SIO1 C#

Table des matières

[I. Un algorithme 2](#_Toc494271214)

[II. Un programme 2](#_Toc494271215)

[1. Qu'est-ce que le C# ? 2](#_Toc494271216)

[2. Les langages compilés 3](#_Toc494271217)

[3. Les langages interprétés 3](#_Toc494271218)

[III. L'environnement de développement 4](#_Toc494271219)

[IV. Premier projet 4](#_Toc494271220)

[1. Création du projet 4](#_Toc494271221)

[2. L’environnement de développement et du code généré 4](#_Toc494271222)

[3. L'exécution du projet 6](#_Toc494271223)

[V. La syntaxe générale du C# 7](#_Toc494271224)

[1. Le caractère de terminaison de ligne 7](#_Toc494271225)

[2. Les commentaires 8](#_Toc494271226)

[3. La complétion automatique 8](#_Toc494271227)

[VI. Les variables 9](#_Toc494271228)

[1. Qu'est-ce qu'une variable 10](#_Toc494271229)

[2. Déclaration des variables 10](#_Toc494271230)

[3. Les différents types de variables 11](#_Toc494271231)

[4. Affectations, opérations, concaténation 12](#_Toc494271232)

[5. La saisie 13](#_Toc494271233)

# Un algorithme

Un ordinateur est une machine électronique programmable servant au traitement de l’information codée sous forme binaire, c’est-à-dire sous forme de tout ou rien (soit le courant passe, soit il ne passe pas).

Un programme est un assemblage et un enchaînement d’instructions élémentaires écrit dans un langage de programmation, et exécuté par un ordinateur afin de traiter les données d’un problème et renvoyer un ou plusieurs résultats.

L'exécution d'un programme est faite de manière séquentielle : de haut en bas et toutes les lignes.

Un algorithme représente l'enchaînement des actions (instructions) nécessaires pour faire exécuter une tâche à un ordinateur (résoudre un problème).

Un algorithme s'écrit le plus souvent en pseudo-langage de programmation (appelé langage algorithmique).

# Un programme

Un ordinateur peut traiter autant de problèmes différents grâce aux actions de base qu'il sait réaliser, il est possible en les assemblant de façon pertinente, de résoudre la plupart des problèmes concernant le traitement de l'information. Il suffit de lui indiquer l'ordre dans lequel il faut qu'il effectue ces actions basiques et avec quelles données. Ces ordres élémentaires sont appelés instructions et sont rassemblées au sein d'un programme.

## Qu'est-ce que le C# ?

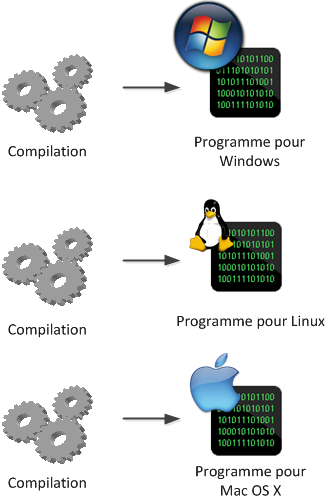
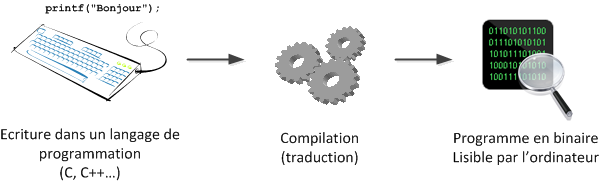
Le C# est un langage de programmation créé par Microsoft en 2002.

Le C# est un langage dont la syntaxe ressemble un peu au C++ ou au Java qui sont d’autres langages de programmation très populaires. Le C# est le langage phare de Microsoft. Il fait partie d’un ensemble plus important. Il est en fait une brique de ce qu’on appelle le « **Framework .NET** ».

Un framework (dans notre cas, le framework .NET de Microsoft) est une sorte d'immense bibliothèque informatique contenant des outils qui vont faciliter la vie du développeur. Le framework .NET est compatible avec le Visual Basic et d'autres langages tels que le C#, le J#.

Le framework .NET est un framework créé par Microsoft en 2002, en même temps que le C#, qui est principalement dédié à la réalisation d’applications fonctionnant dans des environnements Microsoft. Nous pourrons par exemple réaliser des programmes qui fonctionnent sous Windows, ou bien des sites web ou encore des applications qui fonctionnent sur téléphone mobile, etc.

## Les langages compilés



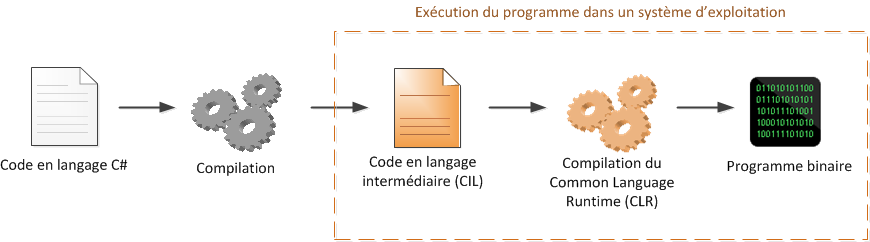
La compilation génère des instructions en binaire, donc directement compréhensible pour la machine. Il faut donc une compilation pour chaque système d’exploitation.

