**Le C#**

La CLR vérifie aussi la sécurité du code ; ainsi en C du code mal pensé (par exemple une mauvaise utilisation des pointeurs) peut entraîner des problèmes pour votre PC, ce que vous risquez beaucoup moins avec le C#. De plus, la CLR dispose du JIT debugger qui permet de lancer Visual Studio si une erreur survient dans un programme .NET pour voir ce qui a causé cette erreur. On parle de code managé.

En conclusion, dans la pratique, le .NET est totalement exploitable sous Windows, ailleurs non.

J'ai dit juste au dessus que le C# était compilé en langage intermédiaire et qu'on le retrouve sous la forme d'un .exe comme les programmes habituels.

**En résumé**

* Le C# est un langage de programmation permettant d’utiliser le framework .NET. C’est le langage phare de Microsoft.
* Le framework .NET est une énorme boîte à fonctionnalités permettant la création d’applications.
* Le C# permet de développer des applications de toutes sortes, exécutables par le CLR qui traduit le MSIL en binaire.
* Il est possible de créer des assemblys de deux sortes : des assemblys de processus exécutables par le CLR et des assemblys de bibliothèques.

c’est en fait l’**outil de développement**. Il va nous fournir les outils pour orchestrer nos développements.

C’est entre autres :

* Un puissant éditeur
* Un compilateur
* Un environ

**L’éditeur de texte** va nous servir à créer des fichiers contenant des instructions en langage C#.  
**Le compilateur** va servir à transformer ces fichiers en une suite d'instructions compréhensibles par l'ordinateur, comme nous l'avons déjà vu.  
**Le moteur d’exécution** va permettre de faire les actions informatiques correspondantes (afficher une phrase, réagir au clic de la souris, etc.), c'est le CLR dont on a déjà parlé.nement d’exécution

Notez que cet outil de développement se nomme également un IDE pour « Integrated Development Environment » ce qui signifie « Environnement de développement intégré ».

## Partie I

# Chapitre Analyse rapide de l’environnement de développement et du code généré

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

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.

Attention, chaque ligne de code doit être correcte syntaxiquement sinon le compilateur ne saura pas le traduire en langage exécutable.

Dans le cas de signalisation d’erreur, visual Studio indique par un message que « Ce sont des erreurs de compilation ». Pour retenir, le C# est **sensible à la casse** (1).

une instruction en code C# s’écrit sur une ligne et se termine par un **point-virgule**

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 }

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.

# Chapitre La complétion automatique

## Les variables

Int string bool double

## Les instructions conditionnelles

| **Opérateur** | **Description** |
| --- | --- |
| == | Egalité |
| != | Différence |
| > | Supérieur à |
| < | Inférieur à |
| >= | Supérieur ou égal |
| <= | Inférieur ou égal |
| && | ET logique |
| || | OU logique |
| ! | Négation |

1. <http://fr.openclassrooms.com/informatique/cours/apprenez-a-developper-en-c/la-syntaxe-generale-du-c>

Le .NET Framework est une plateforme de développement largement utilisée pour la création d'applications destinées à Windows, Windows Store, Windows Phone, Windows Server et Microsoft Azure. La plateforme .NET Framework comprend les langages de programmation [C#](http://msdn.microsoft.com/vstudio/hh341490.aspx) et [Visual Basic,](http://msdn.microsoft.com/vstudio/hh388573.aspx) le [Common Language Runtime,](http://msdn.microsoft.com/library/vstudio/8bs2ecf4.aspx) ainsi qu'une abondante [bibliothèque de classes.](http://msdn.microsoft.com/library/vstudio/gg145045%28v=vs.110%29.aspx)

Le framework .NET contient énormément de méthodes utilitaires de toutes sortes, nous y reviendrons.

Vous aurez peut-être remarqué un détail, nous avons préfixé toutes nos méthodes du mot clé static. J’ai dit que c’était obligatoire dans notre contexte, pour être plus précis, c’est parce que la méthode Main() est statique que nous sommes obligés de créer des méthodes statiques. On a dit que la méthode Main() était obligatoirement statique parce qu’elle devait être accessible de partout afin que le CLR puisse trouver le point d’entrée de notre programme. Or, une méthode statique ne peut appeler que des méthodes statiques, c’est pour cela que nous sommes obligés (pour l’instant) de préfixer nos méthodes par le mot clé static. Nous décrirons ce que recouvre exactement le mot clé static dans la partie suivante.

# Retour d’une méthode

Une méthode peut aussi renvoyer une valeur, par exemple un calcul. C’est souvent d’ailleurs son utilité première.

### En résumé

* Une méthode regroupe un ensemble d’instructions pouvant prendre des paramètres et pouvant renvoyer une valeur.
* Les paramètres d'une méthode doivent être utilisés avec le bon type.
* Une méthode qui ne renvoie rien est préfixée du mot-clé void.
* Le point d'entrée d'un programme est la méthode statique Main().
* Le mot-clé return permet de renvoyer une valeur du type de retour de la méthode, à l'appelant de cette méthode.

## Chapitre Utiliser le framework .NET

= boîte à outils

# L’instruction using

Un espace de nom (en anglais namespace) correspond à un endroit où l’on range des méthodes et des objets. Il est caractérisé par des mots séparés par des points (.).

