

# Tutorial: Memento d'utilisation de Visual

Ce memento fournit un guide de survie pour l'utilisation restreinte de Visual Studio faite dans le cours d'environnement et programmation en BAT4.

## 1. Organisation de Visual Studio

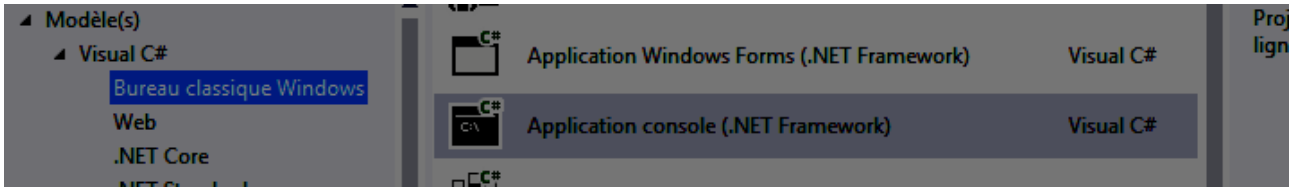
### Principe général de développement sous Visual Studio

Quand vous ouvrez Visual Studio, vous travaillez sur une **unique** *solution*. De façon informelle, une *solution* répond à un besoin client, et fournit différentes fonctionnalités. De façon concrète, une *solution* (décrite dans un fichier .sln) contient un ou plusieurs *projets.* Par exemple, une solution peut contenir un projet de type console, un projet de type graphique, une dll et un projet de type web. Un *projet* contient tout ce qui est nécessaire au fonctionnement du programme correspondant, comme par exemple des fichiers d'images ou de ressources.

### Création d'un nouveau projet

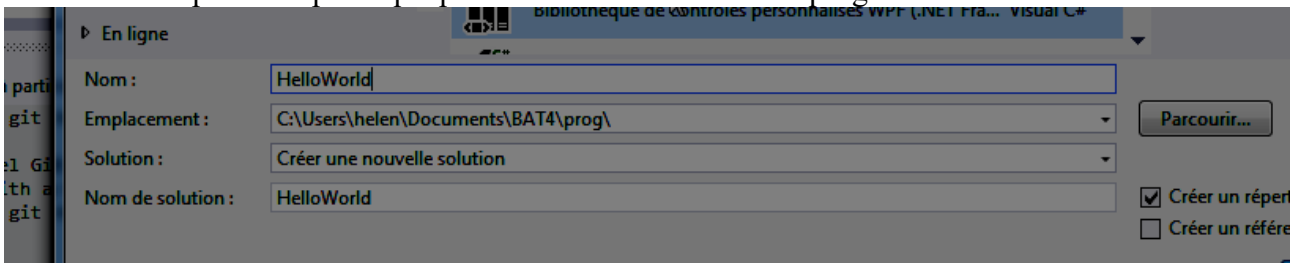
Menu fichier → nouveau projet.

Nous utiliserons les 2 types de projets suivants :




- **Application console** : application qui lit/écrit les informations du programmes sur la console (i.e. Fenêtre d'invite de commandes)
- **Application Windows Forms** : application graphique, les informations sont données dans des composants graphiques (cases à cocher, menus, …)

**Attention** à bien choisir le nom de votre application et l'endroit où le projet est stocké sur la machine, créez-vous un répertoire spécifique pour les fichiers du cours de programmation.





# Tutorial: Memento d'utilisation de Visual

### Importer une solution

Nous vous fournirons des corrigés qui correspondent à des .zip du répertoire associé à la solution. Il faudra **décompresser** l'archive dans votre répertoire de travail pour BAT4. Pour ouvrir la solution, cliquer sur le fichier .sln, cela ouvrira une nouvelle fenêtre Visual Studio. Si vous voulez garder une seule fenêtre Visual Studio, vous pouvez charger la solution en faisant : Fichier → nouveau → projet à partir de code existant.