## Les langages interprétés

Les langages récents, comme le C# et le Java, résolvent ce problème de compatibilité tout en ajoutant de nombreuses fonctionnalités appréciables au langage, ce qui permet de réaliser des programmes beaucoup plus efficacement.

La compilation en C# ne donne pas un programme binaire, contrairement au C et au C++. Le code C# est en fait transformé dans un langage intermédiaire (appelé CIL ou MSIL) que l'on peut ensuite distribuer à tout le monde. Ce code, bien sûr, n'est pas exécutable lui-même, car l'ordinateur ne comprend que le binaire.

Regardez bien ce schéma pour comprendre comment cela fonctionne :



***WIKI :***

***Common Language Runtime*** (CLR) est le nom choisi par [Microsoft](https://fr.wikipedia.org/wiki/Microsoft) pour le composant de [machine virtuelle](https://fr.wikipedia.org/wiki/Machine_virtuelle) du [*framework .NET*](https://fr.wikipedia.org/wiki/Framework_.NET). Il s'agit de l'implémentation par Microsoft du standard [*Common Language Infrastructure*](https://fr.wikipedia.org/wiki/Common_Language_Infrastructure) (CLI) qui définit l'[environnement d'exécution](https://fr.wikipedia.org/wiki/Environnement_d%27ex%C3%A9cution) des codes de programmes. Le CLR fait tourner une sorte de [bytecode](https://fr.wikipedia.org/wiki/Bytecode" \o "Bytecode) nommé [*Common Intermediate Language (CIL)*](https://fr.wikipedia.org/wiki/Common_Intermediate_Language). Le compilateur à la volée transforme le code CIL en code natif spécifique au [système d'exploitation](https://fr.wikipedia.org/wiki/Syst%C3%A8me_d%27exploitation). Le CLR fonctionne sur des systèmes d'exploitation [Microsoft Windows](https://fr.wikipedia.org/wiki/Microsoft_Windows).

Pour linux il existe Mono comme CLR pour C#(par contre pas toutes les fonctionnalités)

Cela offre beaucoup de souplesse au développeur. Le code en langage intermédiaire (CIL) peut être distribué à tout le monde. Il suffit d'avoir installé la machine CLR sur son ordinateur, qui peut alors lire les programmes en C# et les compiler "à la volée" en binaire. Avantage : le programme est toujours adapté à l'ordinateur sur lequel il tourne.

# L'environnement de développement

Un environnement de développement intégré (EDI ou IDE en anglais pour Integrated Development Environment) est un programme regroupant un ensemble d'outils pour le développement de logiciels.

Microsoft a créé un IDE « **Visual Studio** », qui rassemble Visual Basic, Visual C++, Visual C#.

Nous allons donc utiliser **Visual Basic.**

# **Premier projet**

## **Création du projet**

**Nous allons créer un projet C# en mode Console**

**L’application Console** est la forme de projet pouvant produire une application exécutable la plus simple. Elle permet de réaliser un programme qui va s’exécuter dans la console noire qui ressemble à une fenêtre ms-dos, nous resterons concentrés sur la logique du programme et non sur le visuel d’une interface graphique.

## L’environnement de développement et du code généré

**Allons dans le répertoire où vous avez enregistré votre application, nous pouvons constater que Visual Studio a créé un répertoire MaPremiereApplication, c’est le répertoire pour la solution.**

**Dans ce répertoire, nous remarquons notamment un fichier MaPremiereApplication.sln.**

**Si vous ne voyez pas l'extension du fichier, il faut configurer votre explorateur de fichier pour afficher les extensions connues.**

