INITIATION A LA PROGRAMMATION

C#

PROGRAMMATION ORIENTEE OBJET

PROGRAMMATION EVENEMENTIELLE

OJECTIFS DU COURS

- Apprendre la syntaxe du C#
- Apprendre à se servir de Visual Studio
- Apprendre les fondamentaux de la programmation orientée objet (OPP)
- Apprendre les bases de la programmation évènementielle

PROGRAMME DU COURS

- Syllabus
  - Introduction
  - Syntaxe C#
  - Introduction à la OPP
  - Principaux concepts OPP
  - Programmation évènementielle
- Projet personnel

DOCUMENTATION

- MICROSOFT DOCS :

Documentation Microsoft pour les utilisateurs finaux, les développeurs et les professionnels de l'informatique

https://docs.microsoft.com/en-us/?view=netframework-4.8

AIDE SUR INTERNET

- STACKOVERFLOW:

Forum pour développeurs

https://stackoverflow.com/

INTRODUCTION

I. INTRODUCTION

- Passer de Python à C#
  - **Quels sont les concepts de programmation vus avec Steve ?**
    - Variables
    - Types
    - Opérateurs
    - Conditions
    - Boucles
    - Collections
    - Méthodes
    - Classe / Objet
    - …
  - **Comment appliquer ces concepts à un autre langage de programmation ?**

PRESENTATION C#

PRESENTATION DU C#

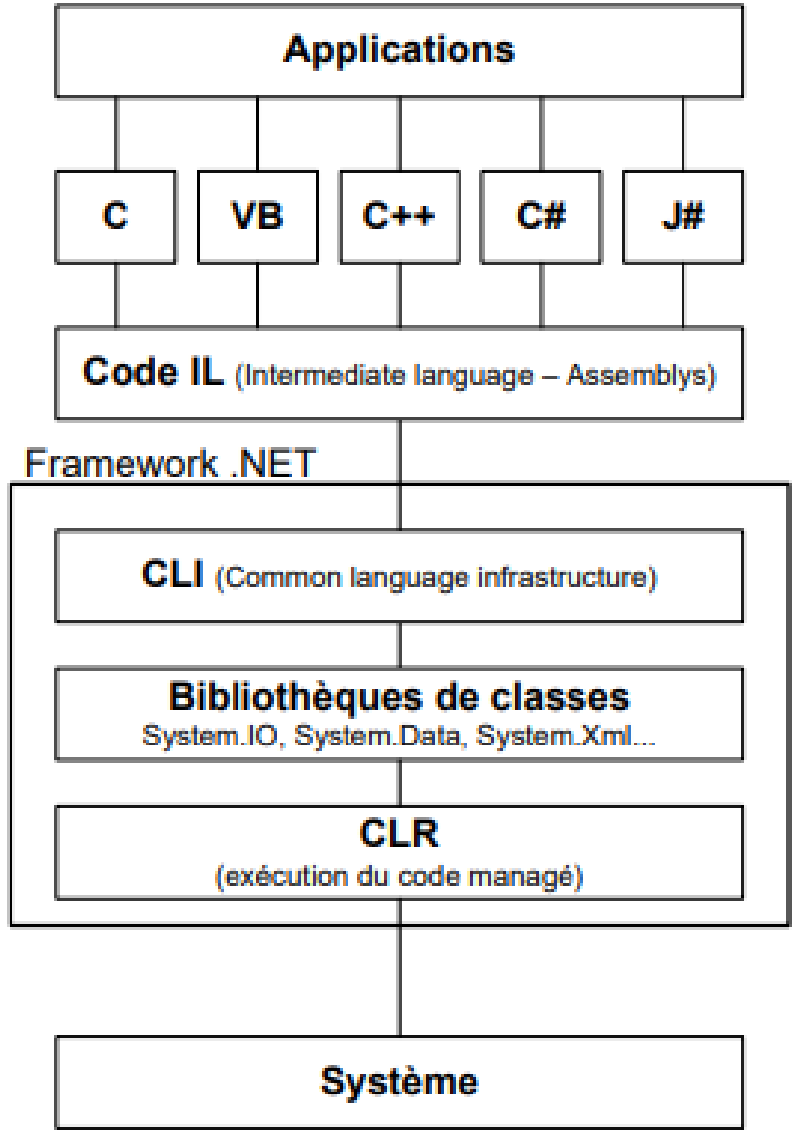
- Langage **Orienté-Objet** et **fortement typé**

- Langage **compilé**:

Code source est compilé par un logiciel, le *compilateur*, en un *code binaire* lisible par l'ordinateur et le système d'exploitation

⇒ programme directement exécuté sur l'ordinateur ➡ en général plus performant que langage interprété

- Langage pris en charge par la **Plateforme .NET**
  - Plateforme de développement généraliste
  - Utilisable par système d'exploitation Windows
  - Norme CLI => tous les langages compatibles CLI peuvent bénéficier des librairies/composants logiciels accessibles sur .NET



