


[Accueil](#) > [Cours](#) > [Apprenez à développer en C#](#) > Utilisez le débogueur

Apprenez à développer en C#

20 heures  Facile

Mis à jour le 04/07/2019



Utilisez le débogueur

 [Connectez-vous](#) ou [inscrivez-vous](#) gratuitement pour bénéficier de toutes les fonctionnalités de ce cours !

Nous allons maintenant faire une petite pause. Le C# c'est bien, mais notre environnement de développement, Visual Studio Express, peut faire beaucoup plus que sa fonction basique d'éditeur de fichiers. Il possède un outil formidable qui va nous permettre d'être très efficaces dans le débogage de nos applications et dans la compréhension de leur fonctionnement.

Il s'agit du débogueur. Découvrons vite à quoi il sert et comment il fonctionne

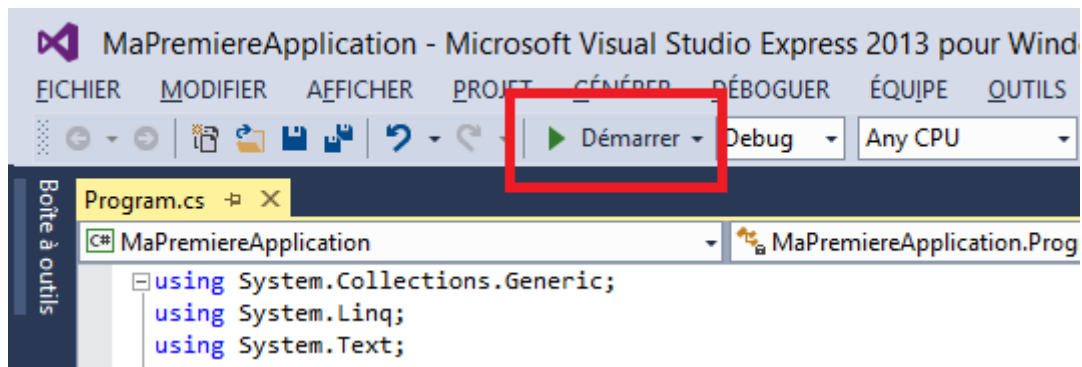
A quoi ça sert ?



Fidèle à son habitude de nous simplifier la vie, Visual Studio Express possède un débogueur. C'est un outil très pratique qui va permettre d'obtenir plein d'informations sur le déroulement de son programme.

Il va permettre d'exécuter les instructions les unes après les autres, de pouvoir observer le contenu des variables, de revenir en arrière, bref, de pouvoir savoir exactement ce qu'il se passe et surtout pourquoi tel bout de code ne fonctionne pas.

Pour l'utiliser, il faut que Visual Studio Express soit en mode « débogage ». Je n'en ai pas encore parlé. Pour ce faire, il faut exécuter l'application en appuyant sur **F5** ou en passant par le menu **Déboguer** > **Démarrer le débogage** ou en cliquant sur le petit triangle vert dont l'icône rappelle un bouton de lecture.



Démarrer le débogage

Pour étudier le débogueur, reprenons la dernière méthode du précédent TP :

csharp

```
1 static void Main(string[] args)
2 {
3     Console.WriteLine(CalculSommeIntersection());
4 }
5
6 static int CalculSommeIntersection()
7 {
8     List<int> multiplesDe3 = new List<int>();
9     List<int> multiplesDe5 = new List<int>();
10
11     for (int i = 1; i <= 100; i++)
12     {
13         if (i % 3 == 0)
14             multiplesDe3.Add(i);
15         if (i % 5 == 0)
16             multiplesDe5.Add(i);
17     }
18
19     int somme = 0;
20     foreach (int m3 in multiplesDe3)
21     {
22         foreach (int m5 in multiplesDe5)
23         {
24             if (m3 == m5)
25                 somme += m3;
26         }
27     }
28     return somme;
29 }
```