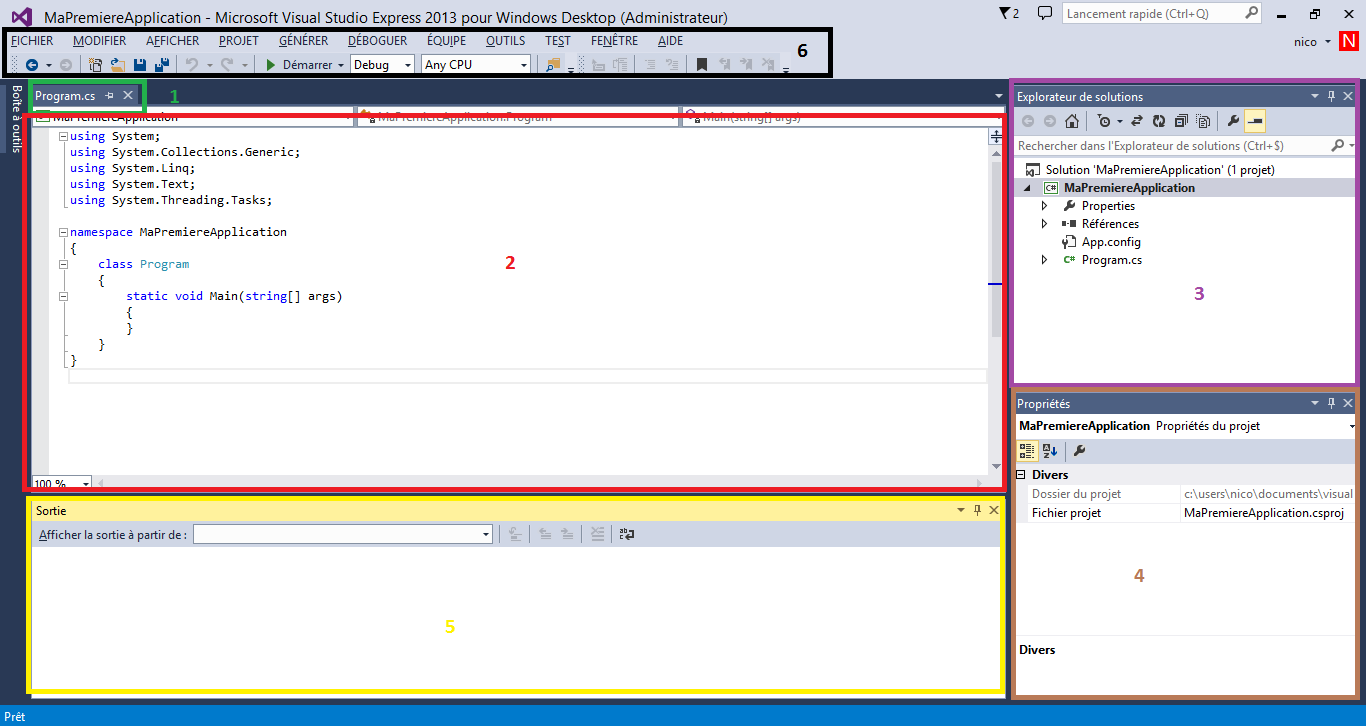
**C’est ce qu'on appelle**le fichier de solution**; il s’agit juste d’un container de projets qui va nous permettre de visualiser nos projets dans Visual Studio.**

Le fichier décrivant un projet écrit en C# est préfixé par cs (csproj).

Penchons nous sur le fichier Program.cs. Les fichiers dont l’extension est .cs contiennent du code C#, c’est dans ce fichier que nous allons commencer à taper nos premières lignes de code.

À noter que l'ensemble des fichiers contenant des instructions écrites dans un langage de programmation est appelé le « code source ». Par extension, le « code » correspond à des instructions écrites dans un langage de programmation.

Si nous retournons dans l’interface de Visual Studio, nous pouvons retrouver quelque chose comme ça :



La zone verte numéro 1 contient les différents fichiers ouverts sous la forme d’un onglet. On voit que par défaut, Visual Studio Express nous a créé et ouvert le fichier Program.cs.

Dans la zone rouge numéro 2, c’est l’éditeur de code. Il affiche le contenu du fichier ouvert. Nous voyons des mots que nous ne comprenons pas encore. C’est du code qui a été automatiquement généré par Visual Studio Express. Nous pouvons observer que les mots sont de différentes couleurs. En effet, l’éditeur Visual Studio Express possède ce qu’on appelle une **coloration syntaxique**, c'est-à-dire que certains mots clés sont colorés d’une couleur différente en fonction de leur signification ou de leur contexte afin de nous permettre de nous y retrouver plus facilement.

La zone numéro 3 en violet est l’explorateur de solutions, c’est ici que l’on voit le contenu de la solution sur laquelle nous travaillons en ce moment. En l’occurrence, il s’agit de la solution « MaPremiereApplication » qui contient un unique projet « MaPremiereApplication ». Ce projet contient plusieurs sous éléments :

* Properties : contient des propriétés de l’application, on ne s’en occupe pas pour l’instant
* Références : contient les références de l’application, on ne s’en occupe pas pour l’instant
* App.config : contient de la configuration, on ne s'en occupe pas pour l'instant
* Program.cs est le fichier qui a été généré par Visual Studio et qui contient le code C#.

La zone 4 en brun est la zone contenant les propriétés de ce sur quoi nous travaillons en ce moment. Ici, nous avons le curseur positionné sur le projet, il n’y a pas beaucoup d’informations excepté le nom du fichier de projet. Nous aurons l’occasion de revenir sur cette fenêtre plus tard.

La zone 5 en jaune n’est pas affichée au premier lancement, elle contient la liste des erreurs, des avertissements et des messages de notre application. Nous verrons comment l’afficher un peu plus bas. C'est la fenêtre de sortie.

La zone 6 en noir est la barre d'outils, elle possède plusieurs boutons que nous pourrons utiliser, notamment pour exécuter notre application.