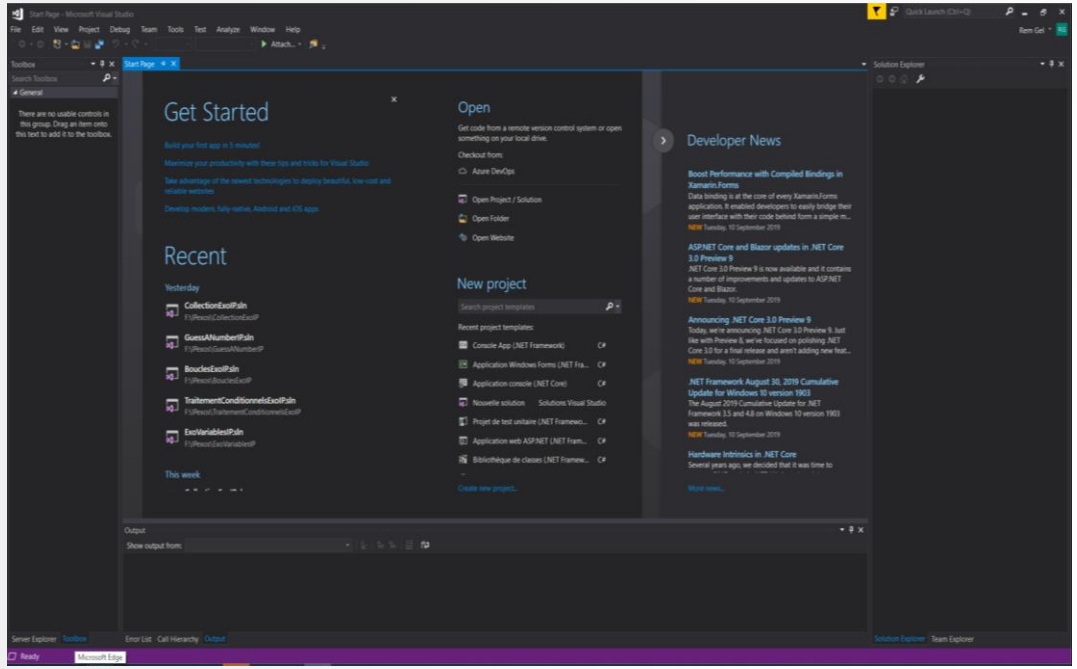
PRISE EN MAIN DE VISUAL STUDIO

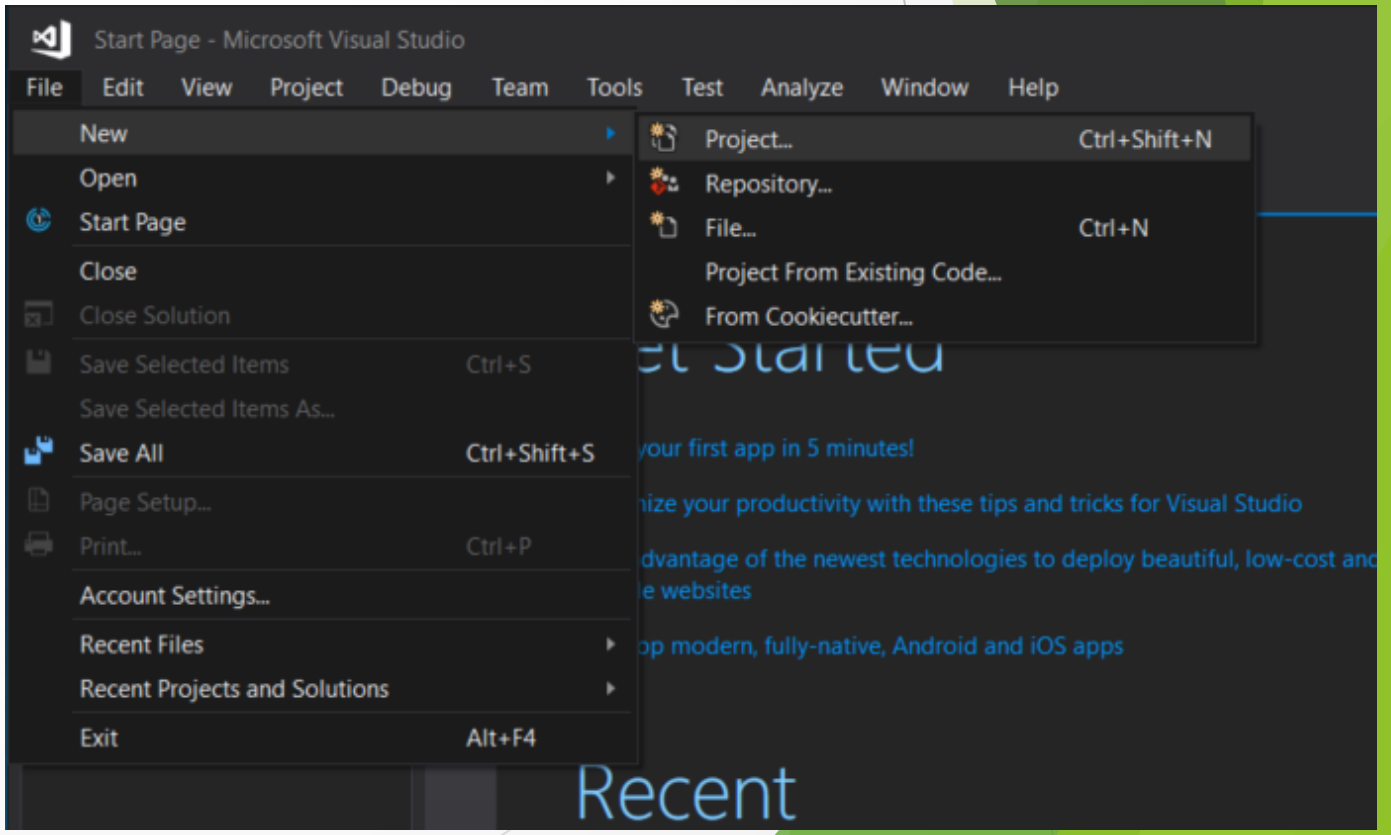
PRISE EN MAIN DE VISUAL STUDIO

- IDE / environnement de développement intégré
- ensemble complet d'outils de développement pour réaliser plus facilement des applications web, bureautique…

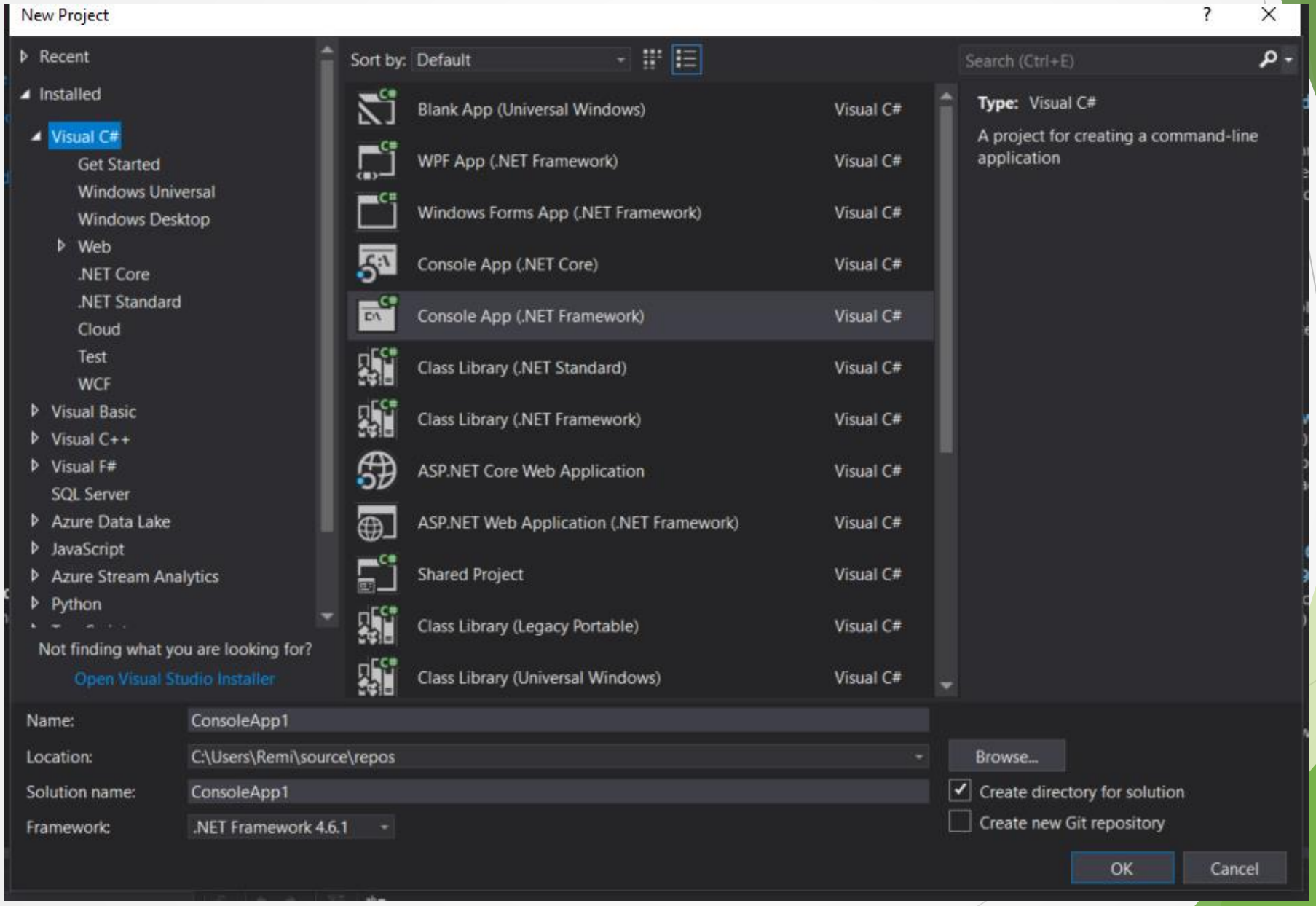
1. Ouvrons Visual Studio
2. Cliquons sur Nouveau Projet
3. Sélectionnons le langage « Visual C# » → « Windows »
4. Sélectionnons « Application Console »
5. Donnons lui un nom : « MaPremiereApplication »
6. Validons