## 2. Edition et exécution d'un programme en mode console

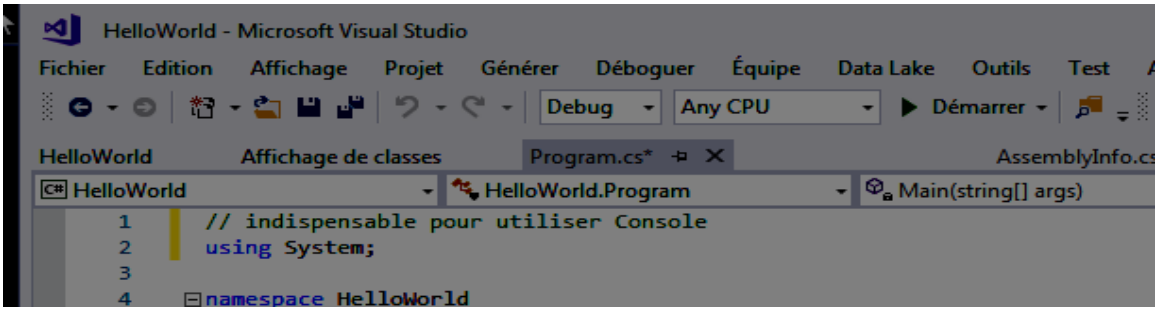
### Premier programme de type console

```
// indispensable pour utiliser Console
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            // affiche "bonjour à tous" dans la fenêtre d'invite de commandes
            Console.WriteLine("bonjour à tous");
            // attend qu'on tape quelque chose au clavier
            // sert à laisser visible la fenêtre où le message s'est affiché.
            Console.ReadLine();
        }
    }
}
```

Ce programme affiche « Bonjour à tous » dans une fenêtre d'invite de commandes windows et attend qu'on tape une touche clavier. Quand le programme est terminé (i.e. on a tapé sur une touche) la fenêtre d'invite de commande se ferme.

### Exécution

Pour exécuter, en haut, flèche verte « démarrer ». Cela exécute le main (point d'entrée du programme) du programme HelloWorld.

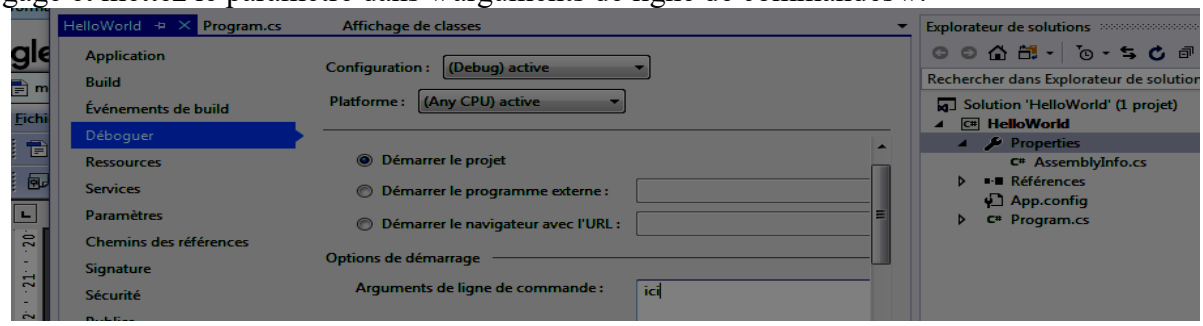




# Tutorial: Memento d'utilisation de Visual

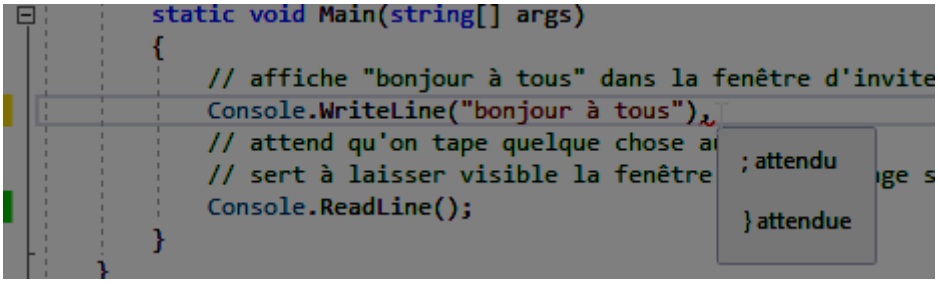
**Attention**, si votre solution contient plusieurs projets, il y a un seul projet de démarrage. On peut le choisir en faisant : projet → définir comme projet de démarrage.

Pour les avancés, si vous voulez passer un paramètre au programme principal, cliquer sur properties → debogage et mettez le paramètre dans « arguments de ligne de commandes ».

