Introduction à la programmation en C# - Bloc A

Qui sommes-nous ?

29a (https://29a.ca/) est une entreprise spécialisée dans 2 volets :

- Cybersécurité ;
- Programmation.

Objectifs

Habiletés à développer:

- Introduction à la programmation C# ;
- Variables ;
- Fonctions ;
- Retour de function ;
- Itération (les boucles) ;

La programmation C#

Introduction à la programmation c#

Les difficultés

Personne n’a mentionné que la programmation s’apprend en 1 journée même si certaine vidéo le mentionne. C’est une activité qui DEMANDE les deux éléments suivants:

1. De la pratique constante ;
2. De la recherche ;

N’oubliez pas que la programmation est une compétence qui se développe et non un savoir (évidemment il faut avoir une base de théorie) !

Première phase de développement

Normalement, la première phase de développement est la phase de sécurité. On analyse la sécurité et on évalue les risques potentiels.

Cependant (et malheureusement) ça ne se passe pas ainsi dans la vraie vie. Du moins, ce n'est pas généralisé ! En général, tout débute par :

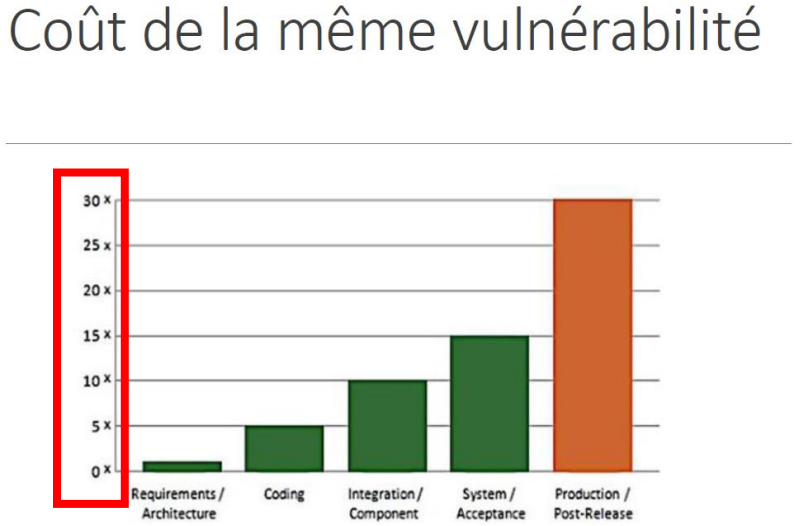
- Un besoin d'affaire ;
- Ensuite, le besoin d'affaire est transformé en besoin fonctionnel ;
- Pour terminer, sur le bureau du développeur ;
- Lorsque l'application est terminée les gestionnaires se disent : **« Bon, il faudrait regarder ce qu'on peut faire pour sécuriser le produit ».**

Coûts des vulnérabilités

Les coûts des réparations des failles de sécurité versus la phase de développement.