PRISE EN MAIN DE VISUAL STUDIO



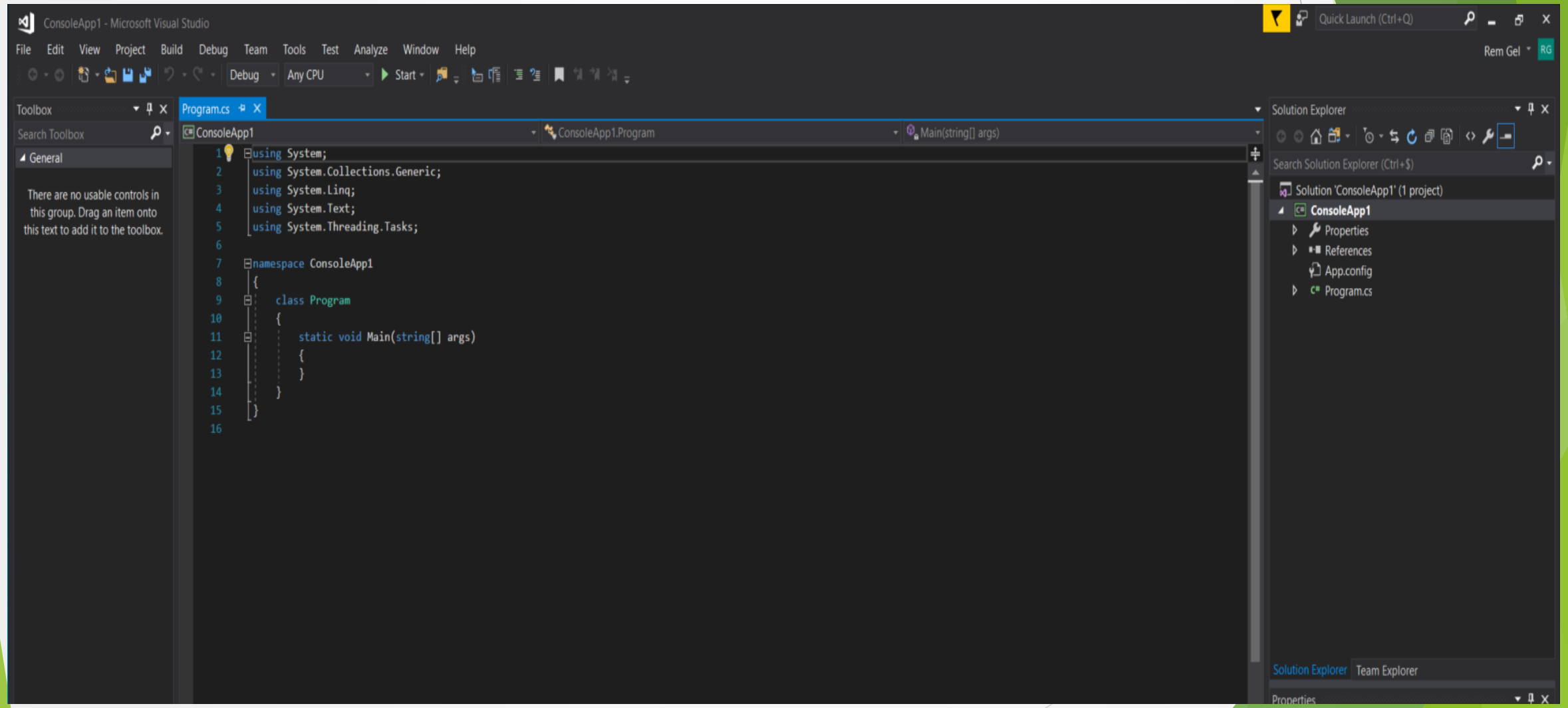


PRISE EN MAIN DE VISUAL STUDIO



PRISE EN MAIN DE VISUAL STUDIO

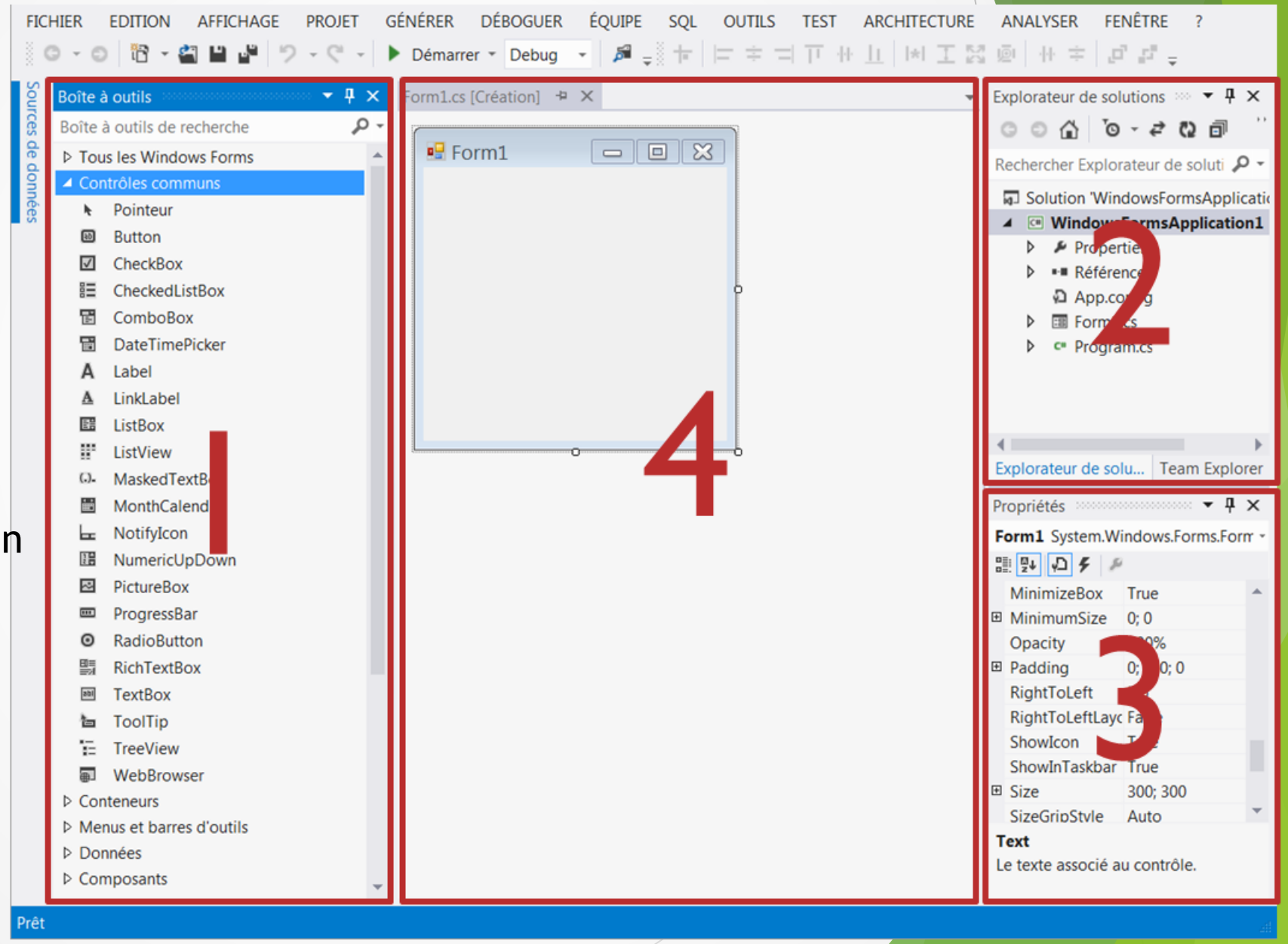
CONSOLE APP



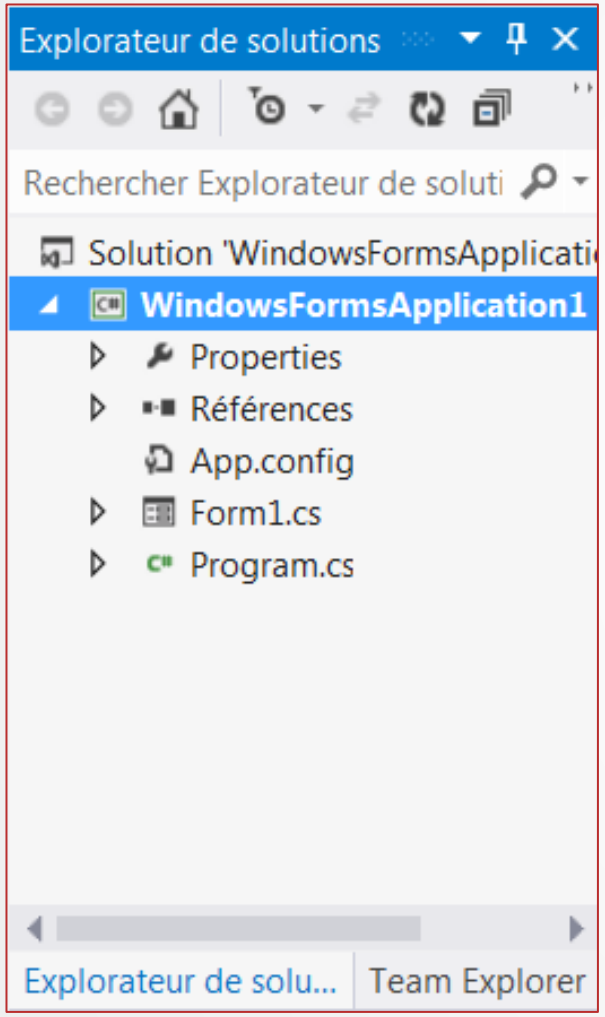
PRISE EN MAIN DE VISUAL STUDIO

WINDOWS FORM APP

1. La boite à outils
2. L'explorateur de solution
3. Les propriétés
4. La zone Design/Code



PRISE EN MAIN DE VISUAL STUDIO

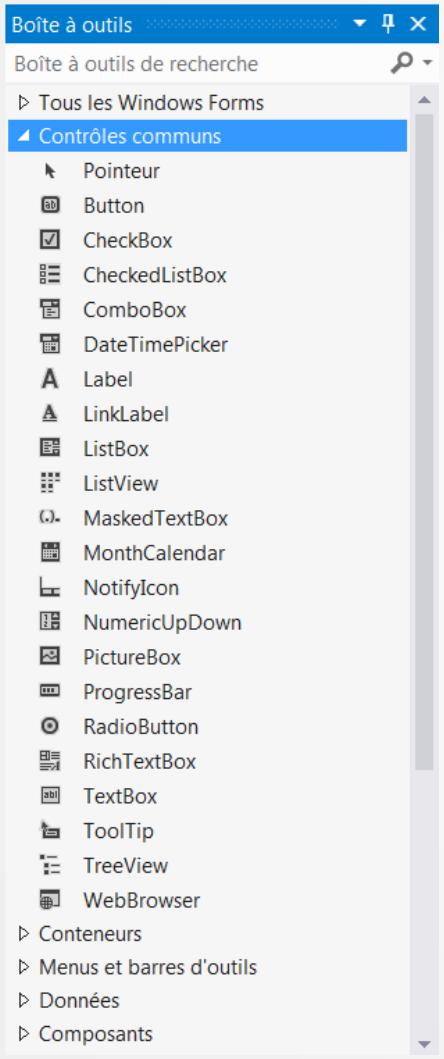