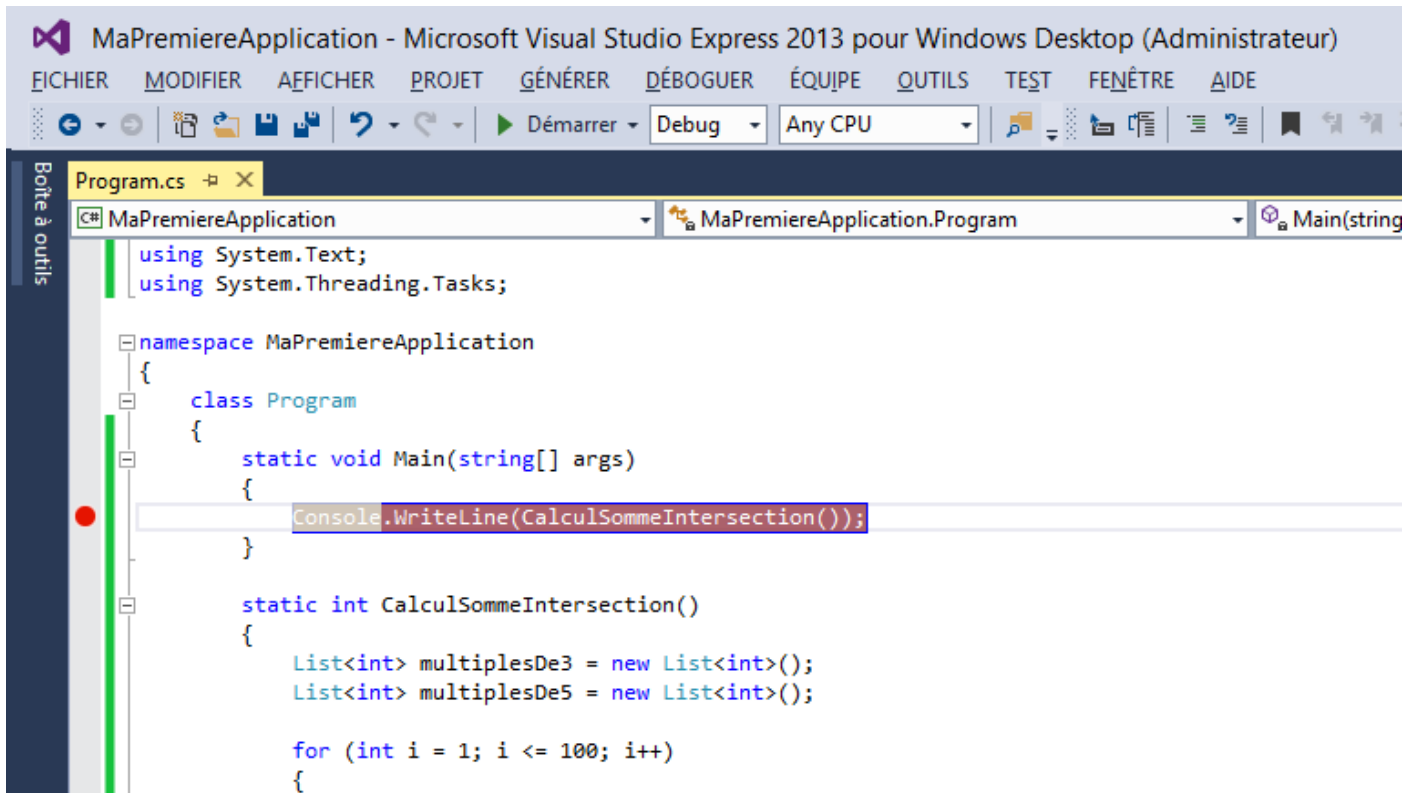
Mettre un point d'arrêt et avancer pas à pas



Pour que le programme s'arrête sur un endroit précis et qu'il nous permette de voir ce qu'il se passe, il va falloir mettre des **points d'arrêts** dans notre code.

Pour mettre un point d'arrêt, il faut se positionner sur la ligne où nous souhaitons nous arrêter, par exemple la première ligne où nous appelons `Console.WriteLine` et appuyer sur **F9**. Nous

