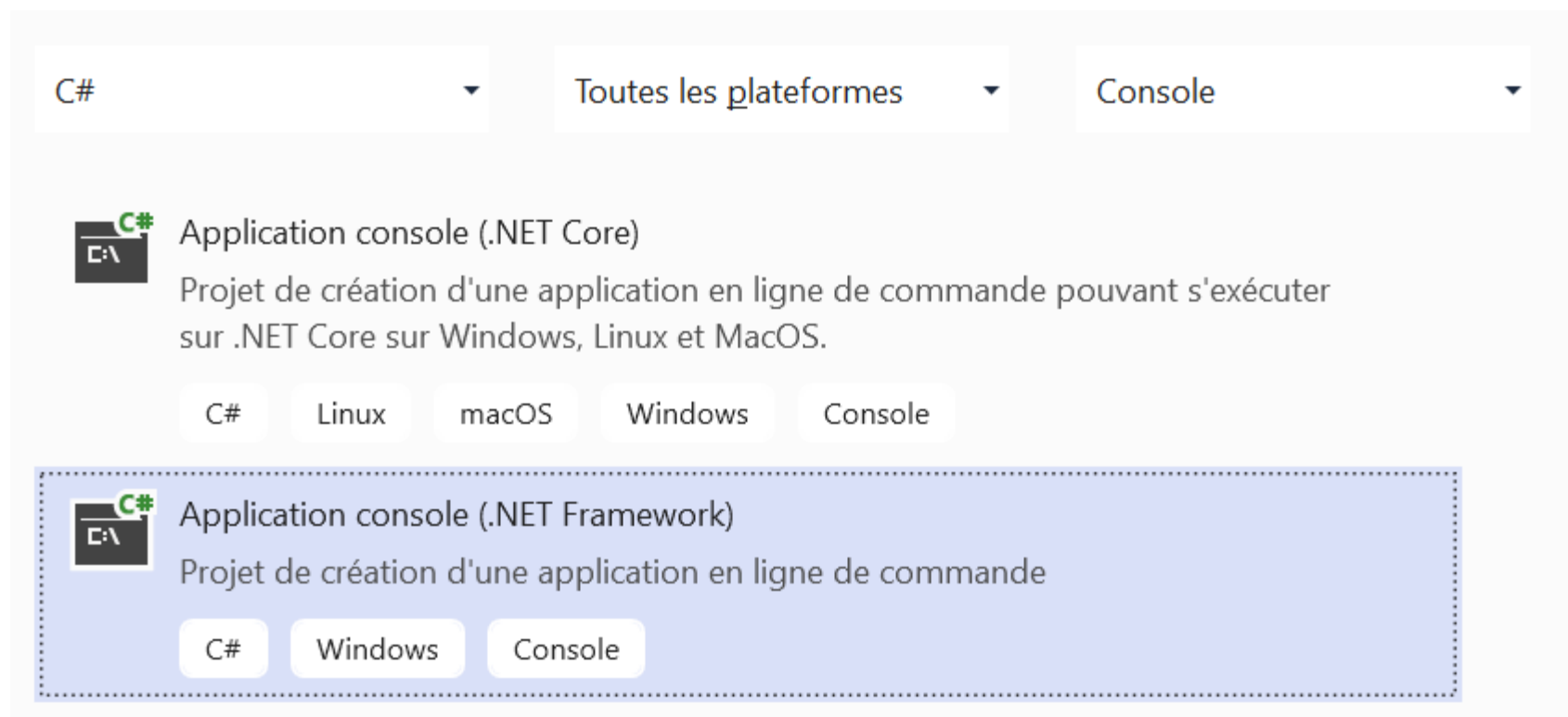
# C# : langage compilé puis interprété

- Conclusion : Compiler : ne génère pas un fichier exécutable par un système d'exploitation, mais un fichier exécutable par une machine virtuelle.
- Spécificité de .net : hétérogénéité des langages



# Organisation du code sous visual studio

- Il faut commencer par créer un projet.



# Organisation du code sous visual studio

- Par défaut, une solution de même nom est créée.