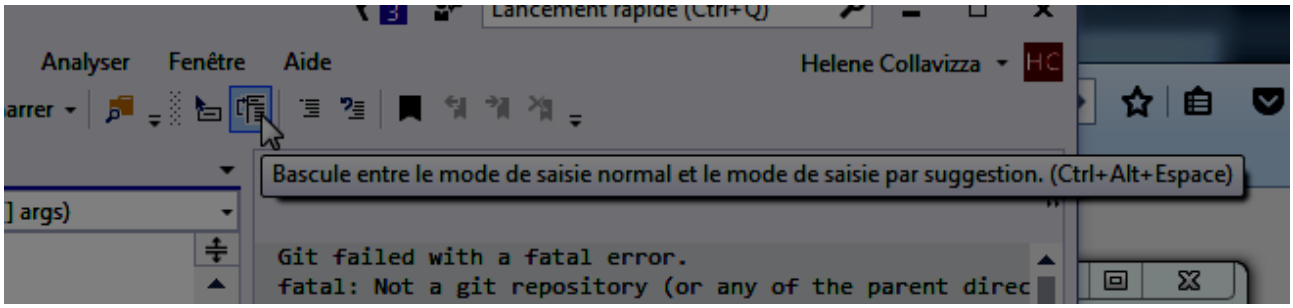



**Compilation incrémentale et mode de saisie par suggestion**

Quand vous modifiez un programme dans l'éditeur, VisualStudio effectue une **compilation incrémentale** et signale les erreurs en les soulignant avec des ~~~. Si vous passez la souris sur la zone il suggère des corrections.

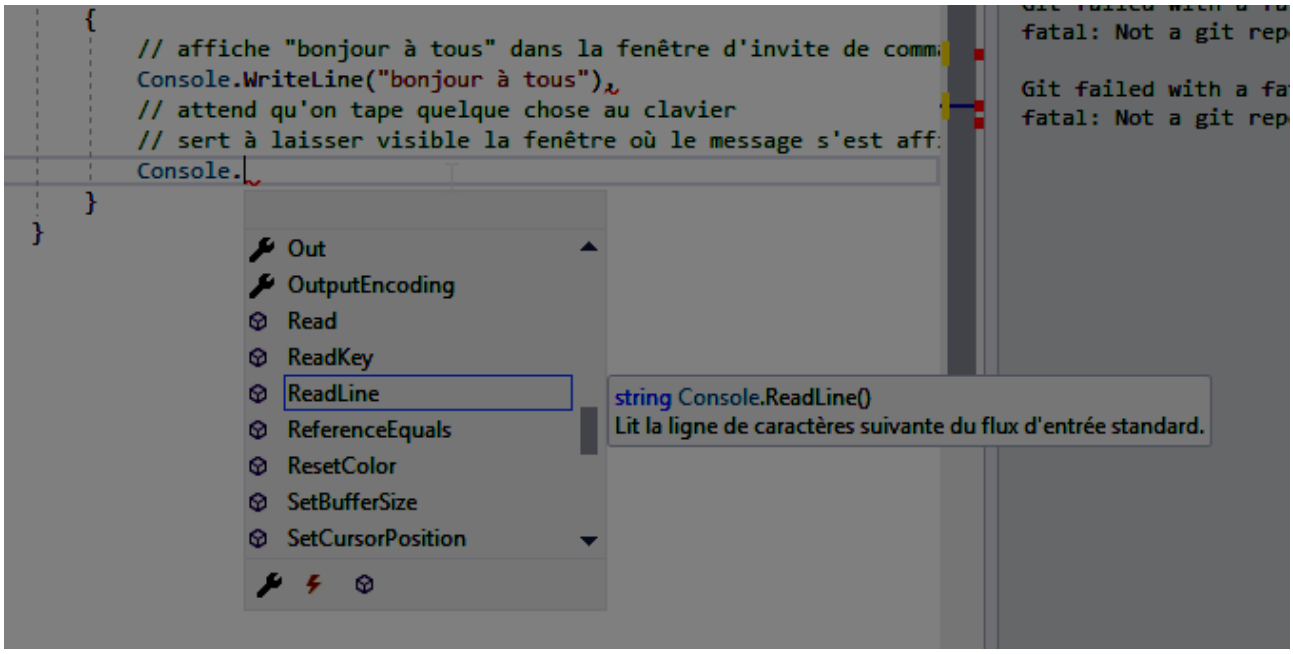



Vous pouvez opter pour la saisie par suggestion, en sélectionnant le bouton ci-dessous.




Dans ce cas, quand vous commencez à taper une instruction C# l'éditeur vous proposera des suggestions en adéquation avec le début du mot tapé.