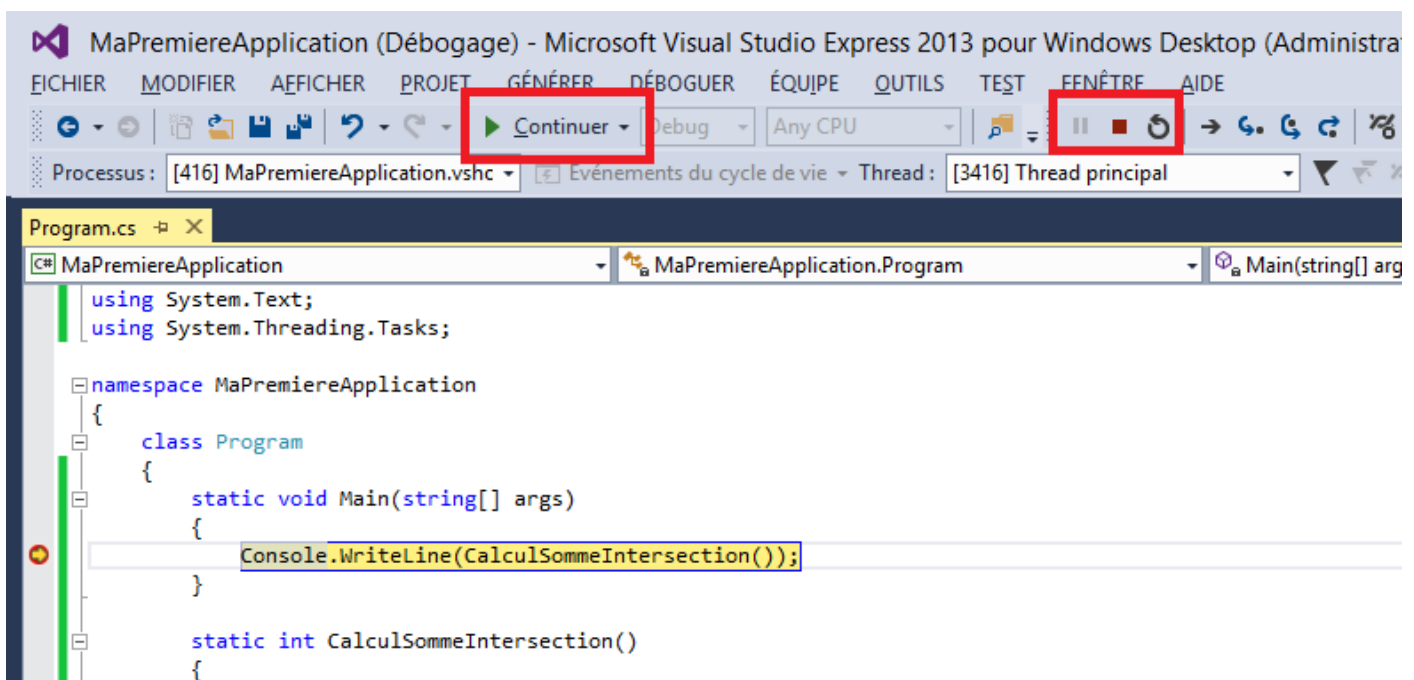
pouvons également cliquer dans la marge à gauche pour produire le même résultat. Un point rouge s'affiche et indique qu'il y a un point d'arrêt à cet endroit.



Mettre un point d'arrêt

Il est possible de mettre autant de points d'arrêts que nous le souhaitons, à n'importe quel endroit de notre code.

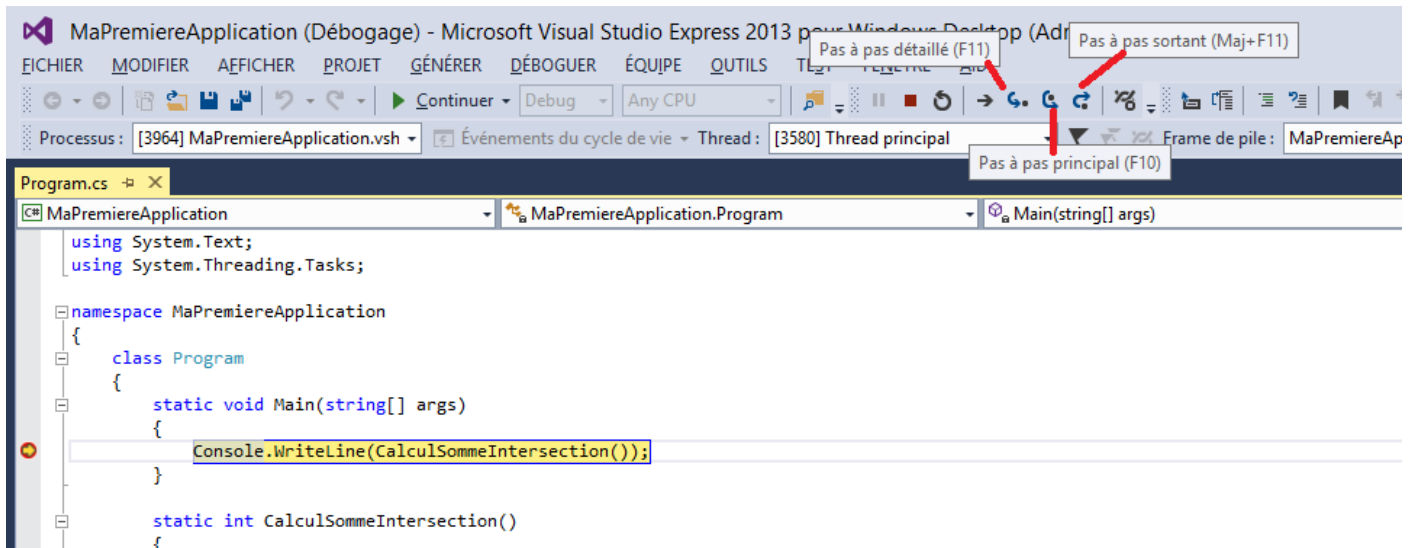
Lançons l'application avec **F5**, nous pouvons voir que Visual Studio Express s'arrête sur la ligne prévue en la surlignant en jaune :



Le débogueur s'arrête sur le point d'arrêt

Nous en profitons pour remarquer au niveau du carré rouge que le débogueur est en pause et qu'il est possible soit de continuer l'exécution (triangle vu précédemment) soit de l'arrêter (carré) soit de redémarrer le programme depuis le début (flèche en forme de cercle).