Application console (.NET Framework) C# Windows Console

Nom du projet

Projet1

Emplacement

D:\IUT-ACY\mesCours\2019-2020\M2103-C#\demo\

Nom de la solution ⓘ

Projet1

☐ Placer la solution et le projet dans le même répertoire

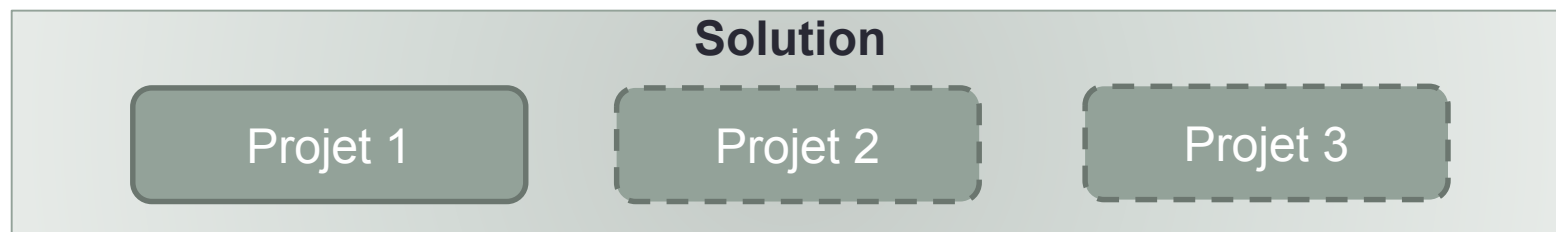
Framework

.NET Framework 4.7.2

Ne pas cocher

# Organisation du code sous visual studio

- Solution = un ou plusieurs projets



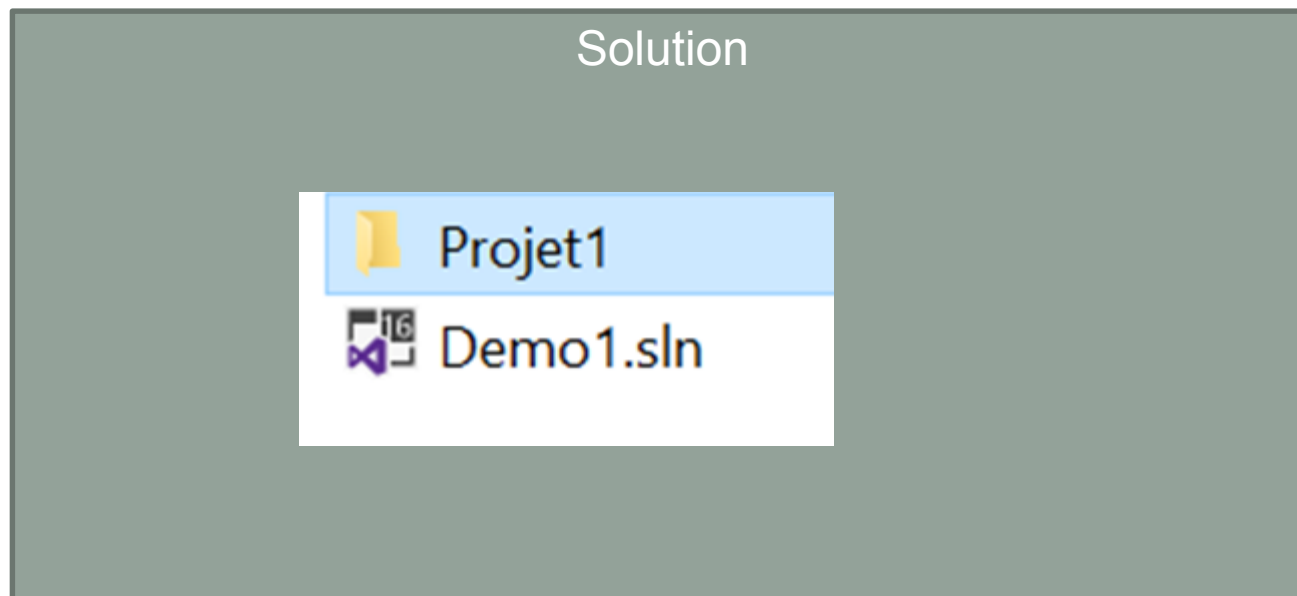
- On peut créer un projet et l'ajouter à une solution existante

The screenshot shows the 'Add New Project' dialog in Visual Studio. It has several fields: 'Nom du projet' with 'Projet2' entered; 'Emplacement' with a file path; 'Solution' with a dropdown menu highlighted by a red rectangle, showing 'Ajouter à la solution'; and 'Nom de la solution' with 'Projet1' entered. An information icon is next to the 'Nom de la solution' label.