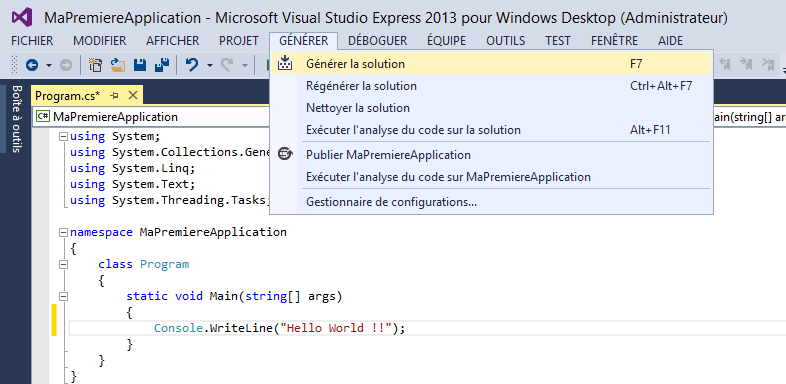
**Application** : Ecrire du texte dans la console

Concepts abordés : le framework.Net est sutructurée en classe (une classe regroupe des outils sur une thème commun ). Ces outils peuvent être une propriété : Console.color ; ou une méthode. Il existe 2 types de méthodes : les procédures qui effectuent un traitement et ne retournent rien ( Console.ReadLine() ) et les fonctions qui effectuent un traitement et nous renvoient une valeur (Console.Write(‘’bjr’’) ) . De plus une méthode peut avoir besoin de données pour fonctionner : ce sont des paramètres ou arguments.

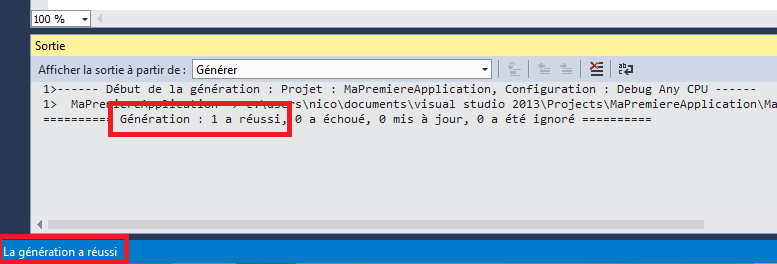
## L'exécution du projet

La première chose à faire est de transformer le langage C# que nous venons d’écrire en programme exécutable. Cette phase s’appelle la « génération de la solution » sous Visual Studio. On l’appelle souvent la « **compilation**» ou en anglais le « **build**».

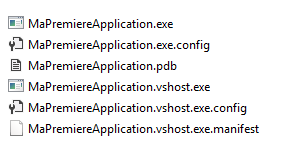
Allez dans le menu Générer  et cliquez sur « Générer la solution  » :



Visual Studio lance alors la génération de la solution et on voit dans la barre des taches en bas à gauche qu’il travaille jusqu’à nous indiquer que la génération a réussi :



Si nous allons dans le répertoire contenant la solution, nous pouvons voir dans le répertoire MaPremiereApplication\MaPremiereApplication\bin\debug  qu’il y a plusieurs fichiers :



Un fichier exécutable a été généré, il nous permet de lancer notre application.

# La syntaxe générale du C#

Nous allons aborder ici la syntaxe générale du langage de programmation C# dans le cadre d’une application console.

## Le caractère de terminaison de ligne

En général, une instruction en code C# s’écrit sur une ligne et se termine par un **point-virgule**.   
Ainsi, l’instruction que nous avons vue plus haut :Console.WriteLine("Hello World !!");

En général, pour que le code soit le plus lisible possible, on écrit une instruction par ligne et on indente le code de façon à ce que les blocs soient lisibles.

Un bloc de code est délimité par des accolades { et }. Nous y reviendrons plus tard.

Indenter signifie que chaque ligne de code qui fait partie d’un même bloc de code commence avec le même retrait sur l’éditeur. Ce sont soit des tabulations, soit des espaces qui permettent de faire ce retrait.

Visual Studio nous aide pour faire correctement cette indentation quand nous écrivons du code. Il peut également remettre toute la page en forme avec la combinaison de touche : ctrl+k+ctrl+d.

## Les commentaires

Pour faciliter la compréhension du code ou pour se rappeler un point précis, il est possible de mettre des commentaires dans son code. Les commentaires sont ignorés par le compilateur et n’ont qu’une valeur informative pour le développeur.

Dans un fichier de code C# (.cs), on peut écrire des commentaires de 2 façons différentes :

Soit en commençant son commentaire par /\* et en le terminant par \*/ ce qui permet d’écrire un commentaire sur plusieurs lignes.

Soit en utilisant // et tout ce qui se trouve après sur la même ligne est alors un commentaire.

Visual Studio colore les commentaires en vert pour faciliter leurs identifications.

