C# : introduction

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com

Plan

C#

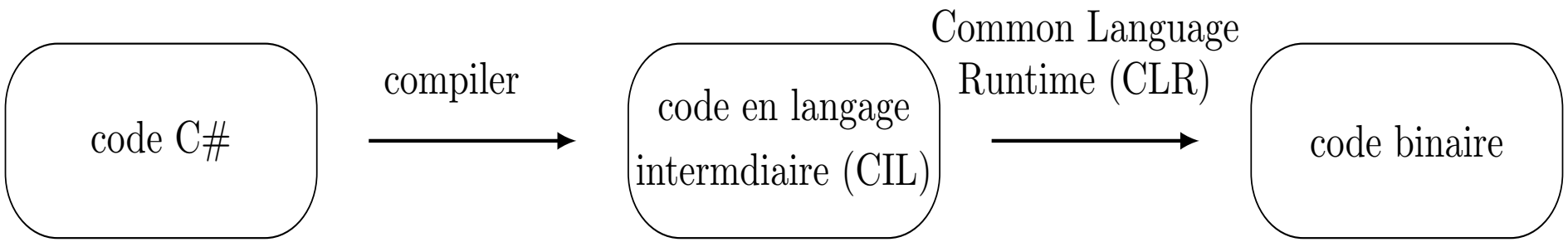
C#, c'est quoi ?

- un langage de programmation orienté objet, fortement typé
- créé par Anders Hejlsberg et présenté officiellement en 2002 par MicroSoft
- permettant de développer des applications qui s'exécutent sur le framework MicroSoft .NET.
- introduit pour concurrencer Java : syntaxe et concept assez proches.
- permettant de développer des applications web, application du bureau (Client lourd), application mobiles (sous windows phone), web services, jeux...

C#

Comment ça fonctionne ?

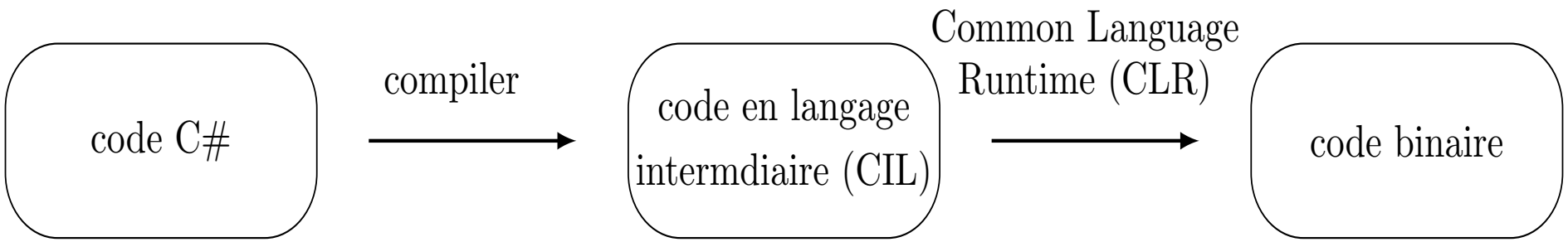
- On écrit un programme C#
- Le code C# sera transformé en un langage intermédiaire (appelé CIL pour *Common Intermediate Language* ou MSIL pour *Microsoft Intermediate Language*) : un fichier `.exe` sans code binaire
- Le code CIL sera compilé par la machine virtuelle CLR pour avoir un code binaire.



C#

code CIL vs code binaire

- CIL : un code intermédiaire qu'on peut exécuter sur n'importe quelle machine Windows.
- code binaire : adapté à la machine sur laquelle il tourne.