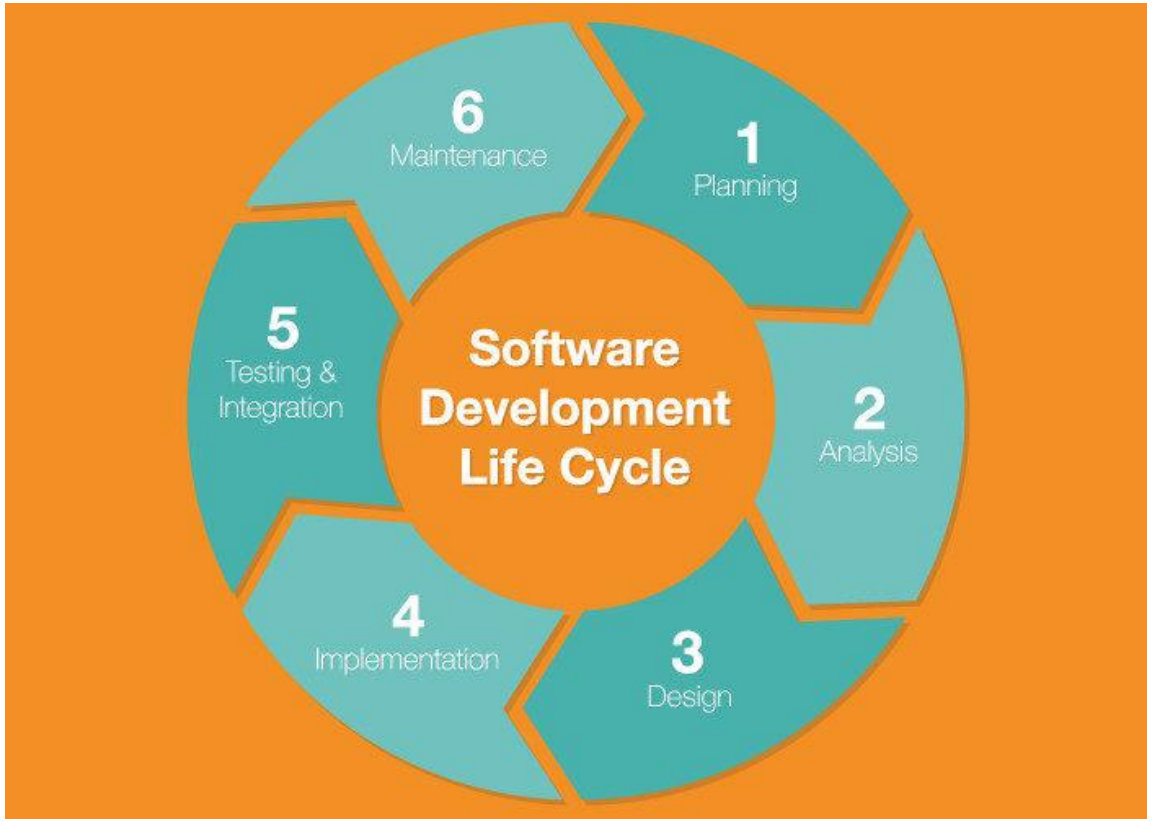
L'objectif n'est pas de vous transformer en spécialiste de la sécurité mais il faut être au courant des impacts avant de faire ses choix de développement

Le prix explose selon les phases/positionnement de l'application



Cycle de vie du logiciel

Un logiciel a une vie et il a un processus de vie! Vous allez souvent entendre le terme ‘Life Cycle’



Source : https://kirkpatrickprice.com/blog/what-is-a-secure-software-development-life-cycle/

Les familles des langages de programmation

- 1- Les langages compilés
- 2- Les langages interprétés

Compilation: Les langages compilés sont directement convertis en langage machine que seul un ordinateur peut comprendre. Ainsi, le processeur est en mesure d'exécuter le programme compilé. En général, ils sont plus rapides et efficaces que les langages interprétés. Avant l'utilisation le programme doit donc être compilé! En C#, lorsqu'on exécute le programme alors le système compile par défaut avant de l'exécuter. Il y a donc une étape de construction avant de pouvoir être executé. Exemples de langages compiles: C#, C, C++, Erlang, Go, etc. Le langage compilé est directement exécuté sur la cible.

Interprétation: Les programmes interprétés n'est pas directement exécuté par la cible. En effet, il y a un 'système' entre les deux qui permet au logiciel de s'exécuter. Le programme est donc exécuté dans un environnement du système d'exploitation.