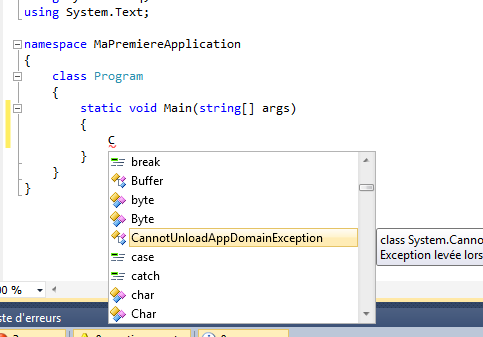
## La complétion automatique

Visual Studio Express est un formidable outil qui nous facilite à tout moment la tâche, notamment grâce à la **complétion automatique**.

La complétion automatique est le fait de proposer de compléter automatiquement ce que nous sommes en train d’écrire en se basant sur ce que nous avons le droit de faire.  
Par exemple, si vous avez cherché à écrire l’instruction :

Console.WriteLine("Hello World !!");

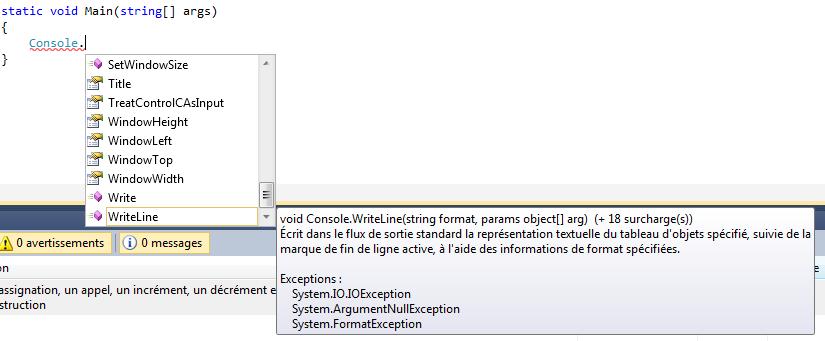
Vous avez pu constater que lors de l’appui sur la touche C, Visual Studio Express nous affiche une fenêtre avec tout ce qui commence par « C » :



Au fur et à mesure de la saisie, il affine les propositions pour se positionner sur la plus pertinente. Il est possible de valider la proposition en appuyant sur la touche Entrée.

Ainsi, finies les fautes de frappe qui résultent en une erreur de compilation ou les listes de mots clés dont il faut absolument retenir l’écriture.

De la même façon, une fois que vous avez fini de saisir « Console » vous allez saisir le point « . » et Visual Studio va nous proposer tous les outils (propriétés et méthodes) disponible dans cette classe.



Nous pourrons ainsi facilement finir de saisir « WriteLine » et ceci sans erreur d’écriture, ni problème de majuscule.

# Les variables

Les programmes ont pour but de traiter différentes données afin de produire des résultats. Les résultats peuvent eux-mêmes être des données pour d'autres programmes.

Les données d'un programme doivent être récupérées en mémoire centrale, à partir du clavier ou d'un fichier par exemple, pour pouvoir être traitées par le processeur qui exécute le programme. Ainsi, toutes les données d'un programme sont mémorisées en mémoire centrale, dans des sortes de cases que l'on appelle variables.

## Qu'est-ce qu'une variable

Une variable peut être représentée par une case mémoire, qui contient la valeur d'une donnée.

Chaque variable possède un nom unique appelé identificateur par lequel on peut accéder à son contenu.

Attention à ne pas confondre la variable et son contenu

Une variable est un contenant, c'est à dire une sorte de boîte, alors que le contenu d'une variable est une valeur numérique, alphanumérique ou booléenne, ou de tout autre type.

Deux variables peuvent avoir la même valeur, mais une variable ne peut pas avoir plusieurs valeurs en même temps.

En revanche, la valeur d'une variable peut varier au cours du programme.

L'ancienne valeur est tout simplement écrasée et remplacée par la nouvelle.

Les variables dont la valeur ne change pas au cours de l'exécution du programme sont appelées variables constantes ou plus simplement constantes.