# Tutorial: Memento d'utilisation de Visual




**Exécution pas à pas (débogage)**

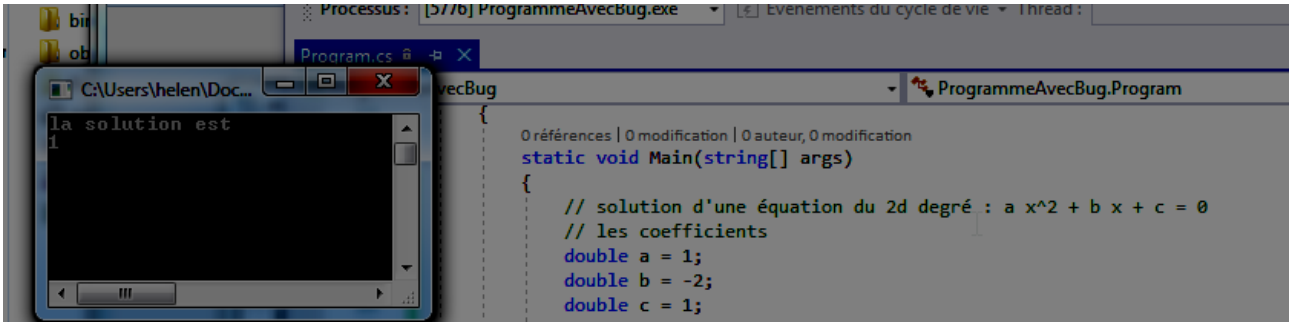
Il est souvent difficile de diagnostiquer la raison d'une erreur dans un programme. Il faut pour cela mener une enquête … Le mode « débogage pas à pas » vous permet de tracer l'exécution de votre programme pas à pas en plaçant des points d'arrêt, ce qui permet de voir en détail le comportement de chaque instruction.

Voici un exemple de programme qui contient un bug. Il calcule (presque) une solution d'une équation du 2d degré mais plante si le discriminant est négatif.

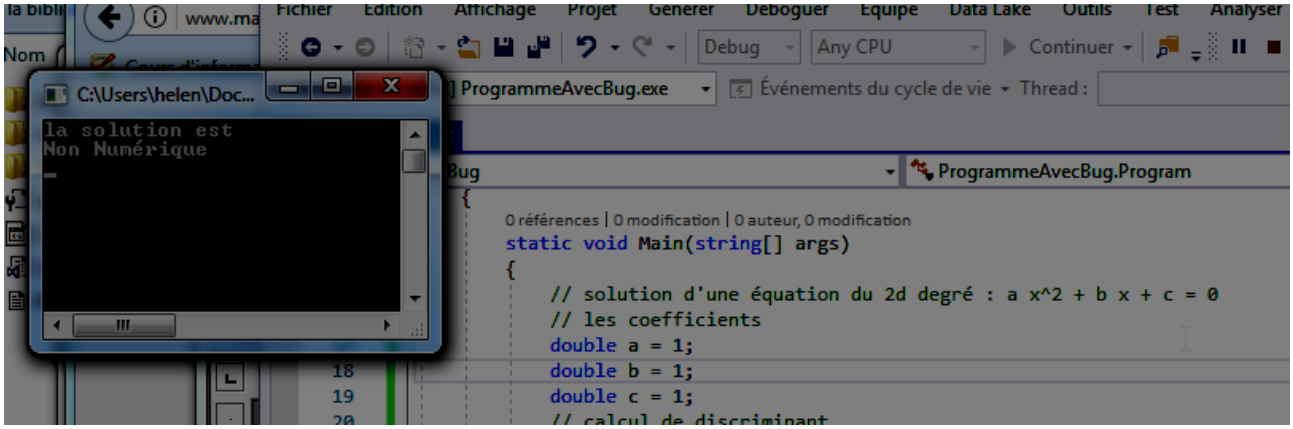
```
using System;

namespace ProgrammeAvecBug
{
    // un exemple de programme avec bug pour montrer l'intérêt
    // d'utiliser le debogueur
    class Program
    {
        static void Main(string[] args)
        {
            // solution d'une équation du 2d degré : a x^2 + b x + c = 0
            // les coefficients
            double a = 1;
            double b = -2; // plante si b=1
            double c = 1;
            // calcul de discriminant
            double delta = b * b - 4 * a* c;
            double racineDelta = Math.Sqrt(delta);
            // calcul et affichage solution
```

```
            double sol = (-b - racineDelta) / (2 * a);
            Console.WriteLine("la solution est");
            Console.WriteLine(sol);
            Console.ReadLine();
        }
    }
}
```

Cas d'exécution sans problème :