Nous pouvons désormais exécuter notre code pas à pas, en nous servant des icônes à côté ou des raccourcis claviers suivants :



Navigation dans le code en debug

Utilisons la touche **F10** pour continuer l'exécution du code pas à pas. La ligne suivante se trouve surlignée à son tour. Appuyons à nouveau sur la touche **F10** pour aller à l'instruction suivante, il n'y en a plus le programme se termine.

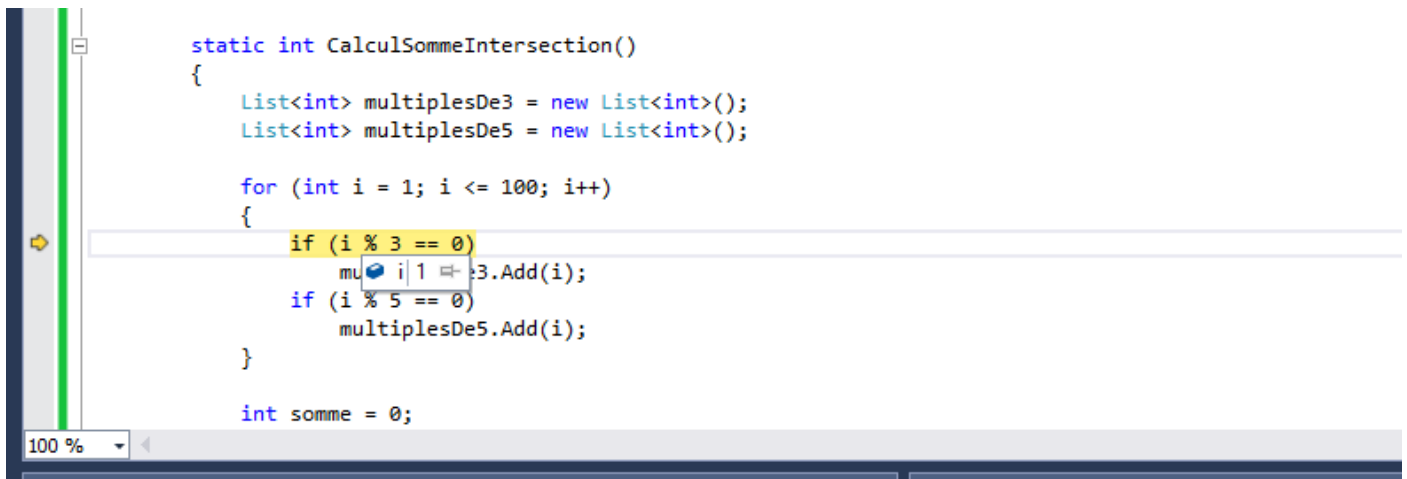
Vous me direz, « c'est bien beau, mais, nous ne sommes pas passés dans la méthode `CalculSommeIntersection()` ». Et oui, c'est parce que nous avons utilisé la touche **F10** qui est le pas à pas principal. Pour rentrer dans la méthode, il aurait fallu utiliser la touche **F11** qui est le pas à pas détaillé.

Si nous souhaitons que le programme se poursuive pour aller jusqu'au prochain point d'arrêt, il suffit d'appuyer sur le triangle (play) ou sur **F5**.

Relançons notre programme en mode débogage, et cette fois-ci, lorsque le débogueur s'arrête sur notre point d'arrêt, appuyons sur **F11** pour rentrer dans le corps de la méthode. Voilà, nous sommes dans la méthode `CalculSommeIntersection()`. Continuons à appuyer plusieurs fois sur **F10** afin de rentrer dans le corps de la boucle `for`.

Observer des variables

À ce moment-là du débogage, si nous passons la souris sur la variable `i`, qui est l'indice de la boucle, Visual Studio Express va nous afficher une mini-information :



Observer la valeur d'une variable pendant le débogage

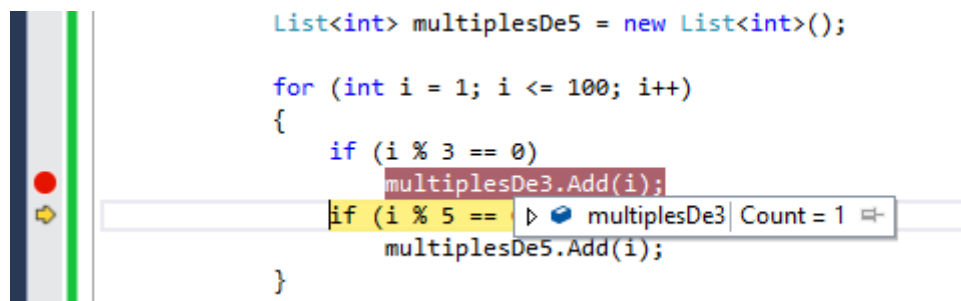
Il nous indique que `i` vaut 1, ce qui est normal, nous sommes dans la première itération de la boucle. Si nous continuons le parcours en appuyant sur **F10** plusieurs fois, nous voyons que la valeur de `i` augmente, conformément à ce qui est attendu.