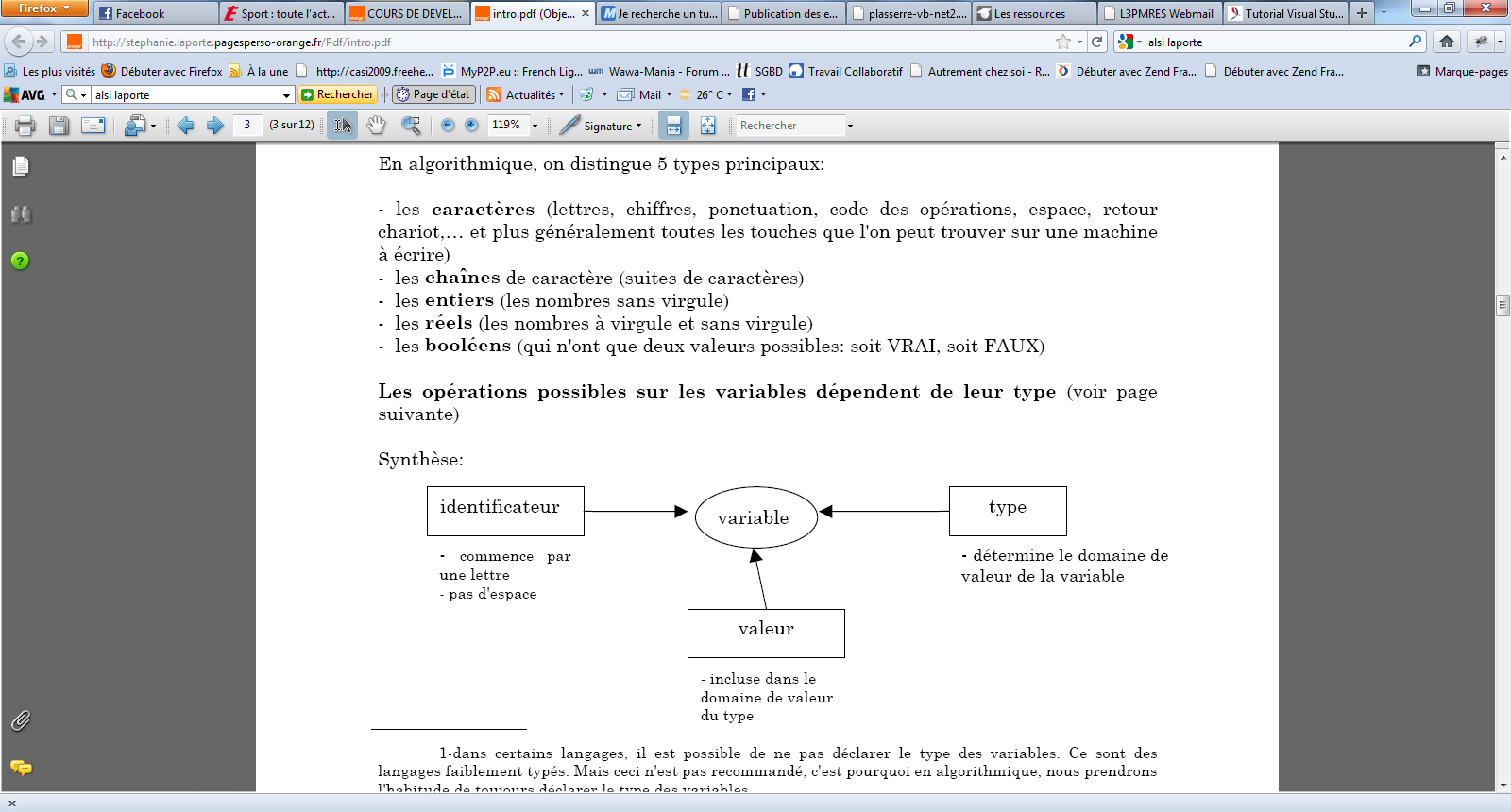
## Déclaration des variables

Pour qu'un programme puisse utiliser une variable, il faut au préalable que cette variable ait été déclarée, c'est-à-dire que le programme lui ait réservé une place en mémoire et ait attribué l'identificateur à cette place.

Mais toutes les variables n'ont pas besoin de la même place en mémoire. Un grand nombre prend plus de place qu'un caractère. Selon le type de l'objet, il faudra lui réserver plus ou moins de place : c'est pourquoi il faut déclarer le type des variables et pas seulement leur nom. Par ailleurs, selon le type des variables, les opérations possibles seront différentes.

Donc la déclaration d'une variable indique deux choses :

* son identificateur (son nom)
* son type (sa taille)



## Les différents types de variables

Nous avons vu juste au-dessus que la variable « age » pouvait être un entier numérique grâce au mot clé int. Le framework .NET dispose de beaucoup de types permettant de représenter beaucoup de choses différentes.

Par exemple, nous pouvons stocker une chaine de caractères grâce au type string.

string prenom = "nicolas";

ou encore un décimal avec :

decimal soldeCompteBancaire = 100;

ou encore un boolean (qui représente une valeur vraie ou fausse) avec

bool estVrai = true;

Il est important de stocker des données dans des variables ayant le bon type.

On ne peut par exemple pas stocker le prénom "Nicolas" dans un entier.

Les principaux types de base du framework .NET sont :

| **Type** | **Description** |
| --- | --- |
| byte | Entier de 0 à 255 |
| short | Entier de -32768 à 32767 |
| int | Entier de -2147483648 à 2147483647 |
| long | Entier de -9223372036854775808 à 9223372036854775807 |
| float | Nombre simple précision de -3,402823e38 à 3,402823e38 |
| double | Nombre double précision de -1,79769313486232e308 à 1,79769313486232e308 |
| decimal | Nombre décimal convenant particulièrement aux calculs financiers (en raison de ses nombres significatifs après la virgule) |
| char | Représente un caractère |
| string | Une chaine de caractère |
| bool | Une valeur booléenne (vrai ou faux) |

Vous verrez plus loin qu'il existe encore d'autres types dans le framework .NET et qu'on peut également construire les siens.