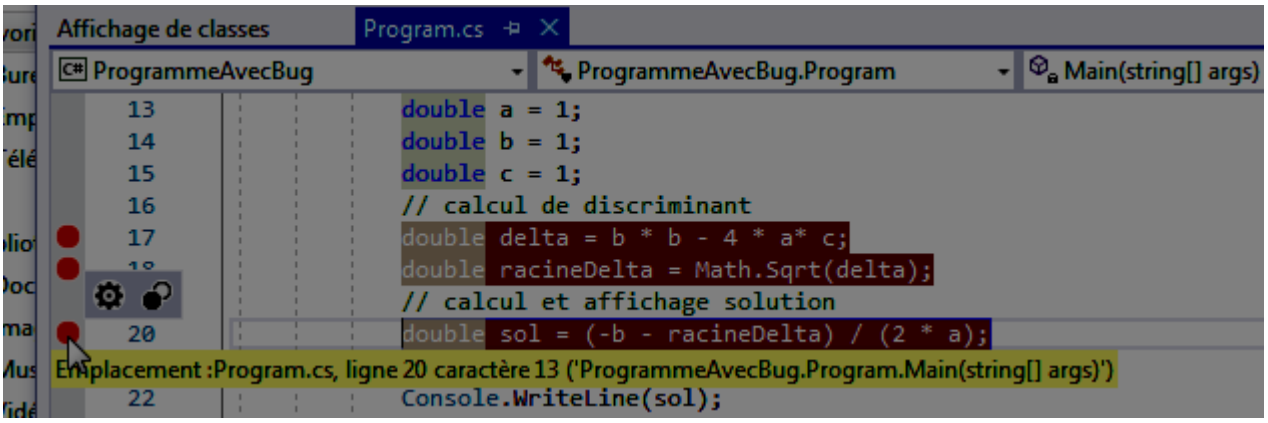

Cas avec erreur



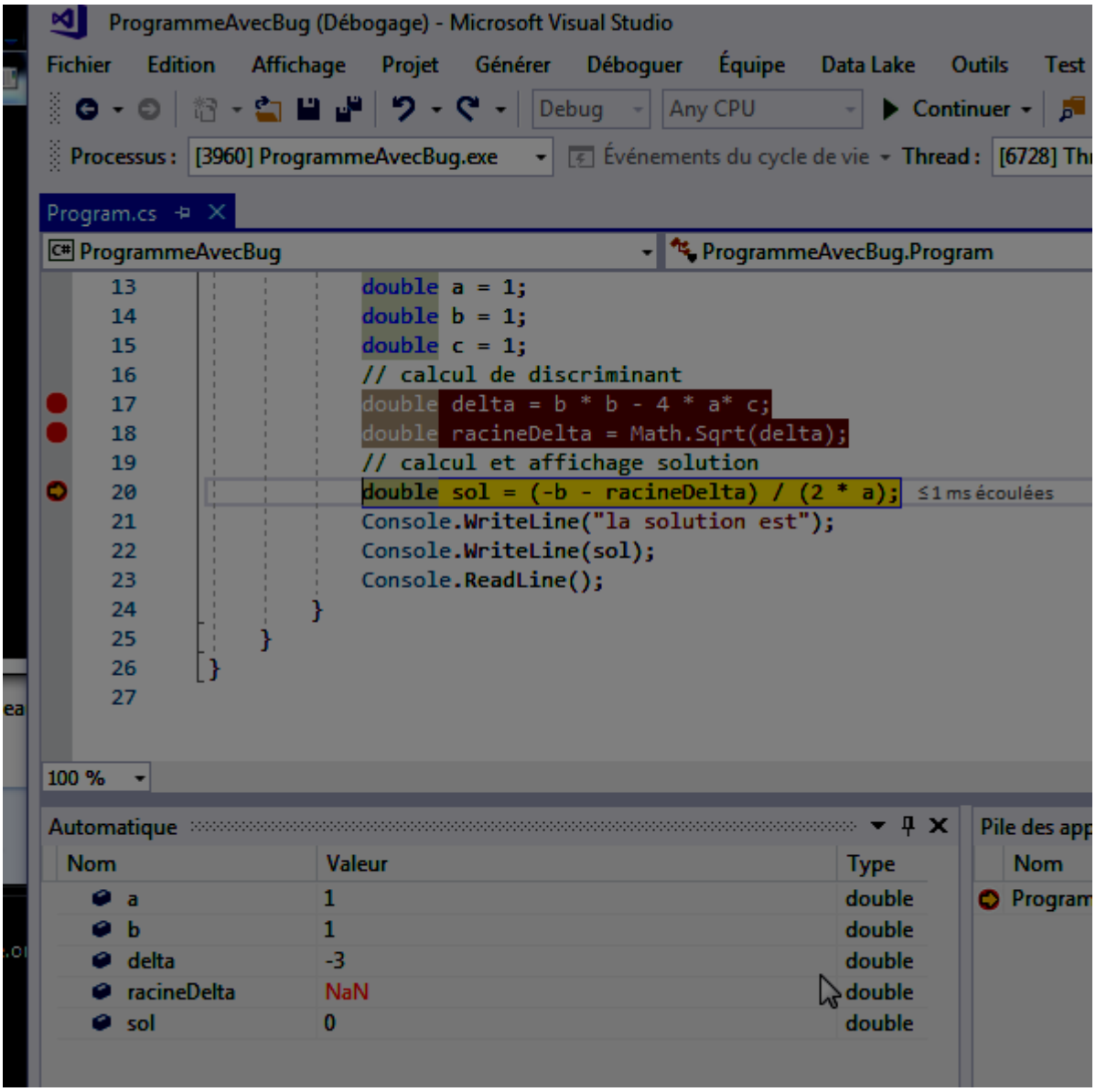


# Tutorial: Memento d'utilisation de Visual

Pour comprendre ce qu'il se passe il faut lancer l'exécution pas à pas pour trouver quelle instruction pose problème. On va donc placer des points d'arrêt dans la zone qui nous semble contenir le problème (calcul du discriminant dans cet exemple). Pour cela, il suffit de cliquer sur la colonne grise en face de la ligne sur laquelle on veut que l'exécution s'arrête.