C#

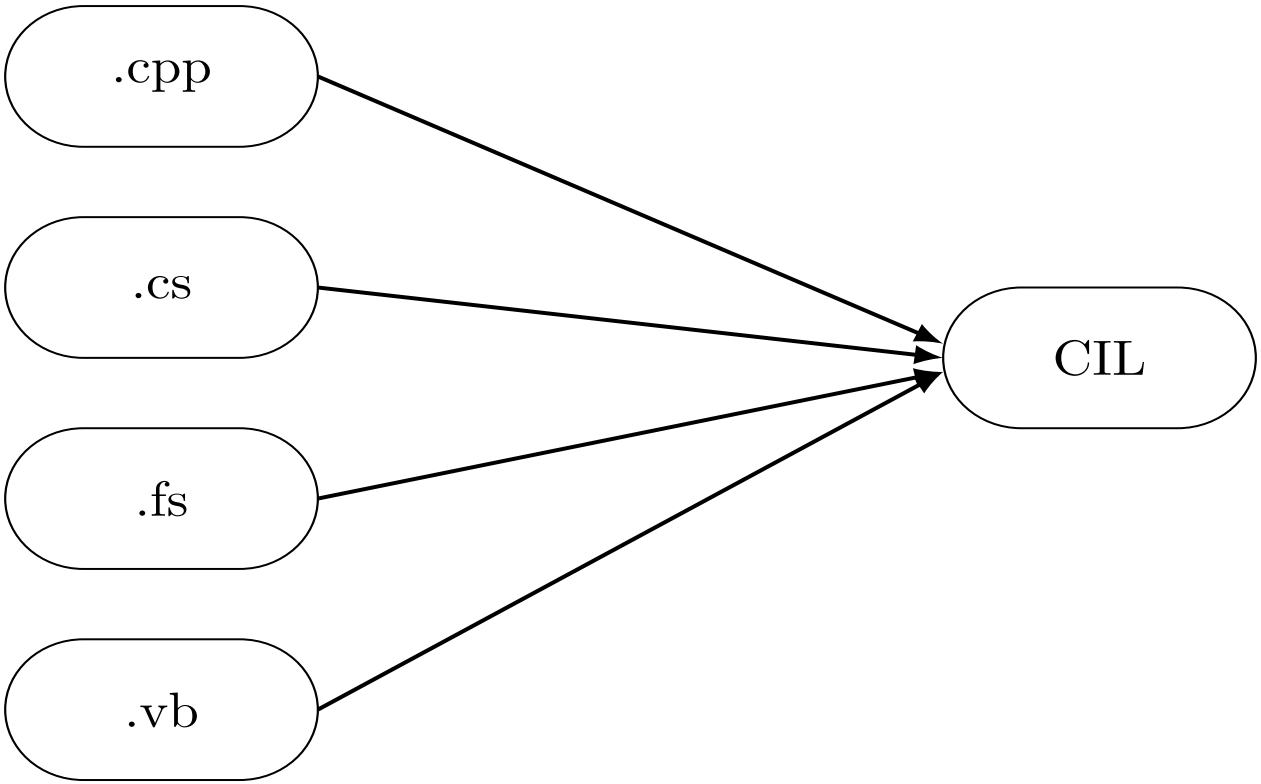
En plus, la machine virtuelle (CLR) dispose de :

- JIT (Just In Time) : pour debugger
- Garbage Collector : pour gérer la mémoire
- CTS (Common Type System) : fournit une bibliothèque contenant les types de données primitif
- CLS (Common Language Specification) : pour vérifier qu'un programme respecte les spécifications .NET
- ...

C#

Dans un framework .NET

- On peut écrire un code C#, et aussi VB, C++, F#.
- Tous ces langages seront compilés en code CIL



C#

À partir d'un programme C#, il est possible

- soit de créer des programmes `.exe`
- soit de créer des bibliothèques sous la forme d'un fichier `.dll`

C#

À partir d'un programme C#, il est possible

- soit de créer des programmes `.exe`
- soit de créer des bibliothèques sous la forme d'un fichier `.dll`

C'est quoi la différence ?

- `.exe` permet de lancer un programme
- `.dll` peut être utilisée par plusieurs programmes `.exe`

C#

À partir d'un programme C#, il est possible

- soit de créer des programmes `.exe`
- soit de créer des bibliothèques sous la forme d'un fichier `.dll`

C'est quoi la différence ?

- `.exe` permet de lancer un programme
- `.dll` peut être utilisée par plusieurs programmes `.exe`

Dans les deux cas

- On parle d'un `assembly`

C#

Et le framework .NET ?

- Framework créé par Microsoft en 2002, en même temps que le C#, et aussi par la même personne
- Permettant de développer des programmes (applications) fonctionnant dans un environnement Microsoft.
- Plusieurs langages possibles : C#, C++, F#...

C#

Et le framework .NET ?

- Framework créé par Microsoft en 2002, en même temps que le C#, et aussi par la même personne
- Permettant de développer des programmes (applications) fonctionnant dans un environnement Microsoft.
- Plusieurs langages possibles : C#, C++, F#...

Exemple

- Paint.net : l'éditeur d'images est réalisé avec le framework .NET

C#

Le .NET est compatible seulement avec Windows

Pour écrire des programmes C# sous Linux ou MAC, on peut utiliser

- le framework **mono**
- le framework **Xamarin**

Visual Studio ?

- Un IDE (Integrated Development Environment) qui nous permet d'écrire des programmes avec le framework .NET

C#

Téléchargement et installation

- Aller sur le lien `https://www.visualstudio.com/fr/downloads/`
- Choisir la version communauté `Visual Studio Community 2017`
- Télécharger puis lancer l'installation

C#

Étapes

- Créer un nouveau projet `Fichier > Nouveau > Projet`
- Cliquer sur `Installé` et choisir `C#`
- Dans `Windows Desktop`, sélectionner `Application console (.NET Framework)`
- Remplir surtout les champs `Nom :` avec `MonProjet` et `Solution` avec `MaSolution` (Dans `Emplacement`, *Visual Studio* nous informe sur le dépot où notre projet sera placé. Par défaut, c'est dans le `c:/utilisateurs/utilisateur/source/repos`.)