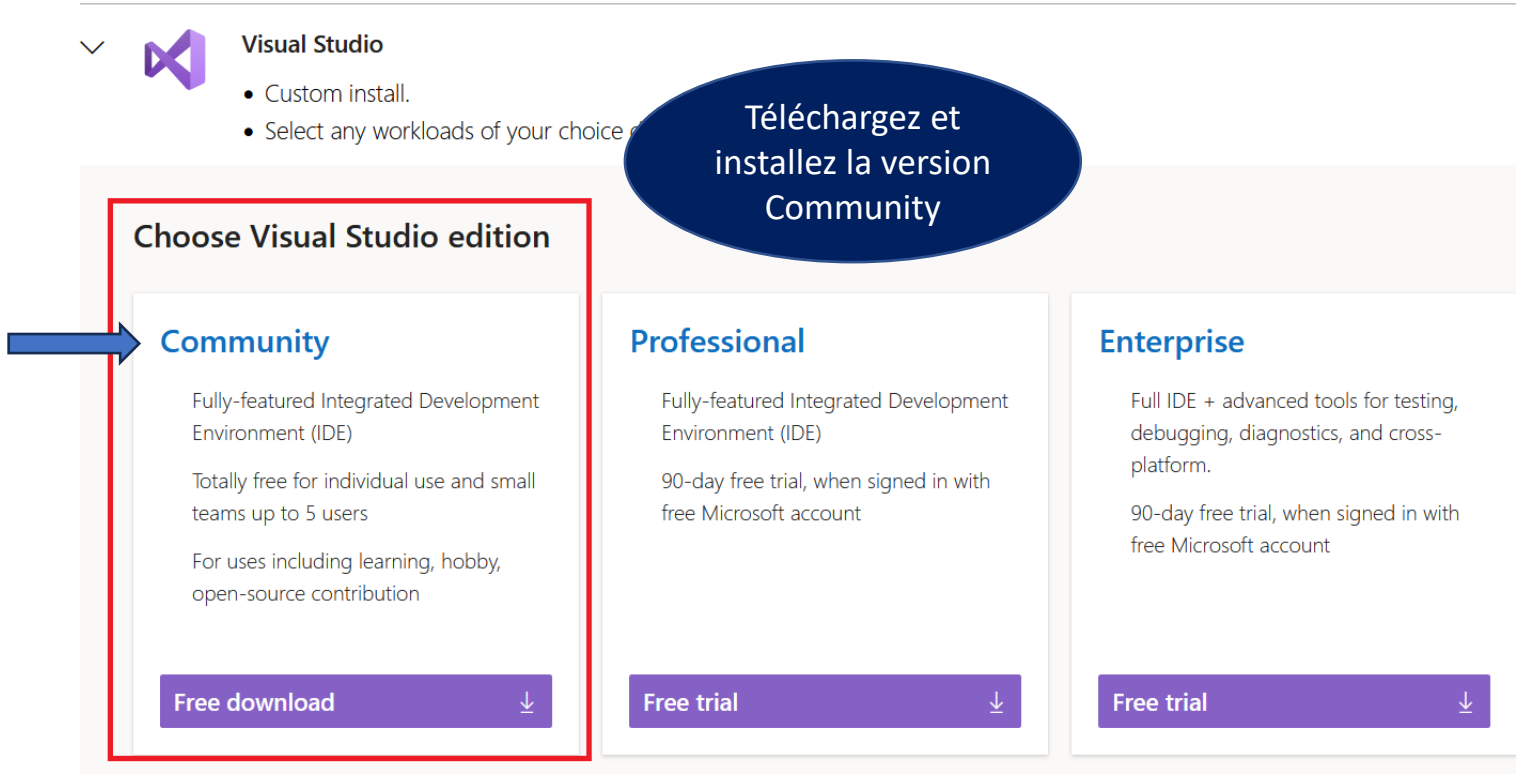
Introduction au C#

- Introduction à la programmation c#

Installation de Visual Studio

Installation de Microsoft Visual Studio

https://visualstudio.microsoft.com/downloads/



Microsoft Visual Studio

Microsoft Visual Studio est un IDE (Integrated Development Environment) complet offrant beaucoup de fonctionnalités. Apprenez à le découvrir et à travailler avec lui! Il est votre compagnon de travail et il vous aide à aller plus vite et à éviter des erreurs.