Maintenant, mettons un point d'arrêt - **F9** - sur la ligne :

csharp

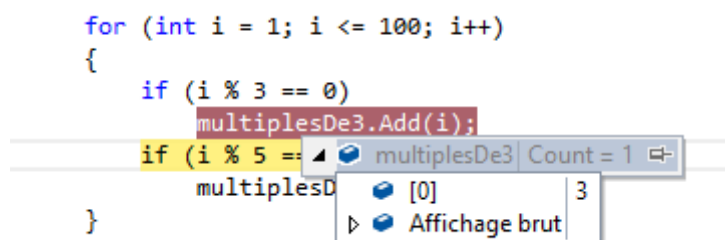
```
1 multiplesDe3.Add(i);
```

et poursuivons l'exécution en appuyant sur **F5**. Il s'arrête au moment où `i` vaut 3. Appuyons sur **F10** pour exécuter l'ajout de `i` à la liste et passons la souris sur la liste :



Inspection d'une liste en debug

Visual Studio Express nous indique que la liste `multiplesDe3` a son « Count » qui vaut 1. Si nous cliquons sur le + pour déplier la liste :



Inspection des valeurs de la liste

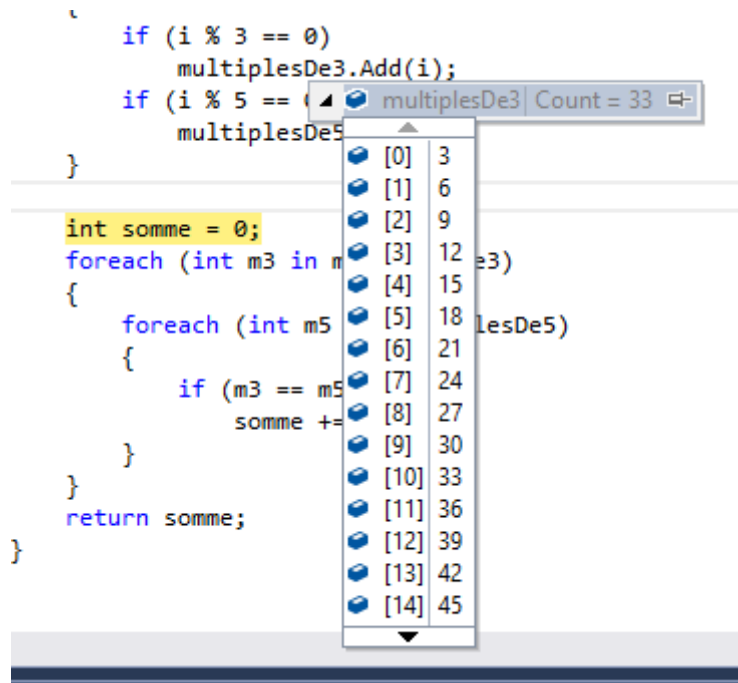
nous pouvons voir que 3 a été ajouté dans le premier emplacement de la liste (indice 0). Si nous continuons l'exécution plusieurs fois, nous voyons que les listes se remplissent au fur et à mesure.

Enlevez le point d'arrêt sur la ligne en appuyant à nouveau sur **F9** et mettez un nouveau point d'arrêt sur la ligne :

csharp

```
1 int somme = 0;
```

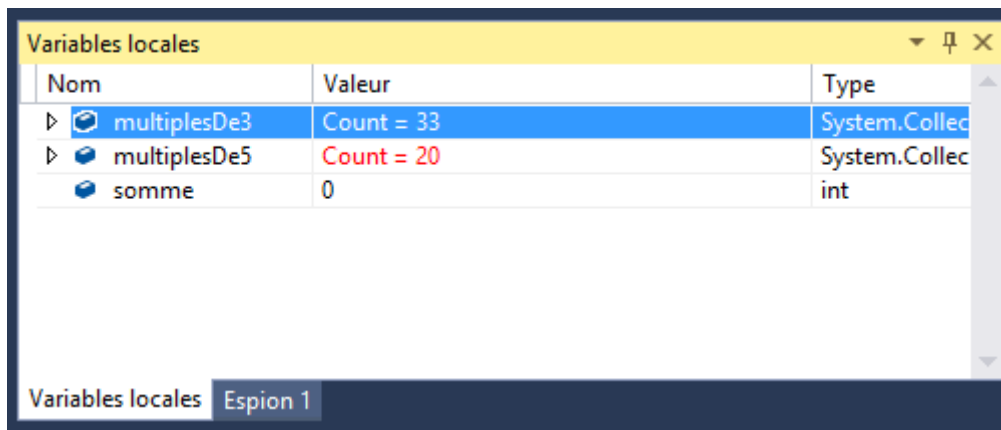
Poursuivez l'exécution avec **F5**, la boucle est terminée, nous pouvons voir que les listes sont complètement remplies :



