

# Le débogueur – Bloc F



# Le débogueur

---

Le débogueur est un outil interne permettant au programmeur d'être en immersion dans le code **en temps réel**.

Wikipedia:

*« Il permet d'exécuter le programme pas-à-pas — c'est-à-dire le plus souvent ligne par ligne —, d'afficher la valeur des variables à tout moment et de mettre en place des points d'arrêt sur des conditions ou sur des lignes du programme. »*

Source: <https://fr.wikipedia.org/wiki/D%C3%A9bogueur>

# Le débogueur

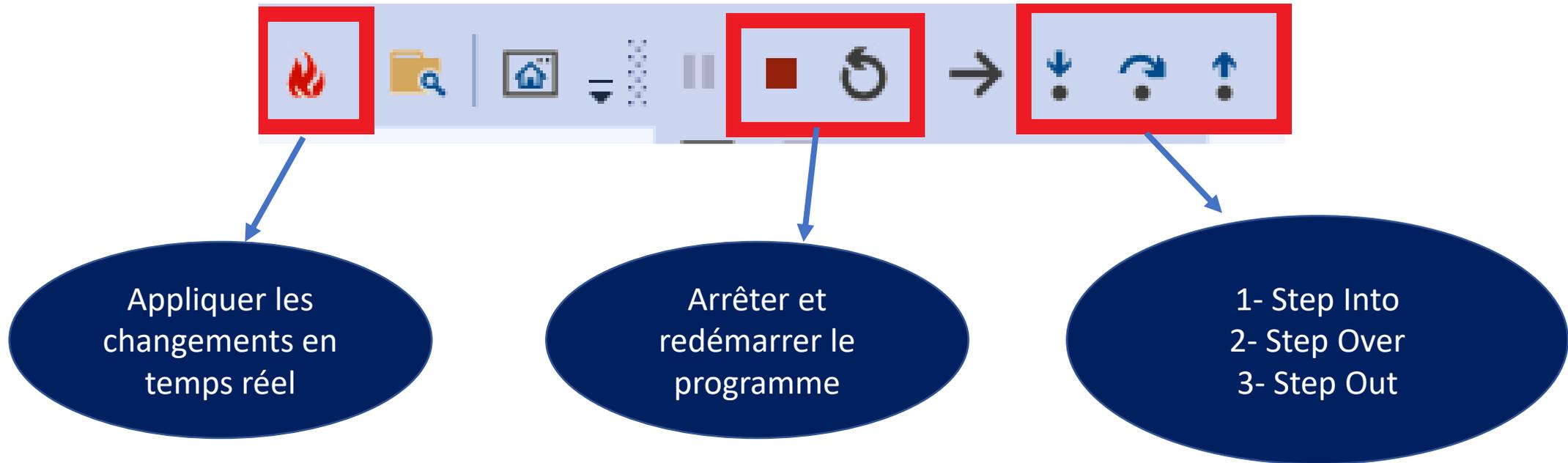
Objectifs d'apprentissage:



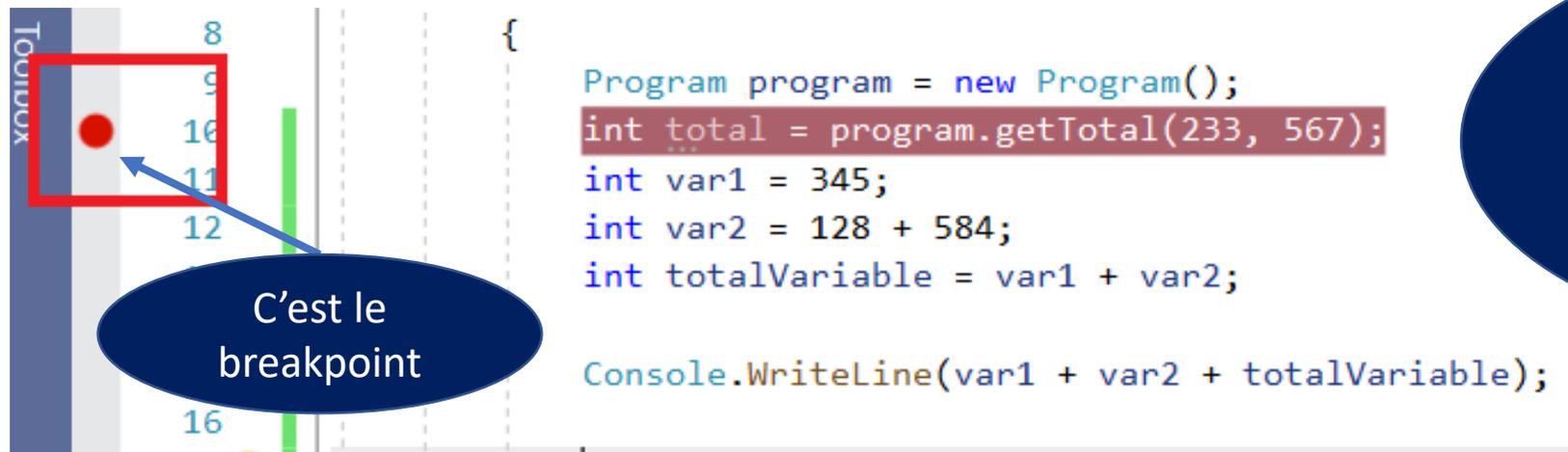
Les breakpoints	Le curseur
Add Watch	Step Into (F11)
Step Over (F10)	Step Out (Shift + F11)
Stop debugging	Apply code change