Solution => composée d'un ou plusieurs projets.

Avantage : facilite la possibilité de faire références aux différents projets composant la solution dans un projet spécifique

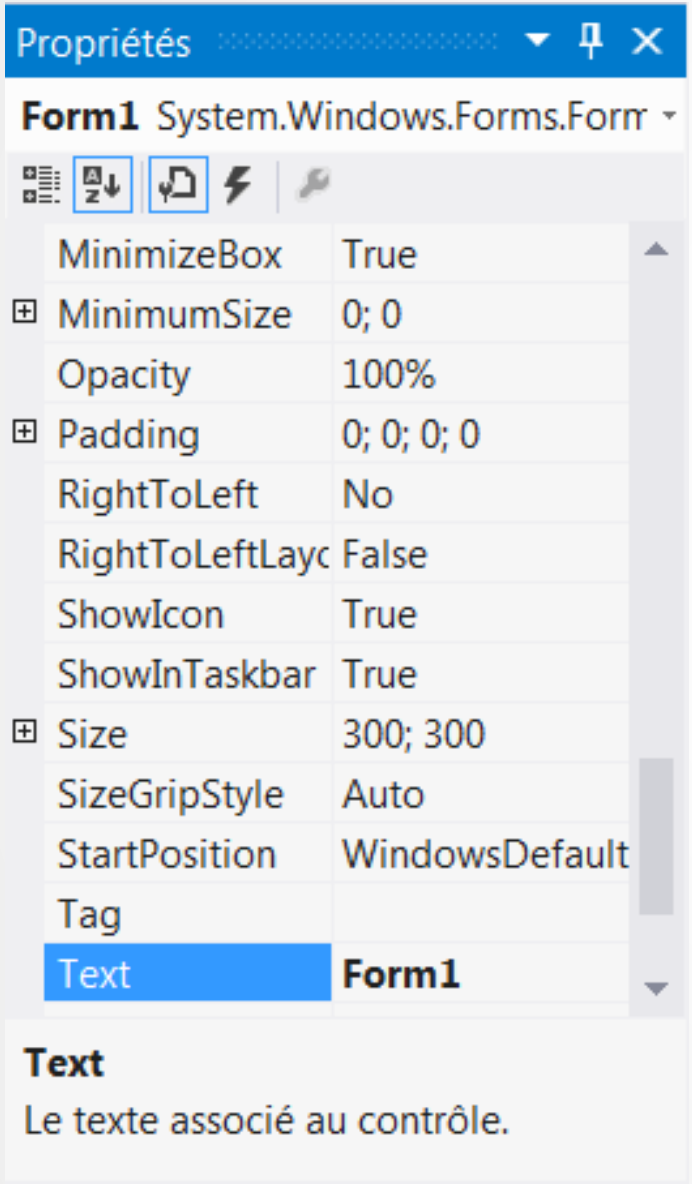
Possible de spécifier le projet de démarrage via les propriétés de la solution.

PRISE EN MAIN DE VISUAL STUDIO



Lorsque nous travaillons avec du design, la boite à outil nous permet d’obtenir tous les composants visuels pouvant être déposés sur notre interface.