- Liste d'éléments importants:
- Output console -> Elle permet de voir les messages de l'IDE
- L'aide -> Appuyer sur F1 pour obtenir de l'aide
- Les *views* placer les fenêtres à votre goût
- Le *solution explorer*
- Ouvrir et fermer des programmes
- L'exécution des programmes
- Le mode débug et les *break point*

Les conventions en C#

Le langage C# ne fait pas exception d'une convention d'utilisation. Dans un monde idéal, vous devez suivre la convention C#:

À lire :

https://docs.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions

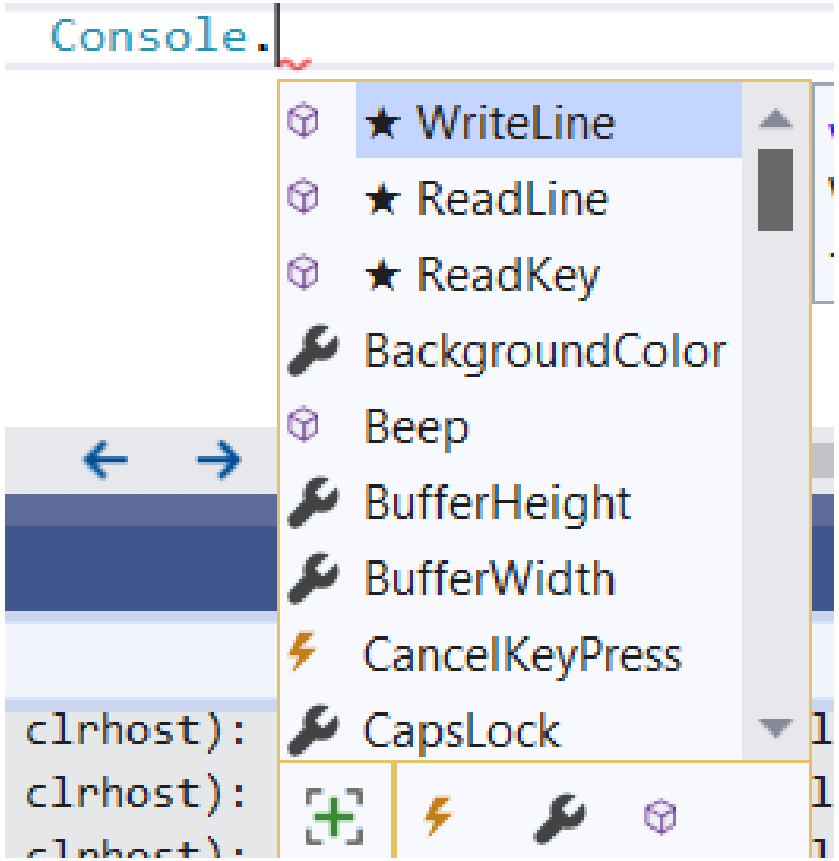
Démarrer en C#

Le C# est un langage de programmation orienté objet qui utilise la **dot.notation**

La dot.notation est utilisé dans la majorité des langages et des *framework*. Utilisez-la pour vous simplifier le travail.