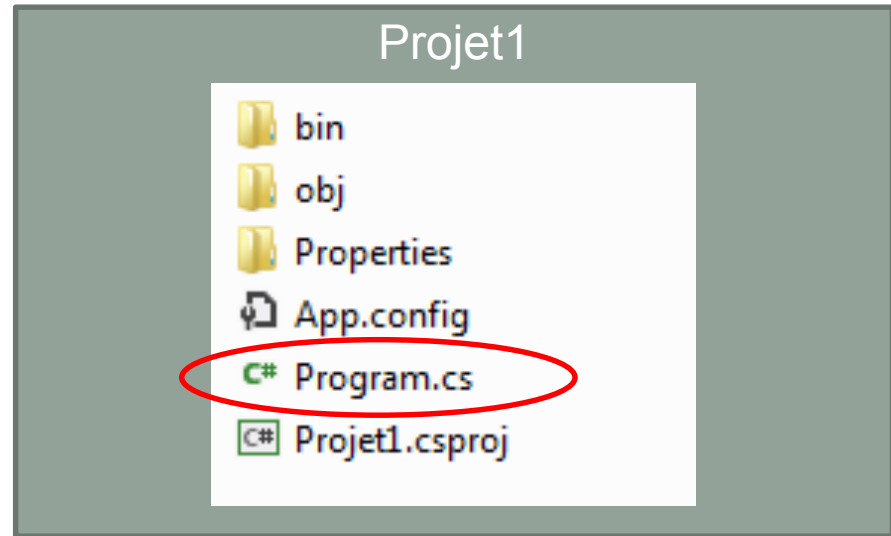
# Contenu d'une solution

- Répertoire(s) : un par projet
- Sln : fichier de configuration (liste des projets ....) de la solution



# Contenu d'un projet

- Répertoires
  - bin : fichiers exécutables
  - obj : fichiers de debug
  - Properties: paramètres de compilation



- **.config: fichier pour définir des variables « paramètres »**  
utilisables dans le code source : (étudié plus tard)
- .csproj: fichier de configuration du projet (liste des fichiers, options de compilation....)
- **.cs: fichier de code source**

# Squelette d'un programme

- Code généré automatiquement

The image shows a C# code editor window titled 'C# Projet1'. The code is a template for a program, with several callouts explaining its parts:

```
C# Projet1
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Projet1
{
    // Oréférences
    class Program
    {
        // Oréférences
        static void Main(string[] args)
        {
        }
    }
}
```

Liens vers les namespaces ( = répertoires) contenant les fonctions les plus utilisées

Namespace : nom du répertoire du projet

Class : ici nom du fichier contenant le main

Main comme en C

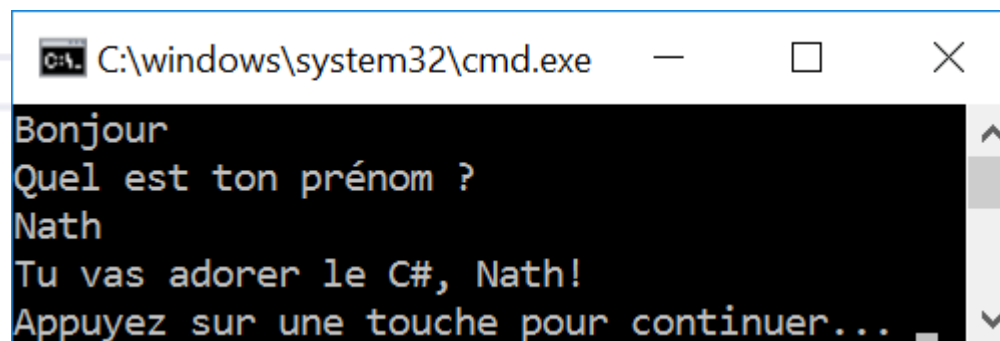
# 1er programme

- On écrit dans la méthode Main :

```
class Program
{
    0 références
    static void Main(string[] args)
    {

        Console.WriteLine("Bonjour");
        Console.WriteLine("Quel est ton prénom ?");
        String prenom = Console.ReadLine();
        Console.WriteLine("Tu vas adorer le C#, " + prenom + "!");

    }
}
```