PRISE EN MAIN DE VISUAL STUDIO



Affiche les propriétés de l'objet actuellement sélectionné.

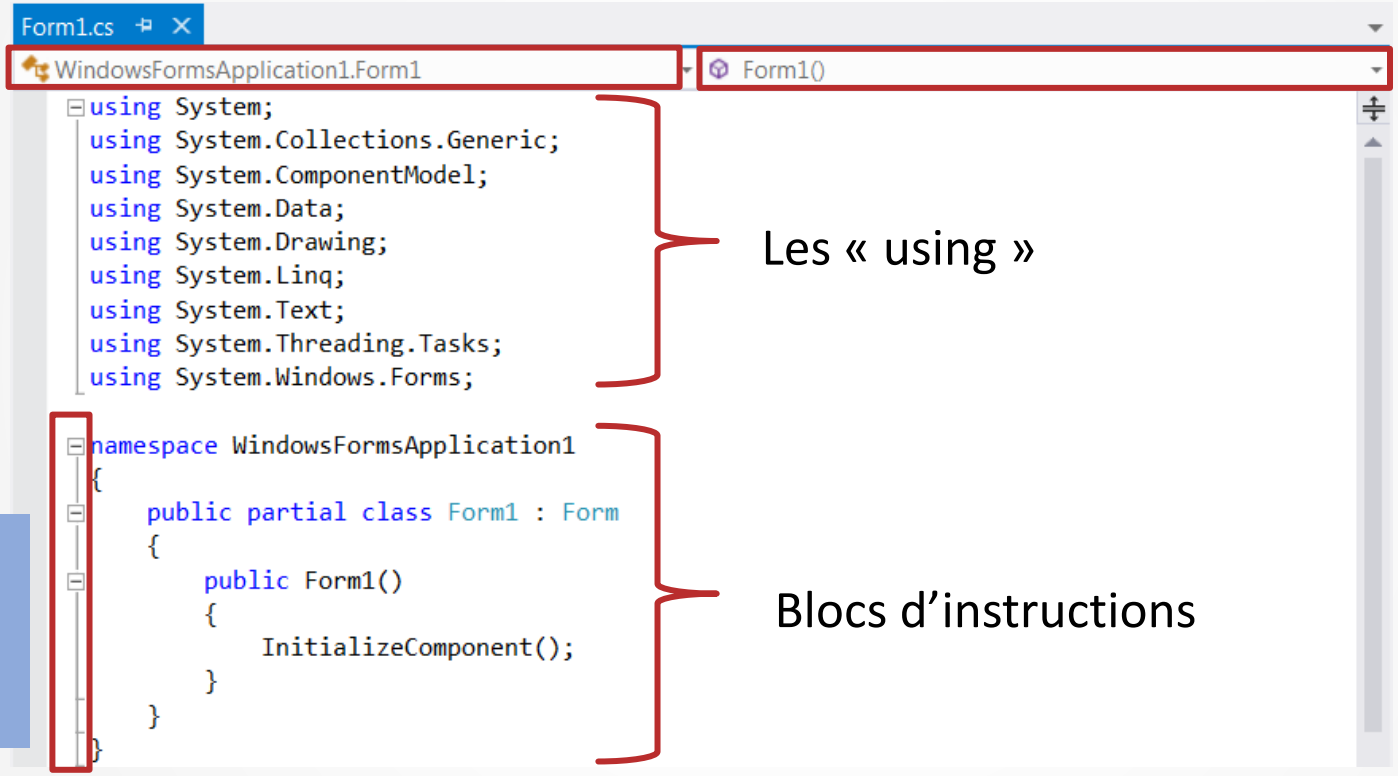
S'il s'agit d'un élément graphique, nous pouvons accéder à ses propriétés et à ses événements.

Nous pouvons les trier par ordre alphabétique ou regroupé par fonctionnalité.