Affichage des éléments de la liste en debug

Grâce au débogueur, nous pouvons voir vraiment tout ce qui nous intéresse, c'est une des grandes forces du débogueur et c'est un atout vraiment très utile pour comprendre ce qu'il se passe dans un programme (en général, ça se passe mal !).

Il est également possible de voir les variables locales en regardant en bas dans la fenêtre « Variables locales ». Dans cette fenêtre, vous pourrez observer les variables qui sont dans la portée courante. Il existe également une fenêtre « Espion » qui permet de la même façon de surveiller une ou plusieurs variables précises.



La fenêtre qui espionne les variables locales

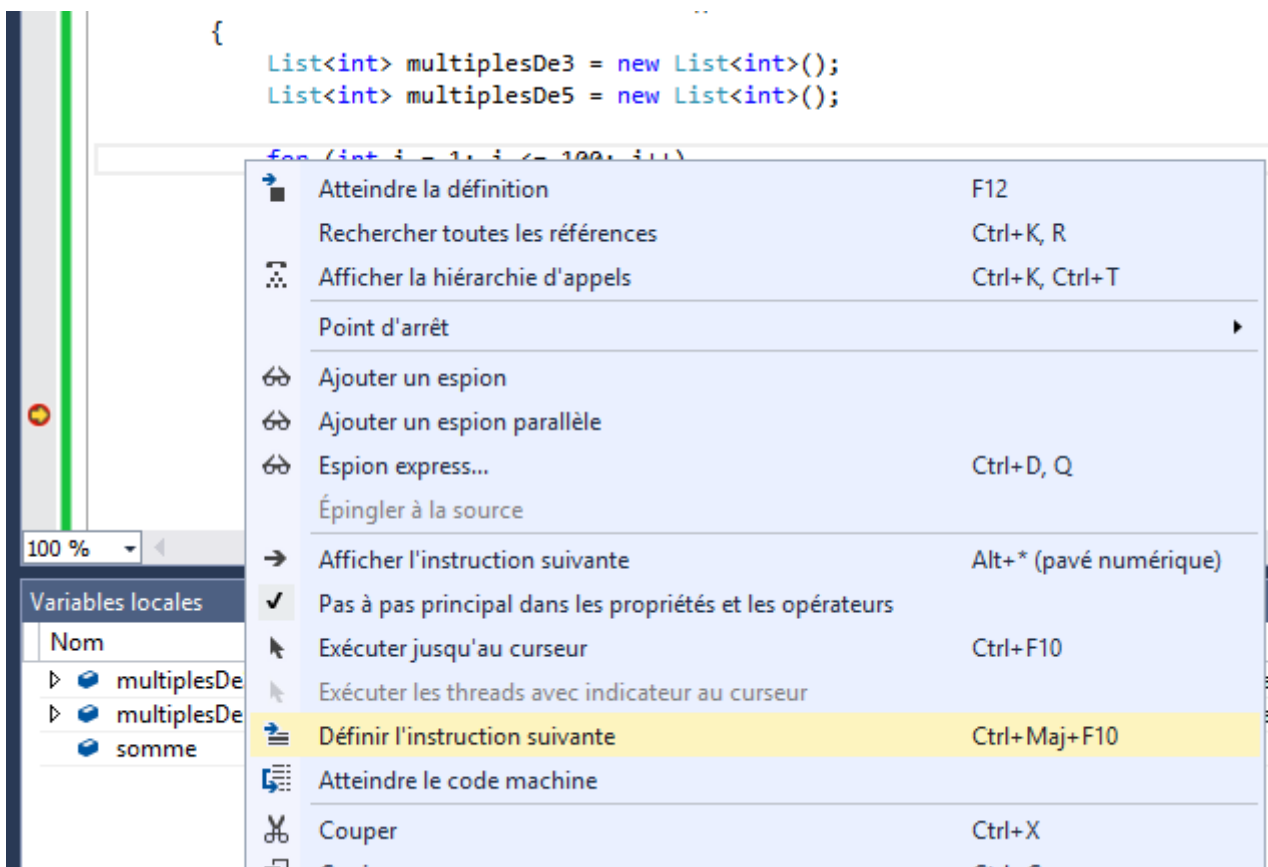
Revenir en arrière



Nom de Zeus, Marty ! On peut revenir dans le passé ?