C#

Code obtenu

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MonProjet
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Syntaxe et concept assez proches de celles de C++ et Java.

C#

Explication

- On utilise `Using` pour importer les namespaces (comme en C++)
- Le `namespace` permet de déclarer un nouveau namespace (comme un package Java)
- `static void Main()` : point d'entrée de notre application console

Afficher le `Hello world!`

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MonProjet
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Hello world");
        }
    }
}
```

Afficher le `Hello world!`

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MonProjet
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Hello world");
        }
    }
}
```

Exécuter en cliquant sur `Démarrer` (ou la touche **CTRL + F5**, ou **[fn +] CTRL + F5**).

C#

Si on ne voit pas l'exécution.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MonProjet
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Hello world");
            Console.ReadKey();
        }
    }
}
```

C#

Constat

- Le titre de la console contient le chemin vers l'exécutable `c:/utilisateurs/utilisateur/source/repos/MaSolution/MonProjet/bin/Debug/MonProjet.exe`

Pour accéder rapidement aux fichiers de notre projet

- Aller dans le menu `Affichage` et cliquer sur `Explorateur de solutions`
- Un clic droit sur `MaSolution` qui apparaît dans le panneau `Explorateur de solutions` et choisir `Ouvrir le dossier dans l'Explorateur de fichiers`

C#

Objectif

- Écrire une solution .NET avec plusieurs langages de programmation.

C#

Étape 1 : créer un projet VB appartenant à MaSolution

- Aller `Fichier > Nouveau > Projet`
- Choisir `Visual Basic`
- Sélectionner `Bibliothèque de classes (.NET Framework)`
- Saisir `MonVB` dans `Nom`
- Dans `Solution :`, choisir `Ajouter à la solution`
- Choisir `MaSolution` et valider
- Vérifier la présence de deux projets dans l'`Explorateur de solutions`

C#

Code obtenu

```
Public Class Class1


End Class
```

C#

Code obtenu

```
Public Class Class1

End Class
```

Étape 2, modifions le

```
Public Class ClassVB
    Sub SayHello()
        Console.WriteLine("Message VB")
    End Sub
End Class
```

Si on vérifie le répertoire `bin/Debug` de ce projet VB, on verra qu'il est vide (pas d'exécutable), donc inexploitable.

C#

Étape 3 : générer l'exécutable

- Aller dans l'`Explorateur de solution`
- Faire un clic droit sur le projet `MonVB`
- Choisir `Générer`

C#

Étape 3 : générer l'exécutable

- Aller dans l'`Explorateur de solution`
- Faire un clic droit sur le projet `MonVB`
- Choisir `Générer`

Si on vérifie le répertoire `bin/Debug`, trois fichiers ont été générés dont un `.dll`

C#

Étape 4 : connecter les deux projets

- Aller dans l'`Explorateur de solution`
- Dans le projet `MonProjet`, Faire un clic droit sur `Références` et choisir `Ajouter une référence`
- Cliquer sur `Projets` et cocher la case `MonVB`
- Valider

C#

Étape 4 : connecter les deux projets

- Aller dans l'`Explorateur de solution`
- Dans le projet `MonProjet`, Faire un clic droit sur `Références` et choisir `Ajouter une référence`
- Cliquer sur `Projets` et cocher la case `MonVB`
- Valider

Vérifier que `MonVB` figure dans la liste de références de `MonProjet`

C#

Étape 5 : utilisons la classe `ClassVB` écrite en *Visual Basic* dans *C#*

```
using System;
using MonVB;

namespace MonProjet
{
    class Program
    {
        static void Main(string[] args)
        {
            ClassVB c = new ClassVB();
            c.SayHello();
            Console.Write("hello world");
            Console.ReadKey();
        }
    }
}
```

C#

Exécuter : le résultat est

```
Message VB
Hello world
```

C#

Pour écrire dans la console

```
Console.WriteLine("Un message et un retour à la
   ligne");
```

Pour écrire sans retourner à la ligne

```
Console.Write("Un message sans retour à la ligne");
```

C#

Commentaire sur une seule ligne

```
// commentaire
```

Commentaire sur une plusieurs lignes

```
/* le commentaire
   la suite
   et encore la suite
*/
```

Commentaire pour la documentation

```
/// un commentaire qui sera inclu dans la
   documentation
```

C#

Modifier la console

- `Console.BackgroundColor = ConsoleColor.Red;` pour mettre la couleur de fond en rouge
- `Console.ForegroundColor = ConsoleColor.Yellow;` pour mettre la couleur de caractères en jaune
- `Console.ResetColor();` pour réinitialiser les couleurs
- `Console.Clear();` pour effacer le contenu de la console
- `Console.SetCursorPosition(50, 50);` pour positionner la console
- ...

C#

La documentation officielle (en français)

`https://docs.microsoft.com/fr-fr/dotnet/csharp/index`