PRISE EN MAIN DE VISUAL STUDIO

Liste des objets

Les méthodes de l'objet

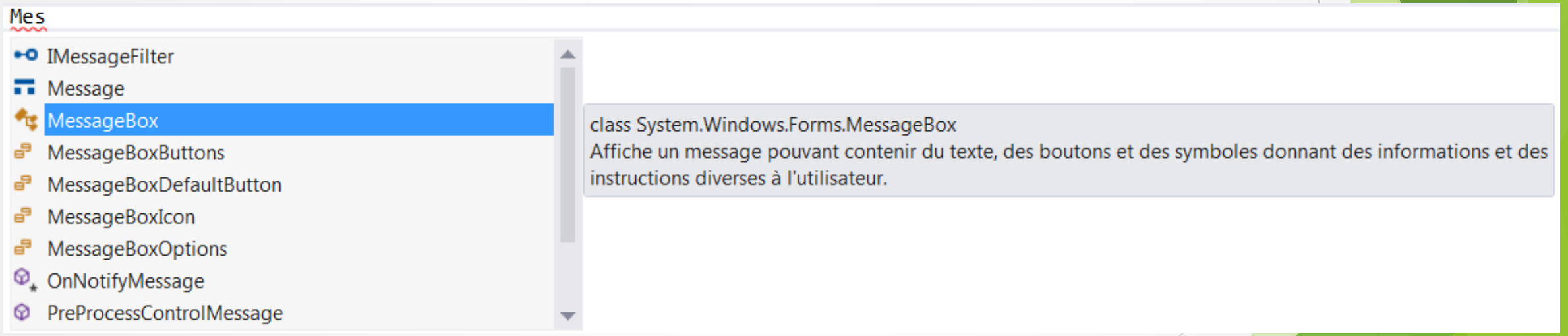


Possibilité de réduire certaine partie de code

PRISE EN MAIN DE VISUAL STUDIO

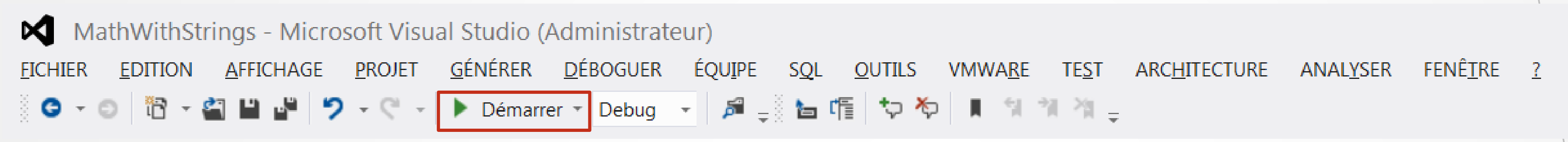
GESTION DU CODE - INTELLISENSE

- **Aide au code**
- **Auto-complétions**

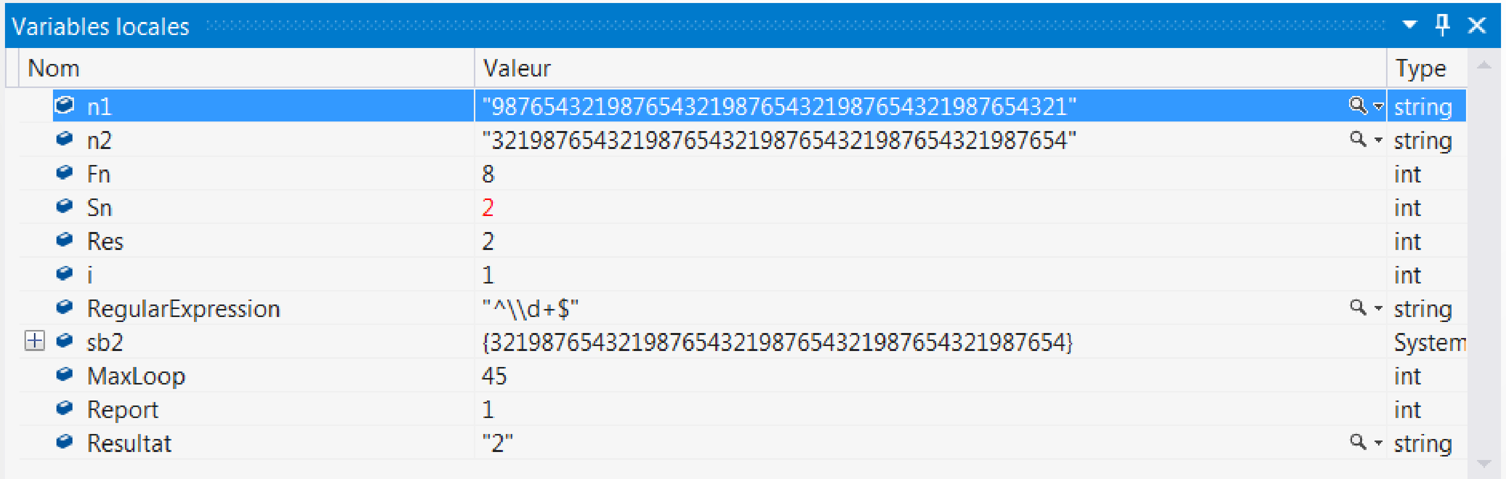


PRISE EN MAIN DE VISUAL STUDIO

DÉBOGGER SON APPLICATION



DEBUG => accès aux types/valeurs en temps réel des **variables locales**
Quand une variable change de valeur, la valeur de celle-ci est indiquée en rouge.



Variables locales

Nom	Valeur	Type
n1	"9876543219876543219876543219876543219876554321"	string
n2	"3219876543219876543219876543219876543219876554"	string
Fn	8	int
Sn	2	int
Res	2	int
i	1	int
RegularExpression	"^\\d+$"	string
sb2	{3219876543219876543219876543219876543219876554}	System
MaxLoop	45	int
Report	1	int
Resultat	"2"	string

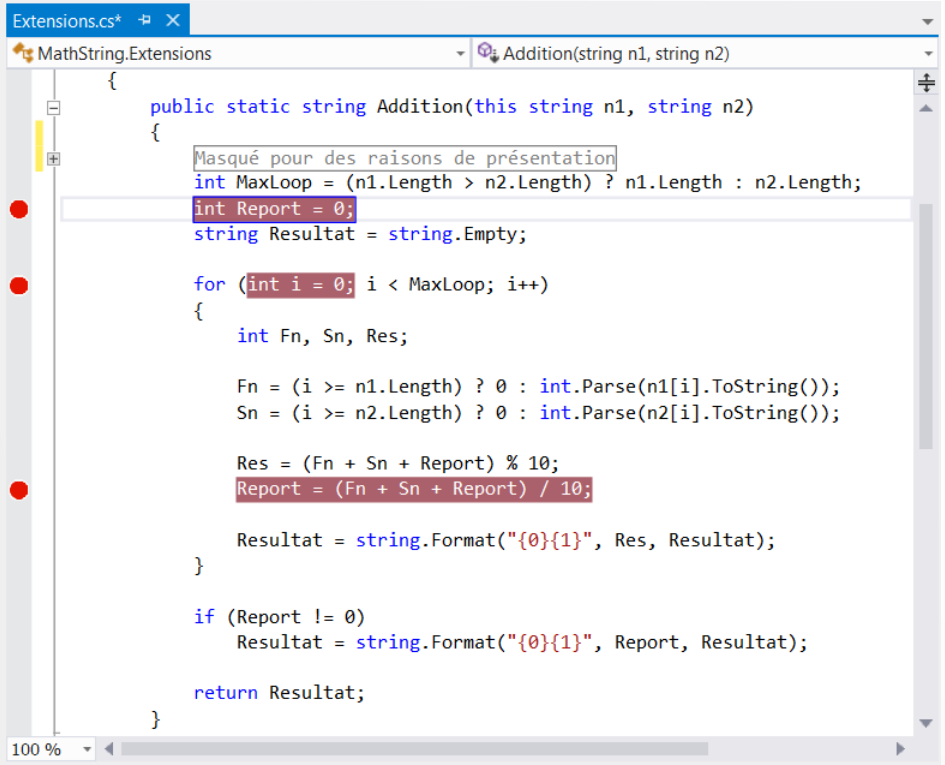
F5 : Exécution en mode débogage

CTRL + F5 : Exécution sans débogage

F11 : Exécution en mode débogage pas à pas

PRISE EN MAIN DE VISUAL STUDIO

DÉBOGGER SON APPLICATION - Breakpoints



Les breakpoints se placent sur la ligne où on désire arrêter le débogage, en cliquant dans la bande grise vertical.

Par défaut, chaque fois que la ligne de code sera atteinte, le compilateur attendra notre décision.

En appuyant sur « F5 », il continuera jusqu'au prochain breakpoint

En appuyant sur « F11 », il continuera pas à pas.

SYNTAXE C#

VARIABLES

2. SYNTAXE DU C# - VARIABLES

```
int nom_variable = 5;
```

- Comment se définie une variable en C#?
- ÉTAPES :
  - DECLARATION
  - AFFECTATION
- **PORTEE DES VARIABLES**

PORTEE DES VARIABLES

```
using System;

namespace CoursCSharpFondements
{
    class Exemple
    {
        //Déclaration de la variable membre X
        int X;

        public void MaMethode()
        {
            //Affectation de la valeur 7 à la variable X
            X = 7;
            //Déclaration de la variable locale Y et initialisation à la valeur 5
            int Y = 5;

            for (int i = 0; i < 10; i++)
            {
                //code
            } // fin de la portée de la variable i
        } // fin de la portée de la variable Y
    } // fin de la portée de la variable X
}
```

TYPES

2. SYNTAXE DU C# - TYPES prédéfinis

Principaux types de variables en C#

TYPE	VALEURS	.NET TYPE
bool	true, false	System.Boolean
char	0000–FFFF	System.Char
string	Texte (ensemble de char)	System.String
double	±5.0 × 10(e−324) to ±1.7 × 10(e308)	System.Double
float	±1.5 × 10(e−45) to ±3.4 × 10(e38)	System.Single
int	−2,147,483,648 to 2,147,483,647	System.Int32

ENTRÉE / SORTIE

2. SYNTAXE DU C# - OPERATIONS ENTRÉE/SORTIE

- **LIRE** - **récupérer info du monde extérieur**

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

2. SYNTAXE DU C# - OPERATIONS ENTRÉE/SORTIE

- **ECRIRE** - **communiquer info au monde extérieur**

```
Console.WriteLine("Bienvenue");
```

EXERCICE

Bonjour Bonjour

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander nom de l'utilisateur
  - Afficher "Bonjour" puis le nom de l'utilisateur dans la Console

OPERATEURS

2. SYNTAXE DU C# - OPERATEURS

OPÉRATEURS ARITHMÉTIQUES

+	Addition
-	Soustraction
*	Multiplication
/	Division et division entière
%	Modulo (reste d'une division d'entiers)

2. SYNTAXE DU C# - OPERATEURS

OPÉRATEURS DE COMPARAISONS

Les opérateurs sur les valeurs		
==	Égale	i == 5
!=	Différent	i != 5
<	Plus petit que	i < 5
<=	Plus petit ou égal	i <= 5
>	Plus grand que	i > 5
>=	Plus grand ou égal	i >= 5