C'est presque ça Doc ! Si pour une raison, vous souhaitez ré-exécuter un bout de code, parce que vous n'avez pas bien vu ou que vous avez raté l'information qu'il vous fallait, vous pouvez forcer le débogueur à revenir en arrière dans le programme. Pour cela, vous avez deux solutions, soit vous faites un clic droit à l'endroit souhaité et vous choisissez l'élément de menu

Définir l'instruction suivante :



Définir l'instruction suivante

soit vous déplacez avec la souris la petite flèche jaune sur la gauche qui indique l'endroit où nous en sommes.



```
    multiplesDe3.Add(m3);  
}  
  
int somme = 0;  
foreach (int m3 in multiplesDe3)  
{
```

Déplacer la flèche jaune pour changer la ligne à exécuter par le débogueur

Il faut faire attention car ce retour en arrière n'en est pas complètement un. En effet, si vous revenez au début de la boucle qui calcule les multiples et que vous continuez l'exécution, vous verrez que la liste continue à se remplir. À la fin de la boucle, au lieu de contenir 33 éléments, la liste des multiples de 3 en contiendra 66. En effet, à aucun moment nous n'avons vidé la liste et donc le fait de ré-exécuter cette partie de code risque de provoquer des comportements inattendus. Ici, il vaudrait mieux revenir au début de la méthode afin que la liste soit de nouveau créée.

Même si ça ne vous saute pas aux yeux pour l'instant, vous verrez à l'usage que cette possibilité est bien pratique. D'autant plus quand vous aurez bien assimilé toutes les notions de portée de variable.

Il est important de noter que, bien que puissant, le débogueur de la version gratuite de Visual Studio est vraiment moins puissant que celui des versions payantes. Il y a plein de choses en moins que l'on ne peut pas faire.

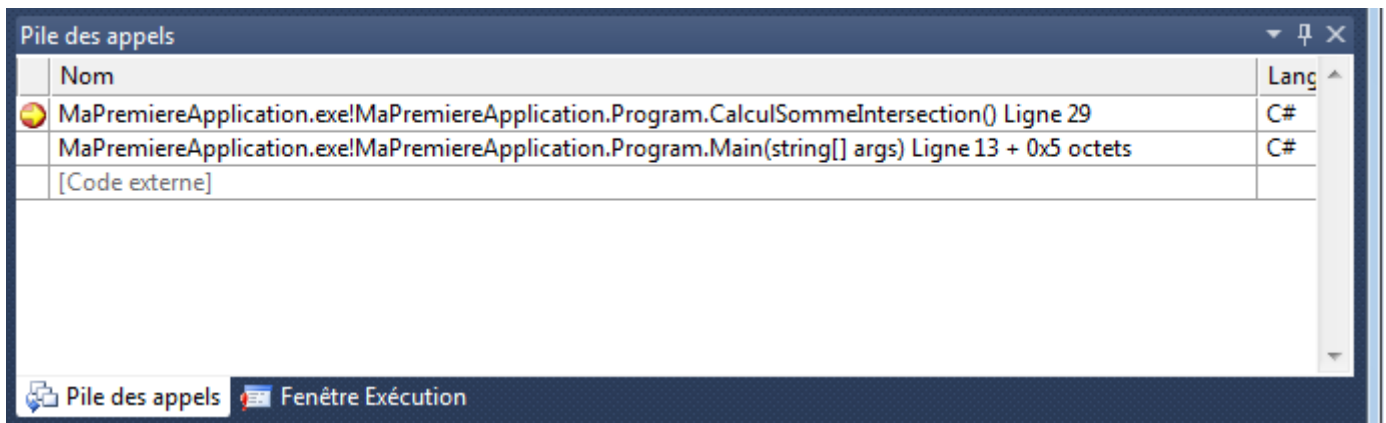
Mais rassurez-vous, celui-ci est quand même déjà bien avancé et permet de faire beaucoup de choses.