C’est un peu comme des répertoires, nous pouvons dire que le fichier « DateTime » est rangé dans le répertoire « System » et quand nous souhaitons y accéder nous devons fournir l’emplacement complet du fichier, à savoir System.DateTime.

Cependant, plutôt que d’écrire le chemin complet à chaque fois, il est possible de dire : « ok, maintenant, à chaque fois que je vais avoir besoin d’accéder à une fonctionnalité, va la chercher dans l’espace de nom « System ». Si elle s’y trouve, utilise la ».

C’est ce qu’il se passe grâce à l’instruction : using System;

### En résumé

* On utilise la boucle for pour répéter des instructions tant qu'une condition n'est pas vérifiée, les éléments de la condition changeant à chaque itération.
* On utilise en général la boucle for pour parcourir un tableau, avec un indice qui s'incrémente à chaque itération.
* La boucle foreach est utilisée pour simplifier le parcours des tableaux ou des listes.
* La boucle while permet de répéter des instructions tant qu'une condition n'est pas vérifiée. C'est la boucle la plus souple.
* Il faut faire attention aux conditions de sortie d'une boucle afin d'éviter les boucles infinies qui font planter l'application.

## Partie II

## 1 Lire le clavier dans la console

## Les conversions entre les types

# Entre les types compatibles : Le casting

Int short

# Entre les types incompatibles

String et int

En résumé

Il est possible, avec le casting, de convertir la valeur d’un type dans un autre lorsqu’ils sont compatibles entre eux.

Le casting explicite s’utilise en préfixant une variable par un type précisé entre parenthèses.

Le framework .NET possède des méthodes permettant de convertir des types incompatibles entre eux s’ils sont sémantiquement proches.

# Lire une phrase

# Lire un caractère

### En résumé

* La méthode Console.ReadLine nous permet de lire des informations saisies par l'utilisateur au clavier.
* Il est possible de lire toute une phrase terminée par la touche Entrée ou seulement un unique caractère.

## Utiliser le débogueur

# A quoi ça sert ?

# Mettre un point d’arrêt et avancer pas à pas

# Observer des variables

# Revenir en arrière

## Partie 3 : Introduction à la programmation orientée objet

La définition de l’objet (ou structure de l’objet) permet d’indiquer ce qui compose un objet, c'est-à-dire quelles sont ses propriétés, ses actions etc. Comme par exemple le fait qu’une chaise ait des pieds ou qu’on puisse s’asseoir dessus.

# L’encapsulation

C’est ce qu’on appelle l’**encapsulation**. Cela permet de protéger l’information contenue dans notre objet et de le rendre manipulable uniquement par ses actions ou propriétés. Ainsi, l’utilisateur ne peut pas accéder au café ni au sucre ou encore moins à la monnaie. Notre objet est ainsi protégé et fonctionne un peu comme une boite noire. L’intérêt est que si la personne qui entretient la machine met du café en grain à la place du café soluble, c’est invisible pour l’utilisateur qui se soucie simplement de mettre ses pièces dans la machine.

L’encapsulation protège donc les données de l’objet et son fonctionnement interne.

# Héritage

# Polymorphisme - Substitution

### Polymorphisme

Le mot **polymorphisme** suggère qu’une chose peut prendre plusieurs formes. Sous ce terme un peu barbare se cachent plusieurs notions de l’orienté objet qui sont souvent sources d’erreurs. Je vais volontairement passer rapidement sur certains points qui ne vont pas nous servir pour me concentrer sur ceux qui sont importants pour ce tutoriel. J’espère que vous ne m’en voudrez pas et que vous m’accorderez quelques dérogations qui pourraient déplaire aux puristes.

En fait, on peut dire qu’une manifestation du polymorphisme est la capacité pour un objet de faire une même action avec différents types d’intervenants. C’est ce qu’on appelle le polymorphisme « ad hoc » ou le polymorphisme « paramétré ». Par exemple, notre objet voiture peut rouler sur la route, rouler sur l’autoroute, rouler sur la terre si elle est équipée de pneus adéquats, rouler au fond de l’eau si elle est amphibie, etc …  
Concrètement ici, je fais interagir un objet « voiture » avec un objet « autoroute » ou un objet « terre »… par l’action qui consiste à « rouler ».

### Substitution

La **substitution** est une autre manifestation du polymorphisme. Pas de régime ou de balance dans ce terme-là, il s’agit simplement de la capacité d’un objet fils à redéfinir des caractéristiques ou des actions d’un objet père.

# Interfaces

### En résumé

* L’approche orientée objet permet de modéliser son application sous la forme d’interactions entre objets.
* Les objets ont des propriétés et peuvent faire des actions.
* Ils masquent la complexité d'une implémentation grâce à l'encapsulation.
* Les objets peuvent hériter de fonctionnalités d'autres objets s'il y a une relation d'héritage entre eux.

Créer son premier objet

### En résumé

* On utilise des classes pour représenter quasiment la plupart des objets.
* On utilise le mot-clé class pour définir une classe et le mot-clé new pour l'instancier.
* Une classe peut posséder des caractéristiques (les propriétés) et peut faire des actions (les méthodes).

### En résumé

* Les classes possèdent une méthode particulière, appelée à l'instanciation de l'objet : le constructeur.
* Une instance d'une classe peut être initialisée avec une valeur nulle grâce au mot-clé null.
* Le mot-clé this représente l'objet en cours de la classe