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\windows\system32\cmd.exe'. The window contains the following text: 'Bonjour', 'Quel est ton prénom ?', 'Nath' (the user's input), 'Tu vas adorer le C#, Nath!', and 'Appuyez sur une touche pour continuer...'.

# Complétion et détection d'erreurs

- Sans compilation : certaines erreurs auto détectées

The screenshot shows the Visual Studio IDE with a C# file named `Program.cs` open. The code defines a `PremierProjet` namespace containing a `Program` class with a `Main` method. Inside the `Main` method, the line `Console.WriteLine(" Premier ordre c# ");` is highlighted with a red circle. Below the code editor, the `Liste d'erreurs` (Error List) window is visible. It shows one error: `'System.Console' ne contient pas de définition pour 'WritLine'`. This error entry is also circled in red. A speech bubble points to the `Liste d'erreurs` menu, with the text `Menu : Affichage/ liste d'erreurs`.


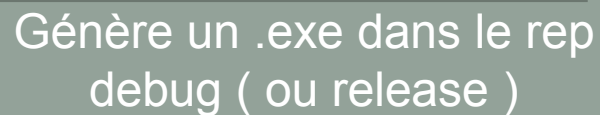
```
namespace PremierProjet
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(" Premier ordre c# ");
        }
    }
}
```

Liste d'erreurs

1 erreur | 0 avertissements | 0 messages

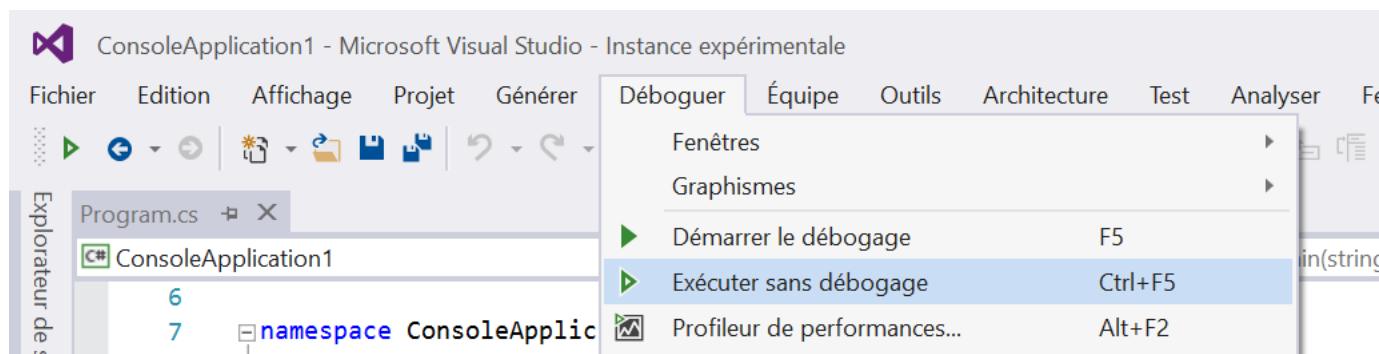
Description	Fichier	Ligne
1 'System.Console' ne contient pas de définition pour 'WritLine'	Program.cs	13

- Générer la solution


**Projet1.exe**

# 1ere execution

- Préférez le mode sans débogage (Ctrl + F5)



# Langage typé

- Il existe 2 grandes catégories de type :

- Les types primitifs. Ex : int, float, double

```
int age = 18 ;
```

Variable simple disposant d'opérateurs ( + - ....)

- Les types complexes :

- Structures ou classes Ex : DateTime, Point, Int16, String ...

```
DateTime remiseDiplome = new DateTime (2017,6,30);
```

Variable complexe disposant d'opérateurs et de fonctions supplémentaires (AddYear, AddMonth...)