On lance maintenant l'exécution qui va s'arrêter sur chaque point d'arrêt en affichant le contenu des variables. Quand on calcule la valeur de sol on voit que la valeur de racineDelta est NaN : Not a Number, c'est donc à cet endroit qu'il y a une erreur.



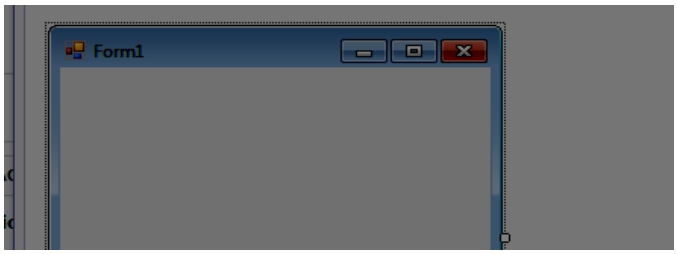

### 3. Création d'un programme en mode graphique (Application Windows Form)

Pour les programmes en mode graphique, vous allez devoir naviguer entre plusieurs fenêtres et menus. Si vous ne voyez pas les fenêtres, aller dans affichage pour ajouter la fenêtre correspondante.
Les fenêtres que vous allez manipuler sont les suivantes :

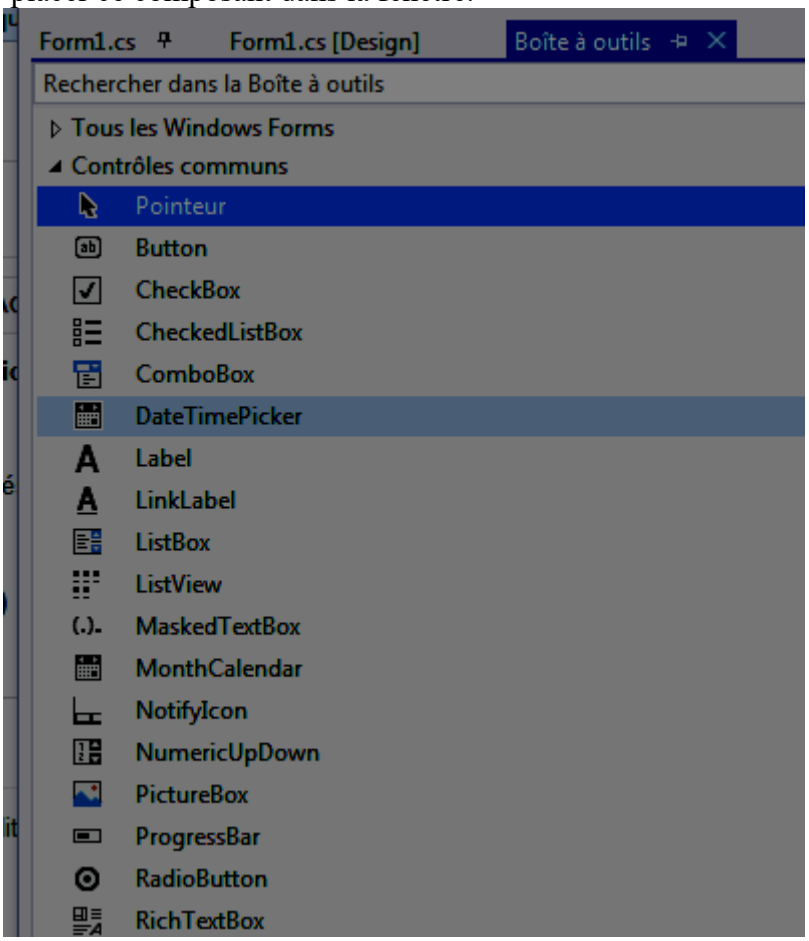
- la fenêtre de **concepteur de vue** : vous permet d'ajouter des composants graphiques et de les disposer dans la fenêtre.