## Affectations, opérations, concaténation

Il est possible d’effectuer des opérations sur les variables et entre les variables. Nous avons déjà vu comment affecter une valeur à une variable grâce à l’opérateur =.

int age = 30;

string prenom = "nicolas";

Les opérations sur les variables dépendent du type de la variable.

La concaténation est le moyen de mettre bout à bout des chaines de caractères

Par exemple : string message = " Bonjour " + prenom;

## La saisie

L’instruction de saisie permet de communiquer des données au programme. Cette instruction assigne une valeur entrée au clavier dans une variable. Tant que l'utilisateur n'entre rien au clavier, le déroulement du programme est stoppé.

Sans instruction de saisie (ou de lecture sur un périphérique quelconque), un programme fournirait toujours le même résultat.

En VB.Net, l'instruction de saisie est Console.ReadLine(). Cette fonction nous retourne une valeur qui faut utiliser : soit l'afficher, soit la stocker dans une variable.

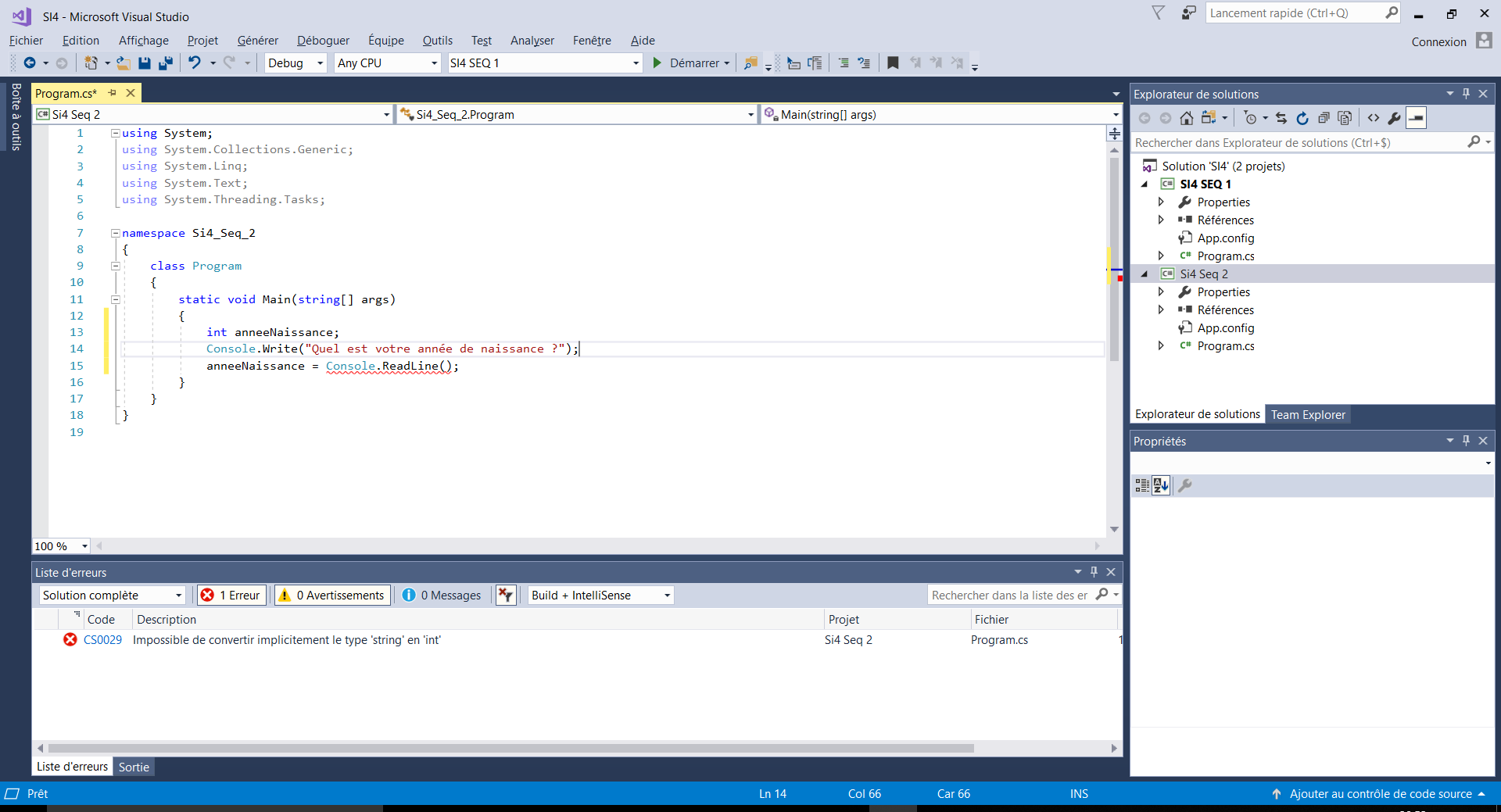
Fin de la séance 1 de 4h – Exo saisie du nom et affichage de bonjour