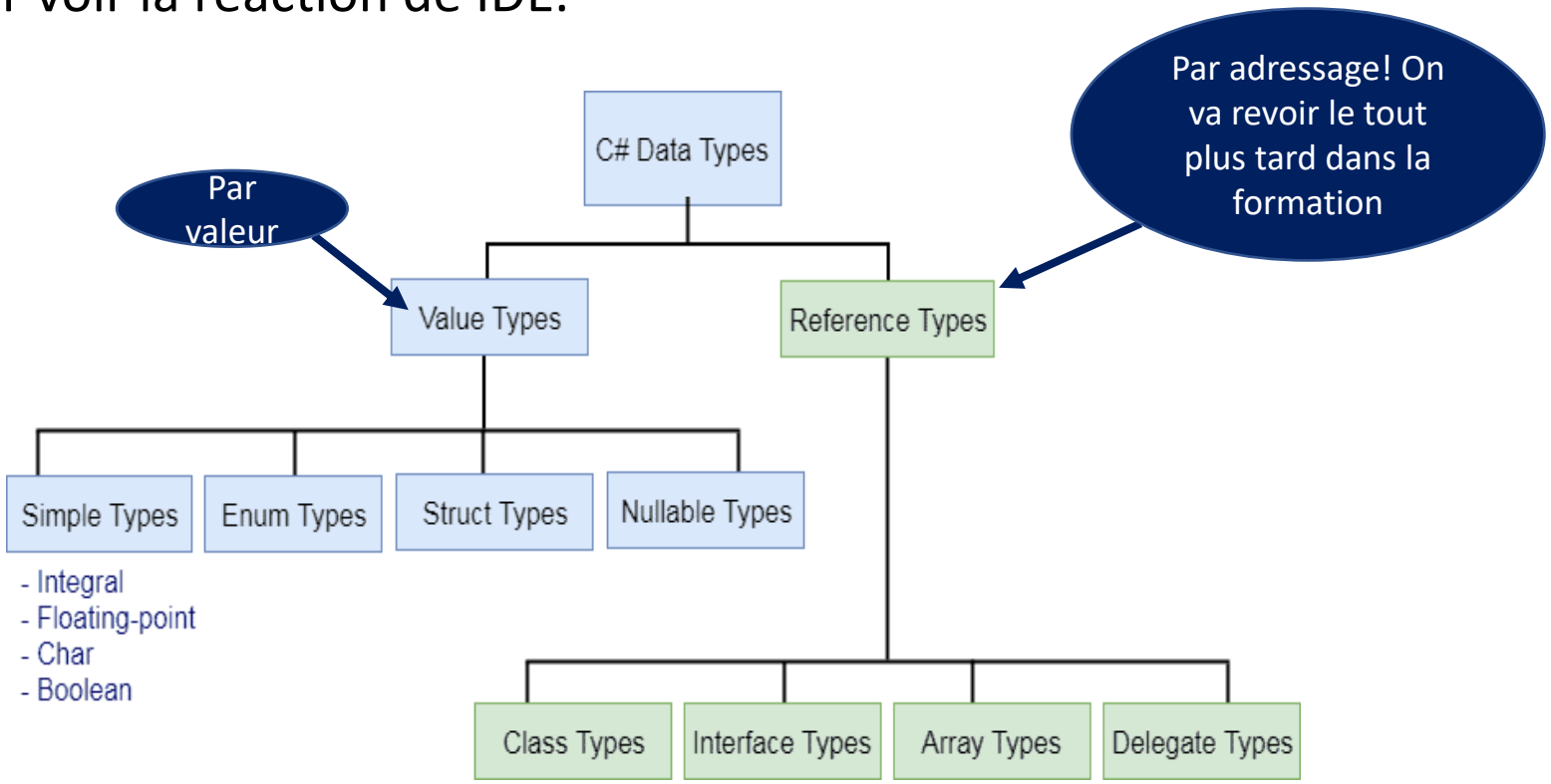
Les objets ont la notion de dot.notation.

Nous aborderons plus loin la notion d'objet. Pour le moment, l'important et de comprendre le concept.



Les variables

La majorité des variables doivent être initialisées avant leurs utilisation. Vous devrez faire des essais pour voir la réaction de IDE.



Affichage à l’écran

En mode console, les programmeurs utilisent la function intégrée *Console.WriteLine();* pour afficher à l’écran de la console.

