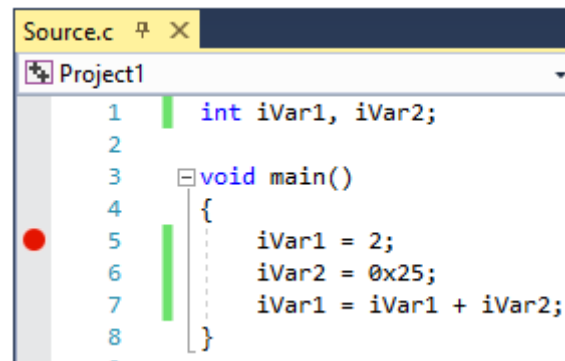


Introduction au débogueur de Visual Studio

1. Après avoir créé un projet, encodez le programme suivant et placez un **point d'arrêt** sur l'instruction `iVar1 = 2` :

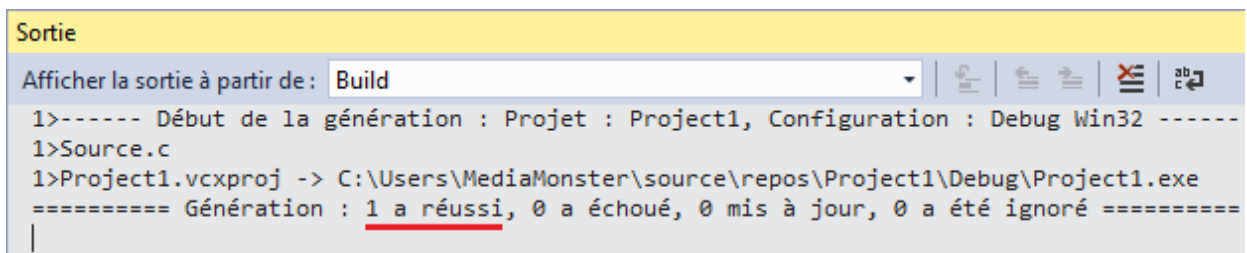


```
Source.c 7 X
Project1
1  int iVar1, iVar2;
2
3  void main()
4  {
5      iVar1 = 2;
6      iVar2 = 0x25;
7      iVar1 = iVar1 + iVar2;
8  }
```

A red dot indicating a breakpoint is placed at the beginning of line 5, which contains the instruction `iVar1 = 2;`.

On place le point d'arrêt (rond rouge) en cliquant au début de la ligne 5. Le point d'arrêt est l'endroit où l'exécution du programme va devoir temporairement s'interrompre, afin de permettre au programmeur de visualiser la valeur stockée dans certaines variables.

2. Générer le fichier exécutable en sélectionnant l'option **Générer la solution** dans le menu **Générer**. On sait si la génération du fichier exécutable a réussi en observant le message affiché dans la fenêtre Sortie :



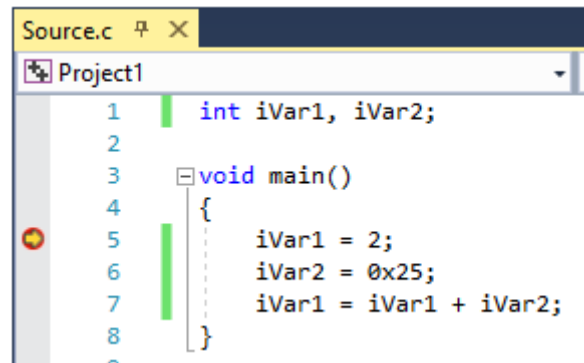
```
Sortie
Afficher la sortie à partir de : Build
1>----- Début de la génération : Projet : Project1, Configuration : Debug Win32 -----
1>Source.c
1>Project1.vcxproj -> C:\Users\MediaMonster\source\repos\Project1\Debug\Project1.exe
===== Génération : 1 a réussi, 0 a échoué, 0 mis à jour, 0 a été ignoré =====
|
```

The output window shows the build process for Project1. The line `Génération : 1 a réussi` is underlined in red, indicating a successful build.

3. Pour démarrer l'exécution du programme dans le débogueur, sélectionnez l'option **Démarrer le débogage** dans le menu **Débogueur**, ou appuyez sur la touche F5, ou encore cliquez sur la flèche verte dans la barre d'icônes.

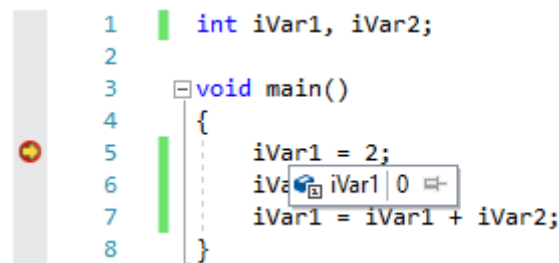
▶ Débogueur Windows local

L'exécution du programme est à présent interrompue sur le point d'arrêt. On voit en effet une flèche jaune sur l'instruction `iVar1 = 2`. Cette instruction sera la **prochaine** instruction exécutée dès la poursuite de l'exécution du programme.