2. SYNTAXE DU C# - OPERATEURS

OPÉRATEURS LOGIQUES

Les opérateurs logiques	
!	Négation
&&	Et
||	Ou
^	Ou Exclusif

2. SYNTAXE DU C# - OPERATEURS

TABLES DE VÉRITÉS

ET

A ∧ B	Vrai	Faux
Vrai	V	F
Faux	F	F

OU

A v B	Vrai	Faux
Vrai	V	V
Faux	V	F

NON

	¬A
Vrai	F
Faux	V

EXERCICE

Ton numéro

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir une nombre entier
  - Afficher le nombre entier dans la Console

EXERCICE

Switch Value

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Déclarer deux variables entières A et B
  - Trouver un moyen d'inverser le contenu de ces deux variables
  - Afficher le résultat dans la Console

EXERCICE

Permutation de 5 variables

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Déclarer cinq variables entières comme suit:

    A = 1, B = 2, C = 3, D = 4, E = 5
  - Trouver un moyen de permuter le contenu de ces variables pour obtenir le résultat suivant:

    A = 4, B = 3, C = 5, D = 1, E = 2
  - Afficher le résultat dans la Console

EXERCICE

Somme et quotient

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir deux variables entières
  - Calculer la somme et le quotient de ces deux nombres
  - Afficher le résultat dans la Console

EXERCICE

Surface d'un rectangle

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir la longueur et la largeur d'un rectangle (float)
  - Calculer la surface (float)
  - Afficher le résultat dans la Console

EXERCICE

Camion et cartons

But : ranger un maximum de cartons pesant k kilos dans un camion pouvant contenir M kilos de marchandise

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir k et M (floats)
  - Calculer le nombre de cartons qui peuvent rentrer dans le camion (entier)
  - Afficher le résultat dans la Console

2. SYNTAXE DU C# - STRUCTURES CONDITIONNELLES

Structures conditionnelles :

- Insérer des branchements & évaluer des conditions (true or false):
  - Si certaines conditions sont satisfaites
  - En fonction de la valeur d'une expression
- Prendre des décisions dans le code

CONDITIONS

2. SYNTAXE DU C# - STRUCTURES CONDITIONNELLES

- **SI… SINON => IF…ELSE**

```
static void Main(string[] args)
{
    bool condition = true;

    if (condition)
    {
        //Code à exécuter si la condition renvoit true
    }
    else
    {
        //Code à exécuter si la condition renvoit false
    }

    Console.ReadLine();
}
```

2. SYNTAXE DU C# - STRUCTURES CONDITIONNELLES

- **SI… SINON SI… SINON => IF…ELSE IF…ELSE**

```
static void Main(string[] args)
{
    int i;

    if (int.TryParse(Console.ReadLine(), out i))
    {
        if (i > 5)
        {
            Console.WriteLine("i plus grand que 5");
        }
        else if (i < 5)
        {
            Console.WriteLine("i plus petit que 5");
        }
        else
        {
            Console.WriteLine("i égale à 5");
        }
    }

    Console.ReadLine();
}
```

2. SYNTAXE DU C# - STRUCTURES CONDITIONNELLES

- **SELON QUE => SWITCH**

```
static void Main(string[] args)
{
    int i;

    if (int.TryParse(Console.ReadLine(), out i))
    {
        switch(i)
        {
            case 1:
                Console.WriteLine("i vaut 1");
                break;
            case 2:
                Console.WriteLine("i vaut 2");
                break;
            case 3:
                Console.WriteLine("i vaut 3");
                break;
            default :
                Console.WriteLine("i ne vaut ni 1, ni 2, ni 3");
                break;
        }
    }

    Console.ReadLine();
```

EXERCICE

Pair ou Impair

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir une nombre entier
  - Déterminer si le nombre est pair ou impair
  - Afficher le résultat dans la Console

EXERCICE

Majorité

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir son âge
  - Déterminer si l'utilisateur est majeur
  - Afficher le résultat dans la Console

EXERCICE

Admis!

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir une nombre (float)
  - Afficher dans la Console:
    - "Ajourné" si la note est inférieure à 8
    - "Rattrapage" si la note est entre 8 et 10
    - "Admis" si la note est supérieure à 10

EXERCICE

Plus petit et plus grand

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir 5 nombres
  - Afficher le plus petit et le plus grand nombre dans la Console

EXERCICE

Chiffre identique

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir trois nombres
  - Vérifier si des chiffres sont identiques
  - Afficher le résultat dans la Console

EXERCICE

Mini calculatrice

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir deux valeurs A et B (int)
  - Demander à l'utilisateur de saisir un opérateur(+, -, *, /)
  - Afficher le résultat de l'opération "A operateur B" dans la Console

EXERCICE

Année Bissextile

- Créer une nouvelle application Console
- Dans la méthode Main():
  - Demander à l'utilisateur de saisir une année (int)
  - Déterminer s'il s'agit d'une année bissextile
  - Afficher le résultat de l'opération dans la Console

BOUCLES

2. SYNTAXE DU C# - STRUCTURES ITERATIVES

BOUCLES:

- Répéter des instructions dans nos algorithms
- Sur la base de conditions précises (true or false)

2. SYNTAXE DU C# - STRUCTURES ITERATIVES

TANT QUE…=> WHILE

- TANT QU'IL Y A DES BALLES, LANCER DES BALLES

```
while (conditions)
{
    instructions;
}
```

- **Instructions répétées tant que la condition est vérifiée**

2. SYNTAXE DU C# - STRUCTURES ITERATIVES

- **FAIRE…TANT QUE…=> DO… WHILE**

  - FAIRE UN CALCUL TANT QUE L'UTILISATEUR SOUHAITE FAIRE UN CALCUL

```
do {
    instruction;
} while (conditions);
```