Exemple: Console.WriteLine("Message à afficher !");

Saisie de données

L'instruction suivante permet de saisir l'information de l'utilisateur: *Console.ReadLine();*

Exemple:

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

La valeur saisie va se retrouver dans la variable value;

Les types de variable en C#

Type	Description	Range	Suffix
byte	8-bit unsigned integer	0 to 255	
sbyte	8-bit signed integer	-128 to 127	
short	16-bit signed integer	-32,768 to 32,767	
ushort	16-bit unsigned integer	0 to 65,535	
int	32-bit signed integer	-2,147,483,648 to 2,147,483,647	
uint	32-bit unsigned integer	0 to 4,294,967,295	u
long	64-bit signed integer	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	l

Les types de variable en C#

Type	Description	Range	Suffix
ulong	64-bit unsigned integer	0 to 18,446,744,073,709,551,615	ul
float	32-bit Single-precision floating point type	-3.402823e38 to 3.402823e38	f
double	64-bit double-precision floating point type	-1.79769313486232e308 to 1.79769313486232e308	d
decimal	128-bit decimal type for financial and monetary calculations	(+ or -)1.0 x 10e-28 to 7.9 x 10e28	m
char	16-bit single Unicode character	Any valid character, e.g. a,*, \x0058 (hex), or\u0058 (Unicode)	
bool	8-bit logical true/false value	True or False	
object	Base type of all other types.		

Exemple de variables en C#

```
// De type numérique
byte myByte = 255; //0 à 255 (max de 255) c'est un 8-bit
short myShort = 32767; //-32,768 à 32,767 c'est un 16-bit
int x = 1345; // -2,147,483,648 à 2,147,483,647 c'est un 32-bit
long y = 1250004; // -9,223,372,036,854,775,808 à 9,223,372,036,854,775,807 C'est un 64-bit
float z = 2555.45F; // précision de -3.402823e38 à 3.402823e38. C'est un 32-bit (même chose qu'un int mais avec les décimals)
double w = 128222555; // précision de -1.79769313486232e308 to 1.79769313486232e308. C'est un 64-bit (même chose qu'un long)
decimal myDecimal = 451245788; // 128-bit

// De type caractère
char unCaractere = 'a'; //16-bit single Unicode character

// Boolean
bool myBoolean = false; // 8-bit

// Une séquence de caractère unicode
string maChaine = "Ma chaine de caractère";

// De type objet
Object myObject = new object();
```

Compilation

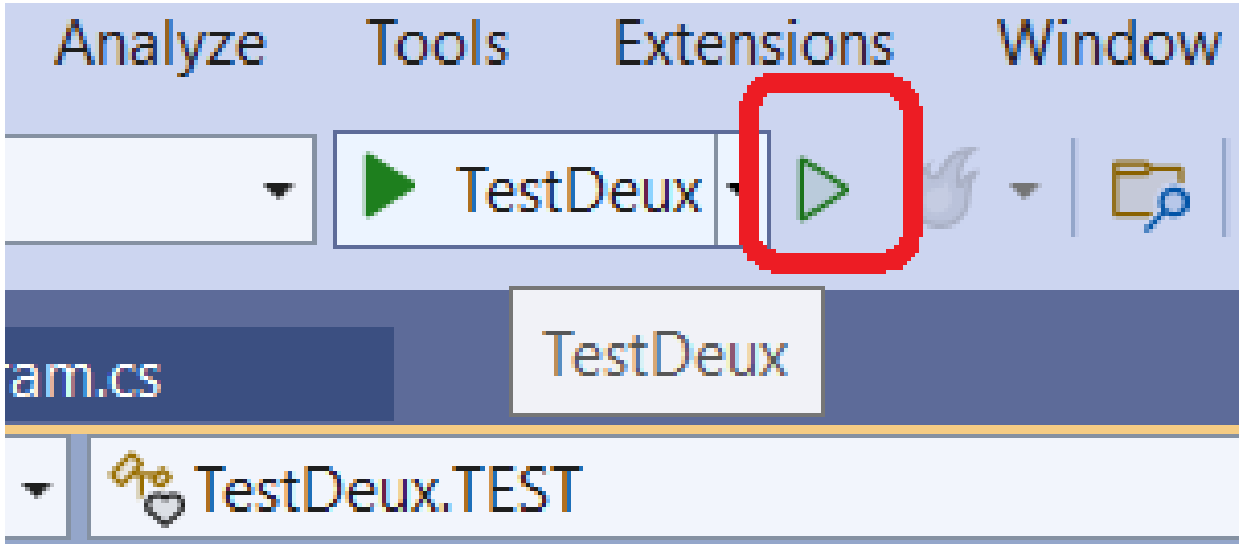
Le C# est un langage compilé! Donc, pour pouvoir être exécuté il est obligatoire que le compiler avant de l'exécuter.