# Le débogueur

---



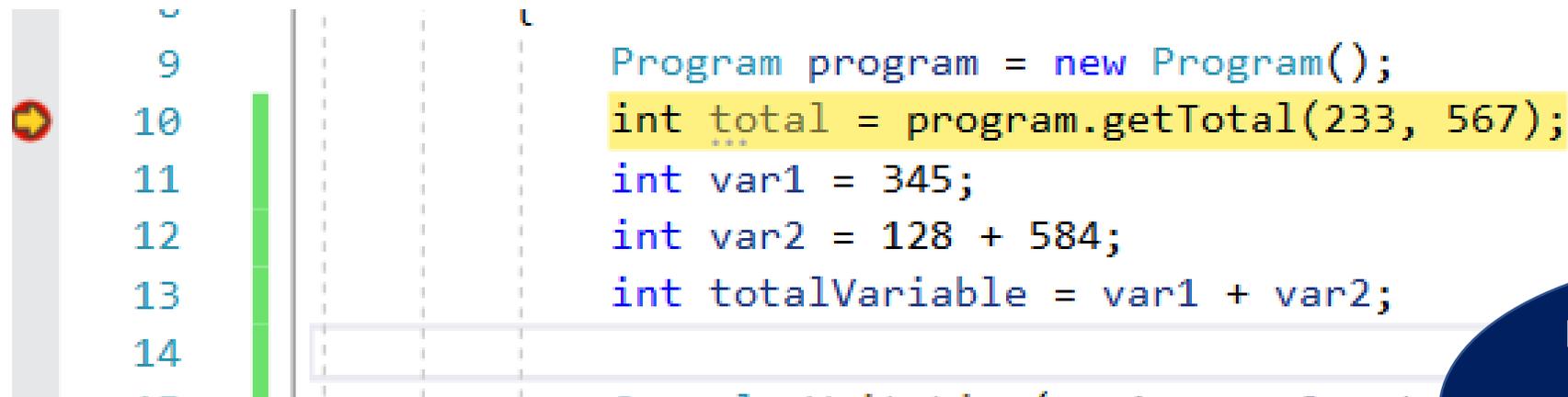
# Les points d'arrêt



L'ajout d'un point d'arrêt (ou breakpoint en anglais) se fait simplement en cliquant dans la ligne grise à l'endroit désirée.

# Les points d'arrêt

Lors de l'exécution, le système va arrêter au point d'arrêt et attendre vos instructions.



```
9  
10  
11  
12  
13  
14  
--
```

```
Program program = new Program();  
int total = program.getTotal(233, 567);  
int var1 = 345;  
int var2 = 128 + 584;  
int totalVariable = var1 + var2;
```

Le système a arrêté à la ligne 10 comme nous l'avons spécifié à l'aide du breakpoint

# Le curseur

---

Le débogueur permet au programmeur d'observer et de contrôler le déroulement d'un programme. Il peut ainsi observer, arrêter ou mettre en pause les instructions de l'application.