La pile des appels

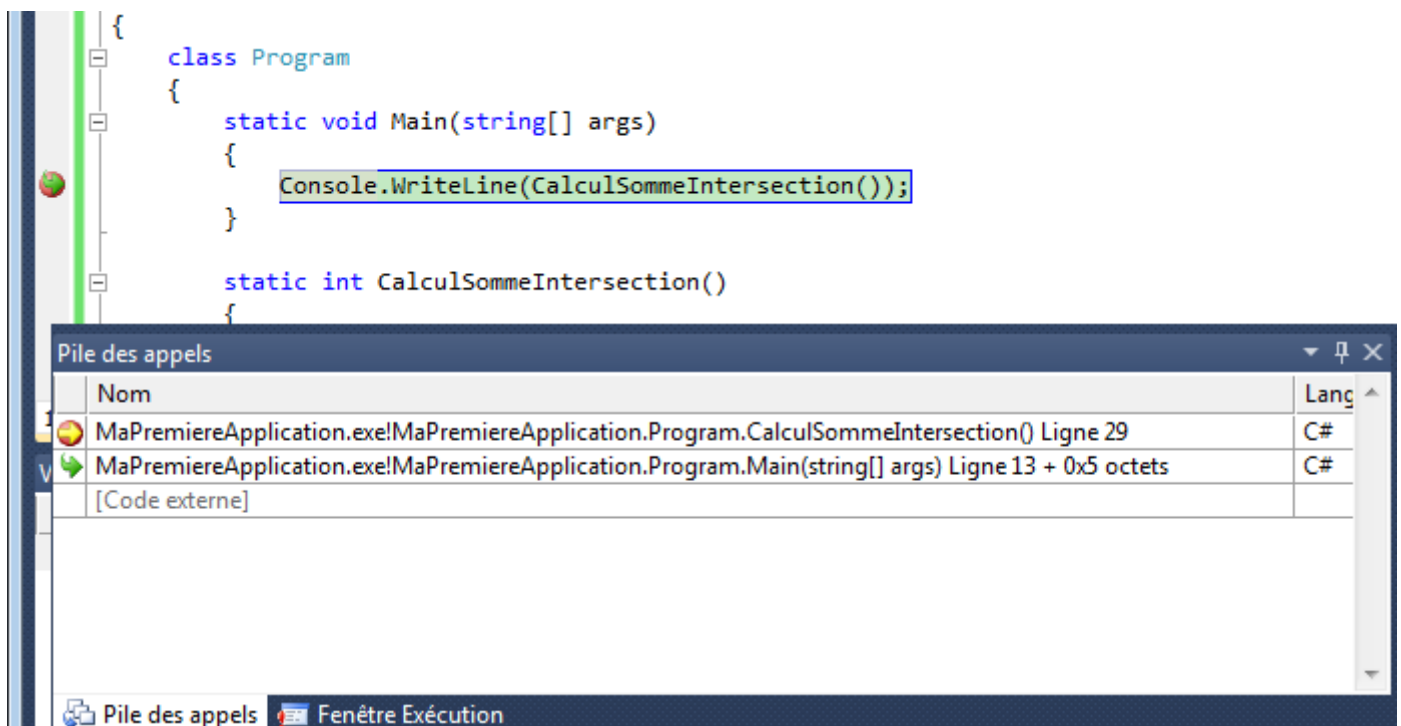


Une autre fenêtre importante à regarder est la pile des appels, elle permet d'indiquer où nous nous trouvons et par où nous sommes passés pour arriver à cet endroit-là.

Par exemple si vous appuyez sur **F11** et que vous rentrez dans la méthode `CalculSommeIntersection()` vous pourrez voir dans la pile des appels que vous êtes dans la méthode `CalculSommeIntersection()` qui a été appelée depuis la méthode `Main()` :



Si vous cliquez sur la ligne du `Main()`, Visual Studio Express vous ramène automatiquement à l'endroit où a été fait l'appel. Cette ligne est alors surlignée en vert pour bien faire la différence avec le surlignage en jaune qui est vraiment l'endroit où se trouve le débogueur. C'est très pratique quand on a beaucoup de méthodes qui s'appellent les unes à la suite des autres.



La pile des appels est également affichée lorsqu'on rencontre une erreur, elle permettra d'identifier plus facilement où est le problème. Vous l'avez vu dans le chapitre sur les conversions entre les types incompatibles.

En tous cas, le débogueur est vraiment un outil à forte valeur ajoutée. Je ne vous ai présenté que le strict minimum nécessaire et indispensable. Mais croyez-moi, vous aurez l'occasion d'y revenir 😊.

En résumé

- Le débogueur est un outil très puissant permettant d'inspecter le contenu des variables lors de l'exécution d'un programme.
- On peut s'arrêter à un endroit de notre application grâce à un point d'arrêt.
- Le débogueur permet d'exécuter son application pas à pas et de suivre son évolution.

[← LISEZ LE CLAVIER DANS LA CONSOLE](#)[TP : LE JEU DU PLUS OU DU MOINS >](#)

Le professeur

Nicolas Hilaire

Expert .NET, artisan logiciel, plusieurs fois honoré du titre Microsoft MVP, mais également curieux des autres technologies.

Découvrez aussi ce cours en...



Livre



PDF

OpenClassrooms

L'entreprise

Alternance

Forum

Blog

Nous rejoindre

Entreprises

Business

En plus


Devenez mentor

Aide et FAQ

Conditions Générales d'Utilisation

Politique de Protection des Données Personnelles

Nous contacter

 Français ▼