Exercice 1 :

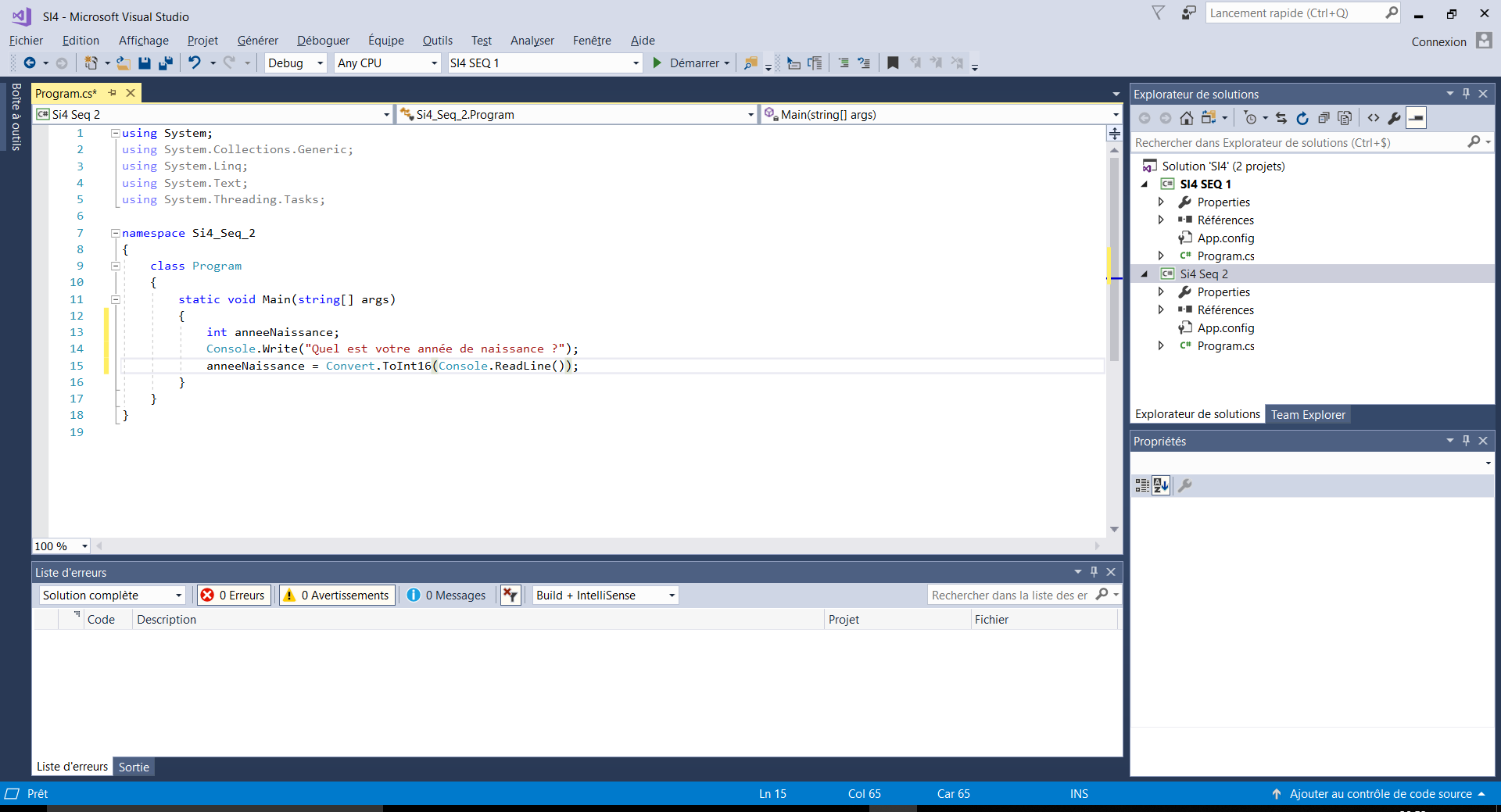
Entrée : nom, prénom et année de naissance

Sortie : Bonjour Sébastien Pernelle vous êtes né en 1977

Problème sur la saisie de l’année



Il faut caster Convert.toInt(…) car c.r retourne une chaine



Exercice 2 :

Ajouter à l’exercice précédent, vous avez donc 40 ans

Et l’année prochaine ça marchera toujours ?

Revenir sur orga du FW, donc une boite Date avec des outils à disposition

Exercice 3 :

Soit le programme suivant :

string var1 = "contenu 1";

string var2 = "contenu 2";

Console.WriteLine("var1 contient :" + var1);

Console.WriteLine("var2 contient :" + var2);

Modifier le programme pour que les valeurs soient échangées