Ainsi, il est possible d'observer les variables et leur contenu pendant le flow d'exécution. Ceci permet de facilement déterminer les causes des défaillances.

```
1  
    Program program = new Program();  
2 ▶ int total = program.getTotal(233, 567);  
3 Console.WriteLine("retour de la fonction est de:{0}", total);  
4 }
```

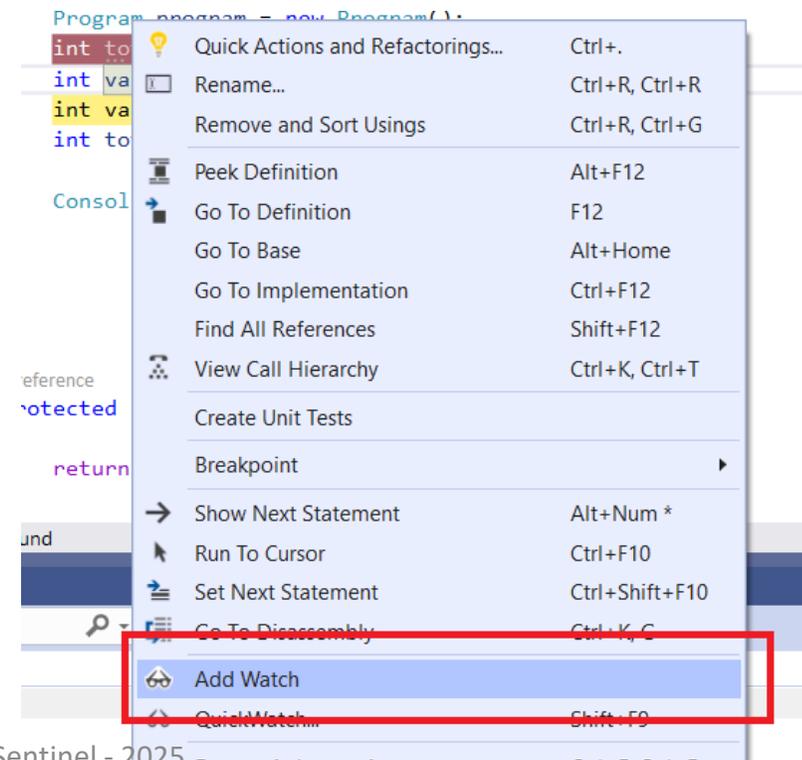
Dans l'exemple on observe la valeur de la variable total. En positionnant la souris sur la variable le débogueur nous informe que la valeur actuelle est de 800

# Utiliser l'inspecteur

Les outils *Add Watch* permettent d'ajouter des surveillances sur plusieurs éléments. Ainsi, nous pouvons avoir une vision d'ensemble et examiner les éléments voulus au fur et à mesure que l'exécution se déroule.

## Utilisation de Add Watch:

- 1- Positionnez le curseur sur une variable
- 2- Bouton de droit de la souris
- 3- Cliquez sur Add Watch



# Utiliser l'inspecteur

## Exemple d'exécution Add Watch

```
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
  
{  
  Program program = new Program();  
  int total = program.getTotal(233, 567);  
  int var1 = 345; ≤ 1ms elapsed  
  int var2 = 128 + 584;  
  int totalVariable = var1 + var2;  
  
  Console.WriteLine(var1 + var2 + totalVariable);  
  
}  
  
1 reference  
protected int getTotal(int x, int y)  
{  
  return x + y;  
}
```

À l'aide du bouton de droit de la souris ajouter les trois variables à Add Watch

Les variables sont dans la fenêtre Watch et on peut voir les valeurs qui leurs sont attribuées

100 % No issues found

Watch 1

Search (Ctrl+E) Search Depth: 3

Name	Value
var1	0
var2	0
totalVariable	0

Add item to watch

# Utiliser l'inspecteur

## Exemple d'exécution Add Watch

The screenshot shows a C# program being executed. The code in the editor is as follows:

```
Program program = new Program();
int total = program.getTotal(233, 567);
int var1 = 345;
int var2 = 128 + 584;
int totalVariable = var1 + var2;

Console.WriteLine(var1 + var2 + totalVariable);
```

The execution is paused at line 15. The Watch window at the bottom shows the following data:

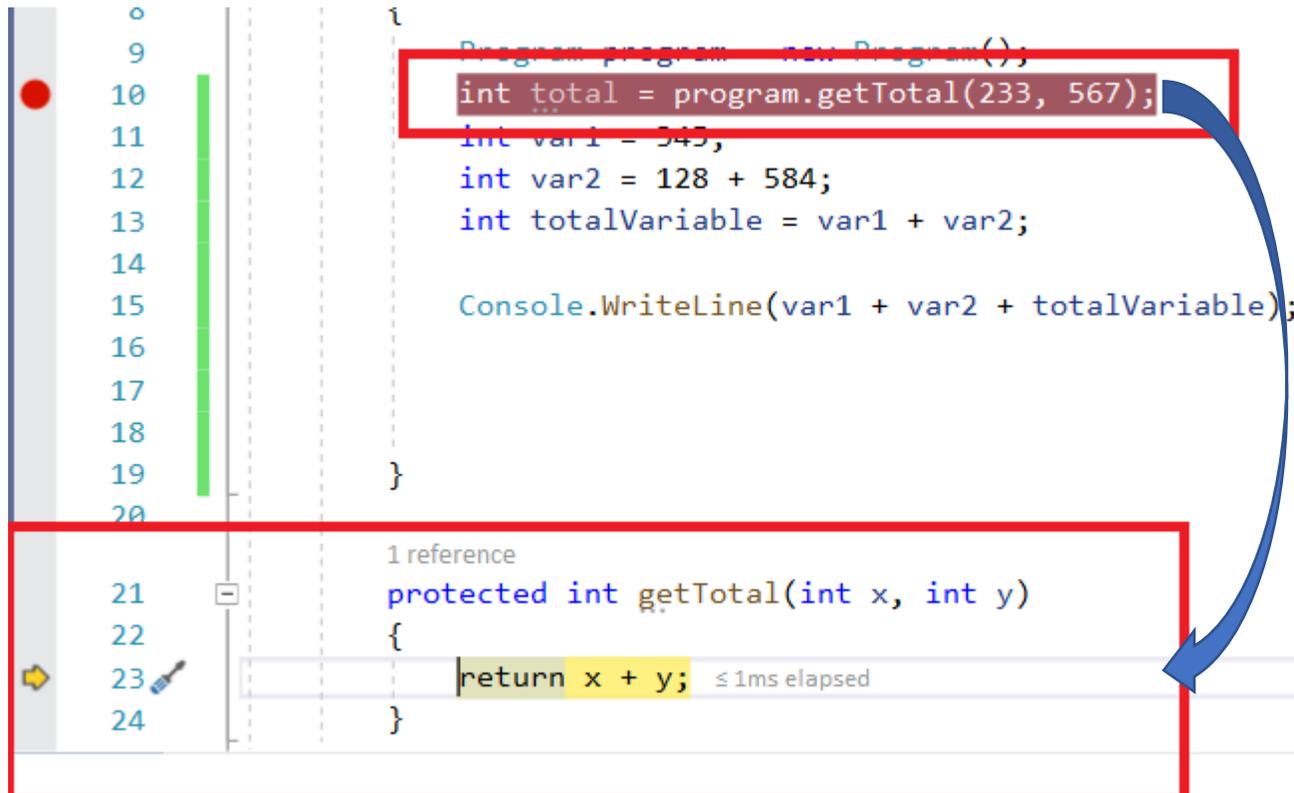
Name	Value	Type
var1	345	int
var2	712	int
totalVariable	1057	int

A blue arrow points from the text in the dark blue oval to the 'var2' row in the Watch window.

Après exécution des instructions 10 à 15 nous avons les valeurs des variables qui ont été ajoutées dans le Watch

# Step Into (F11)

La fonction Step Into vous permet d'entrer dans un block de code.

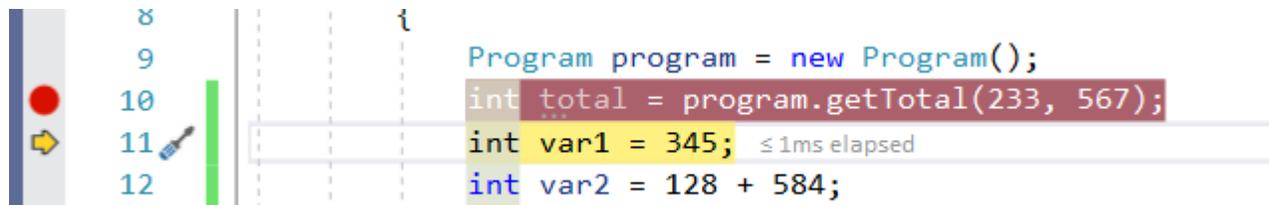


```
0
9
10 Program program = new Program();
11 int total = program.getTotal(233, 567);
12 int var1 = 545;
13 int var2 = 128 + 584;
14 int totalVariable = var1 + var2;
15
16 Console.WriteLine(var1 + var2 + totalVariable);
17
18
19
20
21 1 reference
22 protected int getTotal(int x, int y)
23 {
24     return x + y; ≤ 1ms elapsed
25 }
```