# Types primitifs : Les entiers

Type C#	valeurs	Structure
sbyte (byte)	-128 à 127 ( 0 à 255 )	System.Sbyte (System.Byte)
short (ushort)	-32768 à 32767 (0 à 65 535)	System.Int16 (System.UInt16)
int (uint)	-2147483648 à 2147483647 (0 à 4 294 967 295)	System.Int32 (System.UInt32)
long (ulong)	-9223372036854775808 à 9223372036854775807 ( 0 à 18 446 744 073 709 551 615)	System.Int64 (System.UInt64)

## Exemple :

```
byte age = 25;  
// int age = 25 ;
```

Une structure a été mise en place pour chaque type primitif afin de l'enrichir.

# Types primitifs : Les réels

Type C#	valeurs	précision	Structure
float	$\pm 1.5 \times 10^{-45}$ à $\pm 3.4 \times 10^{38}$	7 chiffres	System.Single
double	$\pm 5.0 \times 10^{-324}$ à $\pm 1.7 \times 10^{308}$	15-16 chiffres	System.Double
decimal	$\pm 1.0 \times 10^{-28}$ à $\pm 7.9 \times 10^{28}$	28-29 chiffres significatifs	System.Decimal

## Exemples :

double prix = 39.99;  
// ou float prix = 39.99F;

Attention : on doit mettre le suffixe F  
pour différencier un float d'un  
double, car par défaut double

# Conversion de types primitifs proches

- Implicite

Exemple :

```
int iAge = 25 ;  
float fAge = iAge ;
```

- Explicite :

Exemple :

```
float fAge = 35.5F;  
int iAge = (int) fAge;
```

iAge = 35

- Remarque : la conversion peut être source d'erreur si la donnée n'est pas adaptée au nouveau type.

# Les autres types primitifs

- Le type caractère : char – Structure associée [System.Char](#)
  - Attention : char et String sont <>

Exemples :

```
char categorie='A';
```

- Le type booléen : bool – Structure associée [System.Boolean](#)

Exemples :

```
bool trouve = true;
```

# Opérations sur les variables

- Opérateurs mathématiques: + - \* / %

Exemple :

```
int age = 30;  
age = age - 5 ; // ou age -= 5;  
age = age / 2 ;
```

25/2 = 12 !  
Rappel : / entiers

- Opérateurs d'incrémentation ou de décrémentation: ++, --

Exemple :

```
int age = 10 ;  
age++;  
char c = 'a';  
c ++ ;
```

# Constante

- Une constante :

Exemple :

**const** int majorite =18 ;

# Structure conditionnelle : if

- Idem C, PHP, ...
- Rappel : attention à la portée des variables

```
if (condition)
```

```
{  
    instructions;  
}
```

Pas besoin  
de { } s'il n'y  
a qu'une  
seule  
instruction

```
if ( condition )
```

```
{  
    instructions ;  
}
```

```
else
```

```
{  
    instructions;  
}
```

```
if (condition)
```

```
{  
    instructions ;  
}
```

```
else if (condition)
```

```
{  
    instructions;  
}
```

```
else
```

```
{  
    instructions;  
}
```

# Structure conditionnelle : switch

- Identique au C : ne teste que l'égalité !
- La variable peut être une chaîne

```
switch ( variable )  
{  
    case valeur1 : instructions;  
                    break;  
    case valeur2 : instructions  
                    break;  
    ...  
    default : instructions;  
              break;  
}
```

```
switch ( jour )  
{  
    case "lundi" : instructions;  
                  break;  
    case "mardi" : instructions ;  
                  break;  
    ...  
    default : instructions;  
              break;  
}
```



# Boucles : For, while, do ... while

- On peut déclarer le compteur au sein du for
- Rappel : attention à la portée des variables