# Tutorial: Memento d'utilisation de Visual

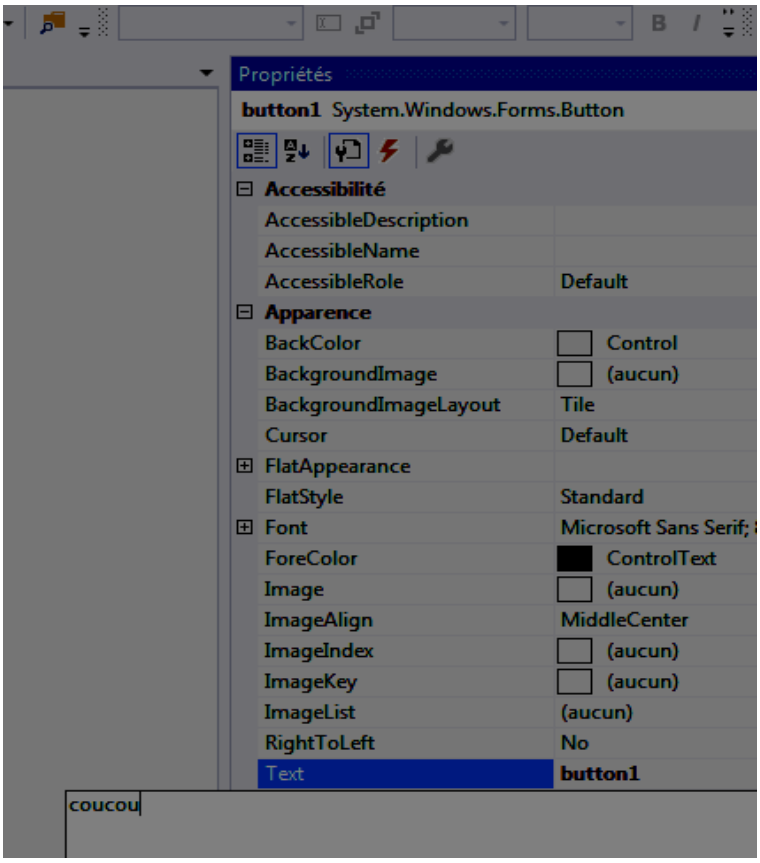



– La **boîte à outils** qui vous permet de sélectionner un composant graphique (exemple : un bouton) et de placer ce composant dans la fenêtre.

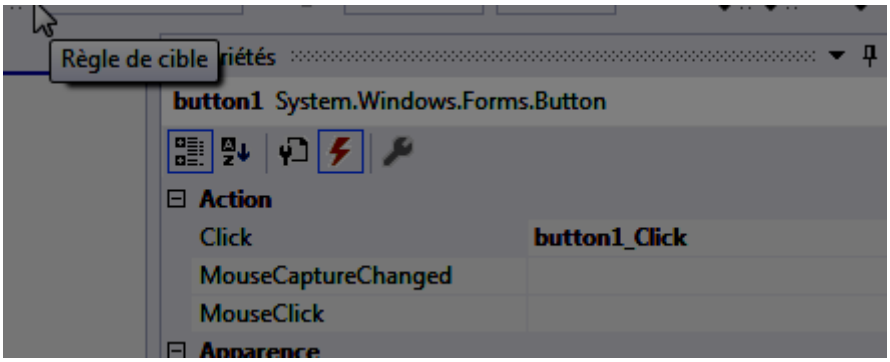



– La fenêtre de **propriété du composant graphique**. Vous permet de changer les propriétés telle la couleur, le texte, la taille, … en cliquant sur la clé à molette