Le Step Into (F11) vous permet d'entrer dans la fonction plutôt que de passer par-dessus.

# Step Over (F10)

La fonction Step Over vous permet de ne pas entrer dans une fonction. S'il n'est pas nécessaire d'entrer dans la fonction pour valider le comportement de cette dernière alors il ne suffit que d'appuyer sur Step Over.

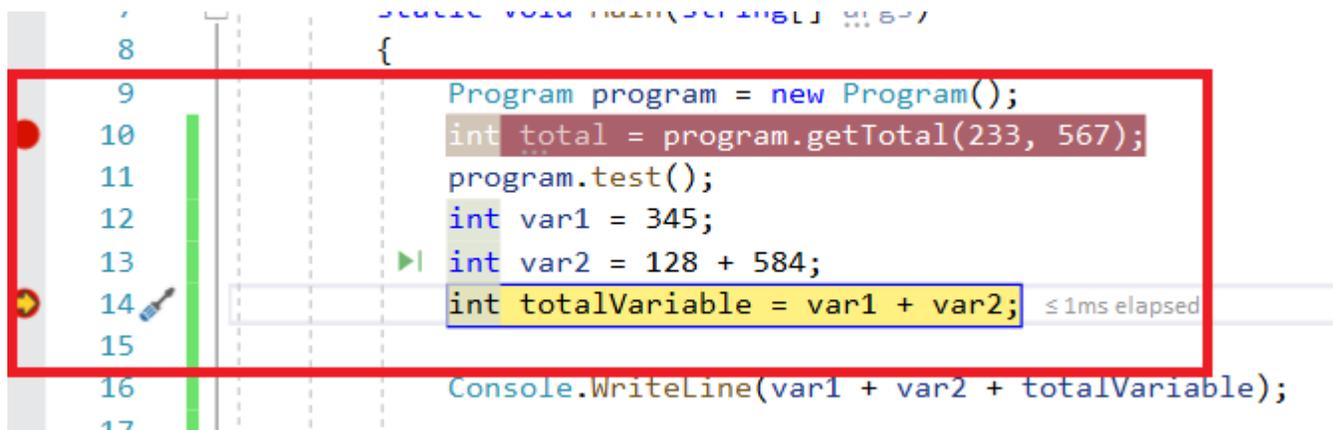


```
8  
9  
10 Program program = new Program();  
11 int total = program.getTotal(233, 567);  
12 int var1 = 345; ≤ 1ms elapsed  
   int var2 = 128 + 584;
```

Le Step Over (F10) vous permet de ne pas entrer dans la fonction appelée à la ligne 10. ATTENTION, la ligne 10 s'exécute mais on n'entre pas à l'intérieur

# Step Out (Shift + F11)

La fonction *Step Out* vous permet redonner le contrôle au programme aussi longtemps qu'il ne rencontre pas un autre point d'arrêt.



```
7
8
9 Program program = new Program();
10 int total = program.getTotal(233, 567);
11 program.test();
12 int var1 = 345;
13 int var2 = 128 + 584;
14 int totalVariable = var1 + var2; ≤ 1ms elapsed
15
16 Console.WriteLine(var1 + var2 + totalVariable);
17
```

Le système a arrêté au premier *breakpoint* et en appuyant sur Step Out le système continue l'exécution jusqu'au prochain *breakpoint*. S'il n'y a pas d'autres *breakpoint* le système continue de s'exécuter normalement

# Debugueur

# Debugging C# Code

Le débogueur sera l'un de vos principaux outils de travail. Vous ne pourrez pas vous en passer et il vous sera d'une très grande utilité!

Apprivoisez-le et utilisez-le régulièrement!

# Questions ?

---

Écrivez-moi à [info@29a.ca](mailto:info@29a.ca)