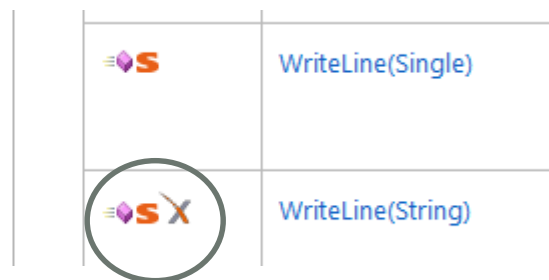
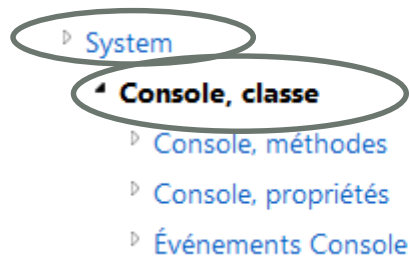
```
for (int i=0;i<5;i++)  
    // instruction à répéter
```

```
int i =0 ;  
do  
{  
    // instructions à répéter  
    i ++;  
} while (i<5);
```

# Afficher : Console.WriteLine( )

Write, WriteLine (avec saut de ligne) sont des méthodes :

- Définies dans la classe Console
- Rangée dans le namespace System
- Statiques : leur appel doit être précédé de la classe
- Surchargées : plusieurs signatures



using System;  
//en début de fichier

```
Console.WriteLine("Bonjour !");
```

```
// ou  
System.Console.WriteLine("Bonjour !");
```

# Récupérer une saisie utilisateur : Console.ReadLine()

**ReadLine** est une méthode :

- Définie dans la classe Console
- Rangée dans le namespace System
- Statique : son appel doit être précédé de la classe
- Renvoie du texte !

## 4 Console, classe

▸ Console, méthodes

▸ Console, propriétés

		l'utilisateur a appuyé. La touche entrée s'ajoute à une ligne dans la fenêtre de console.
	ReadLine	Lit la ligne de caractères suivante à partir du flux d'entrée.

```
Console.WriteLine("Votre prénom : " );  
String prenom = Console.ReadLine();
```

```
using System;  
//en début de fichier
```

# Convertir et vérifier la saisie utilisateur : TryParse()

```
public static bool TryParse (string s, out int result);
```

- Méthode définie pour chaque structure correspondant à un type primitif.
- Essaie de convertir le 1er paramètre. Ici s
- Renvoie true si la conversion est possible, false sinon,
- Met le résultat de la conversion dans le 2eme paramètre ici result (passé par &)

```
Console.WriteLine("Quel est ton age ? ");  
String saisie = Console.ReadLine();  
int age;  
while (Int32.TryParse(saisie, out age) == false)  
{  
    Console.WriteLine("Vous devez saisir un entier.");  
    saisie = Console.ReadLine();  
}
```

out age ⇔ &age

# Manipuler une chaîne

- Une chaîne = un objet de classe String
- Objet = Variable complexe
- Classe = Type complexe
- Initialisation simplifiée identique aux variables simples

Exemple :

```
String prenom = "julien";
```

string alias de String : on peut donc omettre la majuscule

# String : une classe

- Pour en savoir plus : [documentation MSDN](#)

# Manipuler les propriétés d'un String

- Pour utiliser les propriétés : objet.Propriété

Version

.NET Framework 4.8

Rechercher

String

Constructeurs

> Champs

▼ Propriétés

Chars[]

**Length**

## String.Length Propriété

Espace de noms: [System](#)

Assembly: mscorlib.dll

Obtient le nombre de caractères de l'objet [String](#) actuel.

C# Copier

```
public int Length { get; }
```

### Exemple :

```
String prenom = "julien";
```

```
Console.WriteLine("Nombre de lettres :" + prenom.Length);
```

```
Console.WriteLine("1ere lettre :" + prenom.Chars[0] );
```



# Utiliser les méthodes d'un String

- Pour utiliser les méthodes : objet.methode( )
- Objet = variable complexe qui peut exécuter une méthode (traitement)

Exemple :

```
String prenom = "julien";
```



```
String prenomEnMaj = prenom.ToUpper();  
Console.WriteLine("prenom :" + prenomEnMaj);
```

Et non ToUpper (prenom) !!!



# Concaténation

- Opérateur : +

Exemple :

```
int age = 25;  
String msg = "J'ai " + age + " ans";
```

Remarque : on peut concaténer tout type de données...