# Tutorial: Memento d'utilisation de Visual

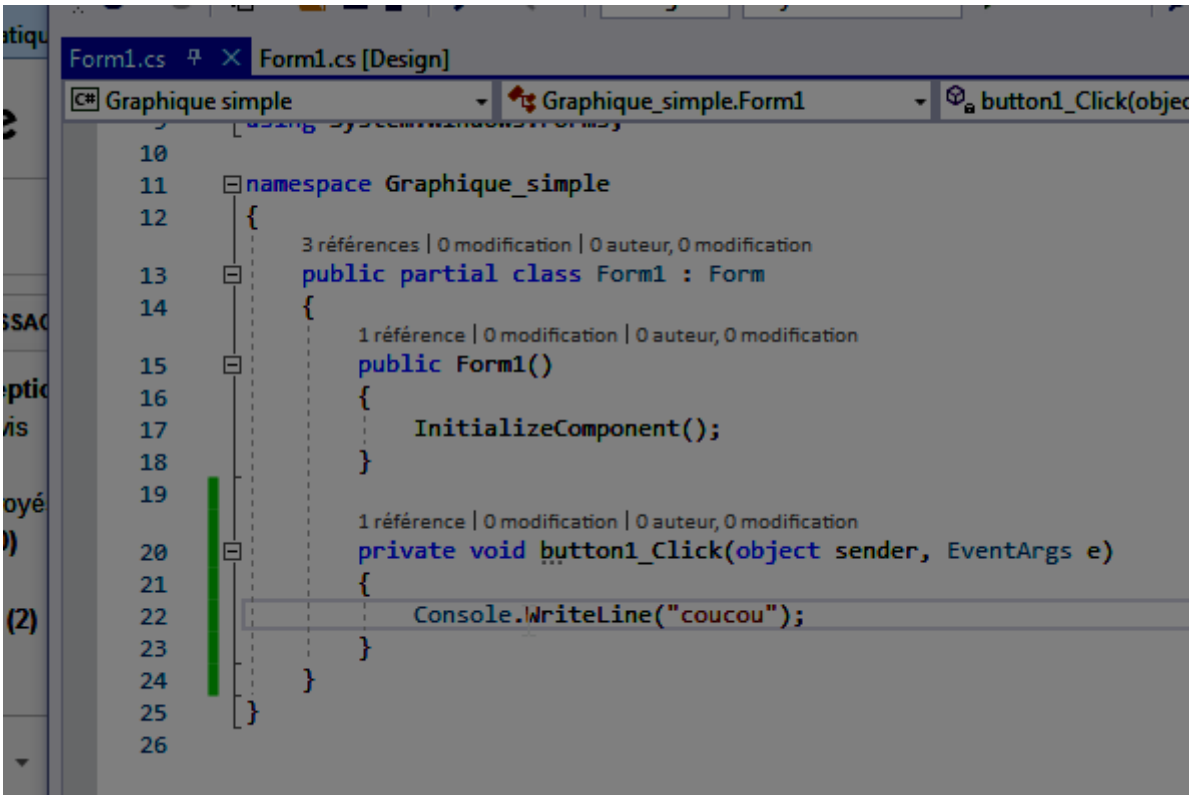



– En sélectionnant l'**éclair dans la fenêtre de propriétés du composant**, vous pouvez associer des actions aux événements. Par exemple, en double cliquant sur Action → click cela créé une fonction qui sera appelée à chaque clic sur le bouton.

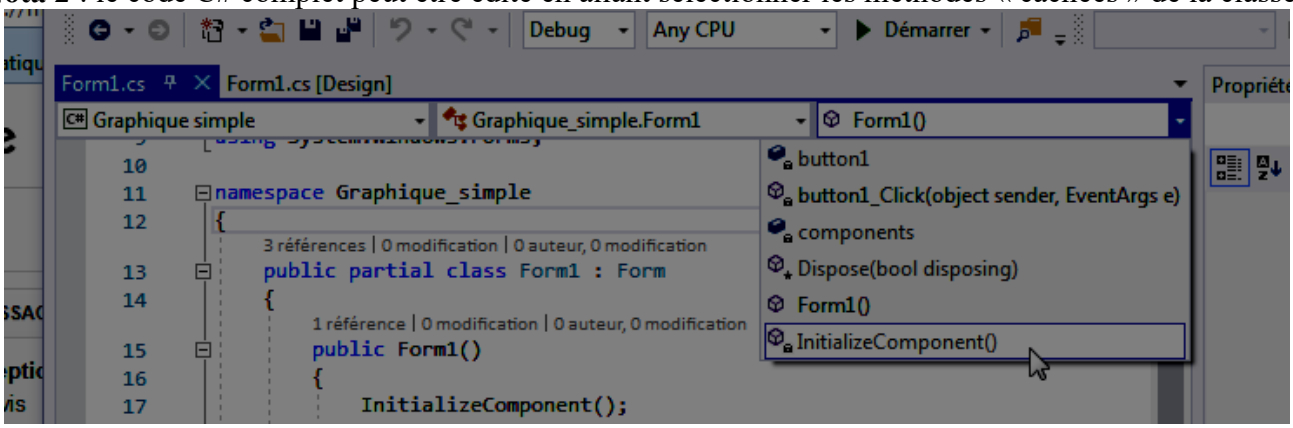



– **La fenêtre de code** où vous pourrez associer une action à un événement sur un composant graphique (exemple, une action à effectuer quand on clique sur un bouton). Par exemple, on affiche « coucou » quand on clique sur le bouton.

# Tutorial: Memento d'utilisation de Visual




**Nota 1** : Form1.cs est le programme C#, Form1.cs[Design] est le concepteur de vue de ce programme, il vous permet de modifier le programme en déplaçant et en ajoutant des composants.
**Nota 2** : le code C# complet peut être édité en allant sélectionner les méthodes « cachées » de la classe.




Cela ouvre une troisième fenêtre Form1.Designer.cs qui contient le code C# complet qui a été généré. Dans certains cas, vous aurez besoin (ou envie) de modifier directement ce code.



# Tutorial: Memento d'utilisation de Visual