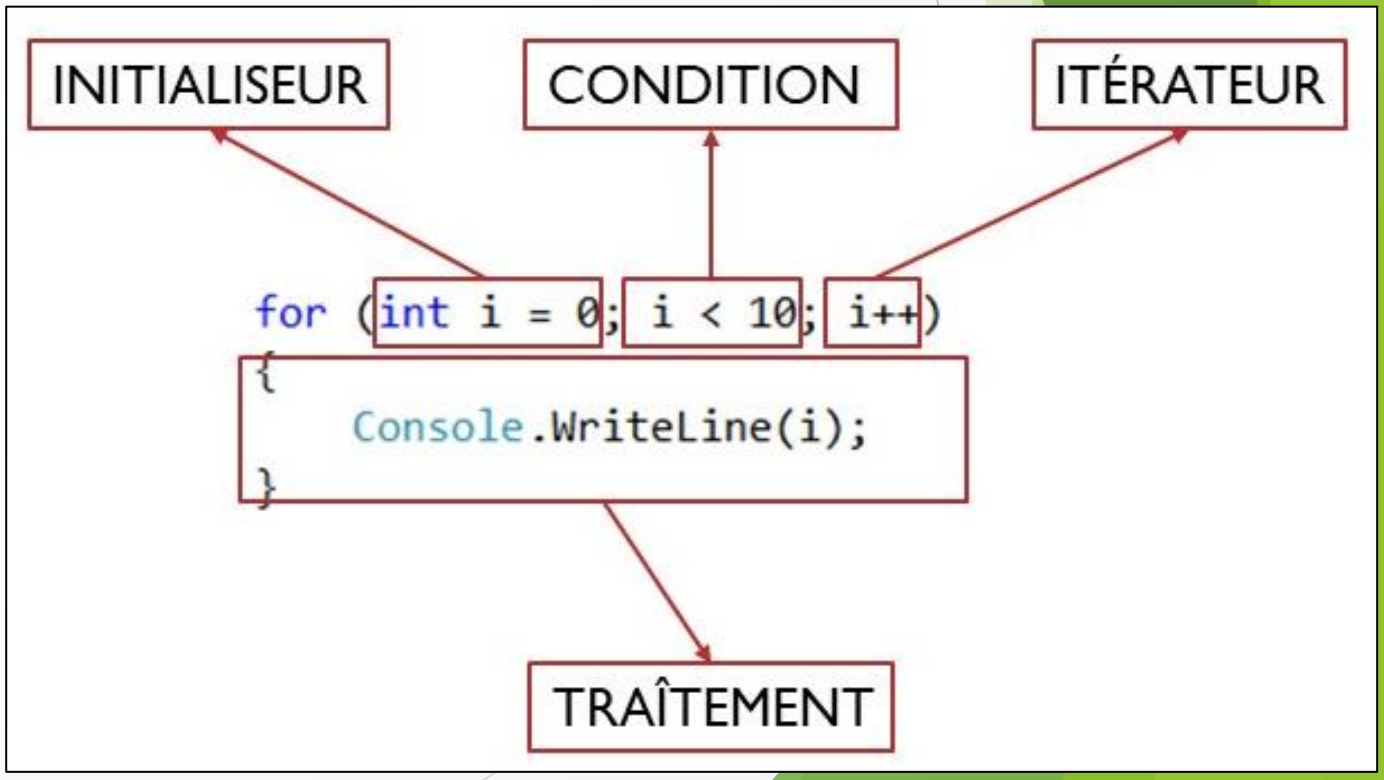
- **Instructions effectuées une première fois avant d'évaluer si il est necessaire de les réitérer**

2. SYNTAXE DU C# - STRUCTURES ITERATIVES

POUR…=> FOR

- PARCOURIR UNE COLLECTION ET EFFECTUER DES INSTRUCTIONS

```
for (initialisation; conditionArret; pasIncrementation)
{
    instruction;
}
```



2. SYNTAXE DU C# - STRUCTURES ITERATIVES

POUR CHAQUE…=> FOREACH

- POUR CHAQUE ELEMENT D'UNE COLLECTION, EFFECTUER DES INSTRUCTIONS

```
foreach(<type> <variable> in <tableau>)
                <instructions>
```

2. SYNTAXE DU C# - STRUCTURES ITERATIVES

DIFFERENCE FOR / FOREACH :

- FOREACH : on ne **peut pas** modifier les éléments de la liste

- FOR : on **peut** modifier les éléments de la liste

COLLECTIONS

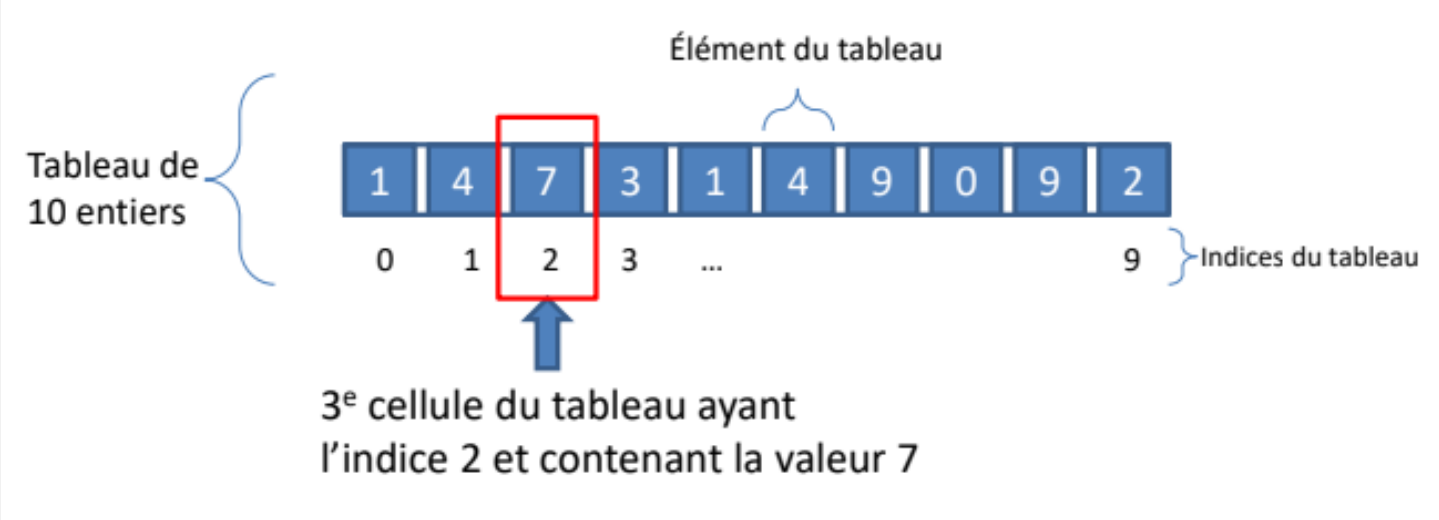
2. SYNTAXE DU C# - COLLECTIONS

- TABLEAU:
  - Collection de données
  - Ensemble de données du même type
  - Taille fixe

- **DECLARATION ET INITIALISATION:**

```
type[] nomTableau = new type[tailleTableau];
```

2. SYNTAXE DU C# - COLLECTIONS



2. SYNTAXE DU C# - COLLECTIONS

- LISTE:
  - Collection de données
  - Ensemble de données du même type
  - Taille FLEXIBLE

- **DECLARATION ET INITIALISATION:**

```
List<type> nom_liste = new List<type>() ;
```

EXERCICE

Tables de multiplication

- Créer une nouvelle application Console
- À l'aide d'une boucle, afficher la table de multiplication par 2.

EXERCICE

Peupler un tableau

- Créer une nouvelle application Console
- Créer un tableau stockant avec une boucles les chiffres de 0 à 10, puis afficher contenu du tableau dans la console,
- Afficher le contenu de ce tableau une fois qu'il est rempli dans la Console

EXERCICE

Peupler un tableau puis une liste

- Créer une nouvelle application Console
- Initialiser une liste d'entiers avec les valeurs 1, 15, 65, 12, 99, 45, 126 avec la technique de l'object initializer
- Afficher le contenu du tableau et de la liste dans la Console

EXERCICE

Ton carré

- Créer une nouvelle application Console
- Demander à l'utilisateru de saisir une valeur n
- Afficher un carré dans la Console selon input (nombre de lignes, nombre de colonnes)

EXERCICE

Chercher dans le tableau et supprimer

- Réalisez un algorithme dans lequel nous devons rechercher une valeur (entrée par l'utilisateur) dans un tableau d'entiers.
- Si on trouve cette valeur, nous devons d’abord afficher sa position puis la supprimer du tableau.

EXERCICE

GuessANumber (WindowsForm)

- Créer une nouvelle application Console
- L'ordinateur choisi un nombre aléatoirement entre 1 et 100
- L'utilisateur est invité à entrer un nombre et l'algorithme nous répond "C'est plus" ou "C'est moins".
- Lorsqu'on a trouvé le bon nombre, l'algorithme affiche « C’est gagné » et le nombre de tentatives effectuées pour trouver le résultat.