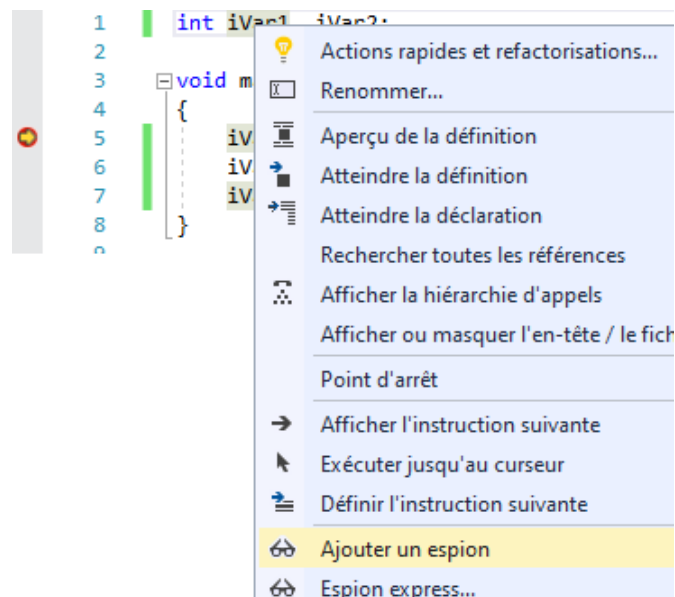
```
Source.c
Project1
1  int iVar1, iVar2;
2
3  void main()
4  {
5      iVar1 = 2;
6      iVar2 = 0x25;
7      iVar1 = iVar1 + iVar2;
8  }
```

4. Pour visualiser le contenu de la variable `iVar1`, déplacez le pointeur de la souris sur le nom de cette variable sans cliquer dessus. Une petite fenêtre apparaît alors temporairement dans laquelle figure la valeur que stocke cette variable.




```
1  int iVar1, iVar2;
2
3  void main()
4  {
5      iVar1 = 2;
6      iVar2 = 0x25;
7      iVar1 = iVar1 + iVar2;
8  }
```

5. Il est possible de choisir explicitement les variables dont on veut suivre l'évolution de la valeur. Pour ce faire, cliquez avec le bouton droit sur la variable concernée, par exemple `iVar1`, puis sélectionnez l'option **Ajouter un espion**.




```
1  int iVar1, iVar2;
2
3  void main()
4  {
5      iVar1 = 2;
6      iVar2 = 0x25;
7      iVar1 = iVar1 + iVar2;
8  }
```

Une fenêtre appelée **Espion 1** apparaît. Elle montre la valeur courante en décimal de la variable iVar1.

Espion 1	
Nom	Valeur
 iVar1	0

6. On peut aussi ajouter explicitement le nom d'une variable dans la fenêtre Espion 1 ainsi :



Espion 1	
Nom	Valeur
 iVar1	0
iVar2	

7. Dans la fenêtre du fichier source, appuyez à 2 reprises sur F10 pour exécuter les 2 instructions iVar1 = 2 et iVar2 = 0x25. La flèche jaune se trouve à présent sur la ligne 7, tandis que la fenêtre Espion 1 montre le contenu actuel des variables iVar1 et iVar2.

Source.c

Project1

```
1 int iVar1, iVar2;
2
3 void main()
4 {
5     iVar1 = 2;
6     iVar2 = 0x25;
7     iVar1 = iVar1 + iVar2;
8 }
```

Espion 1	
Nom	Valeur
 iVar1	2
 iVar2	37



Dans le code source, la valeur 0x25 assignée à iVar2 est exprimée en hexadécimal. Elle correspond à la valeur 37 en décimal.

8. Appuyez sur F10 une seule fois pour exécuter l'instruction iVar1 = iVar1 + iVar2. La fenêtre Espion 1 montre que la variable iVar1 stocke actuellement la valeur 39.

Source.c

Project1

```
1 int iVar1, iVar2;
2
3 void main()
4 {
5     iVar1 = 2;
6     iVar2 = 0x25;
7     iVar1 = iVar1 + iVar2;
8 }
```

Espion 1	
Nom	Valeur
 iVar1	39
 iVar2	37

Note générale : on peut sans problème placer plusieurs points d'arrêt dans un programme. Depuis l'endroit où se trouve la flèche jaune, pour directement atteindre le prochain point d'arrêt placé dans le programme, il suffit d'appuyer sur F5 ou de cliquer sur la flèche verte dans la barre d'icônes.