Il y a deux façons de compiler

- Directement exécuter l'application
- « Build » ou « Rebuild »

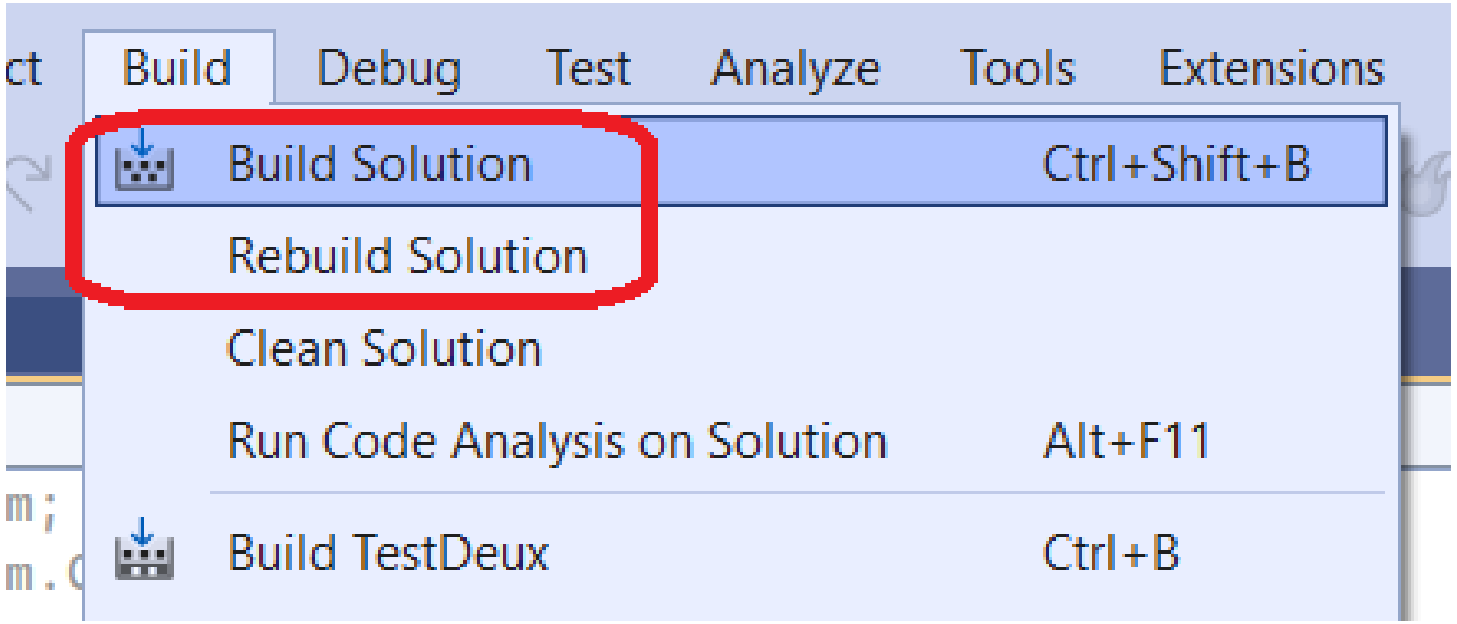
Compilation – *Execute*

L’exécution permet de faire le « Build » et ensuite d’exécuter l’application



Compilation – Build Solution et Rebuild Solution

- *Build Solution* = permet de compiler les fichiers qui ont changés.
- *Rebuild Solution* = supprime tous les fichiers compilés et recompile tous les fichiers.



Questions ?

- Contact : info@29a.ca
- Site : https://29a.ca/