

# Modo Debug

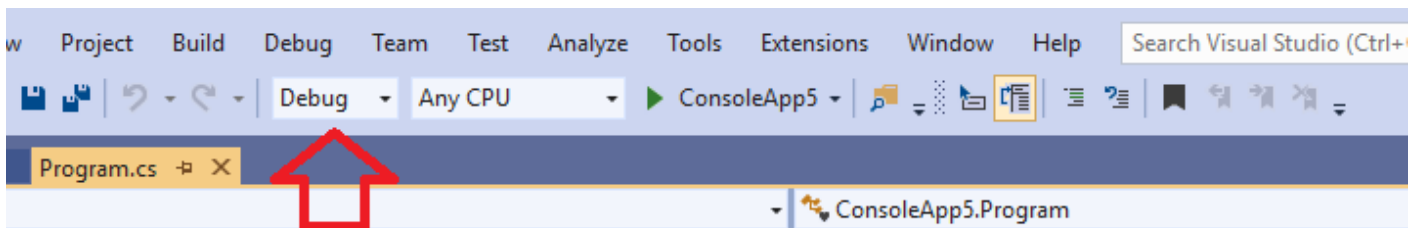
## Índice

- 1 - Qué es debuguear?
- 2 - Cómo debuguear el código
- 3 - Ventajas de Debuguear

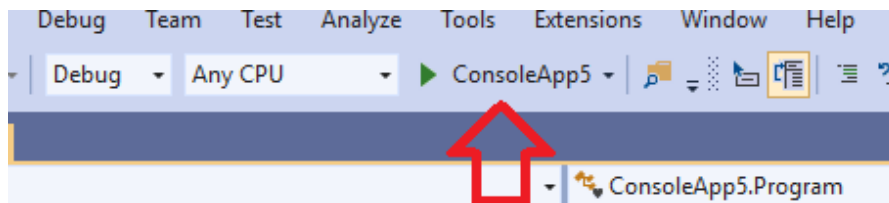
### 1 - Qué es debuguear?

Debuguear es lo que los desarrolladores hacemos para iterar sobre el código mientras este se esta ejecutando. Esto nos permite poder acceder a los bugs y analizarlos, ya que sin ello seria mucho mas complicado encontrar y arreglar errores.

Además, indicar que Visual Studio ( o nuestro IDE) tiene que estar en el Modo Debug, en VS la opción está arriba en el menú



Para entrar a iterar el codigo lo que tenemos que hacer es pulsar el "play" que tambien vemos en la imagen previa



eApp5

Una vez estamos dentro vamos a iterar sobre el código.

```
int contador = 0;
do
{
    Console.WriteLine("Iteración número " + contador);
    contador++;
} while (contador < 10);
```

### 2 - Cómo debuguear el código

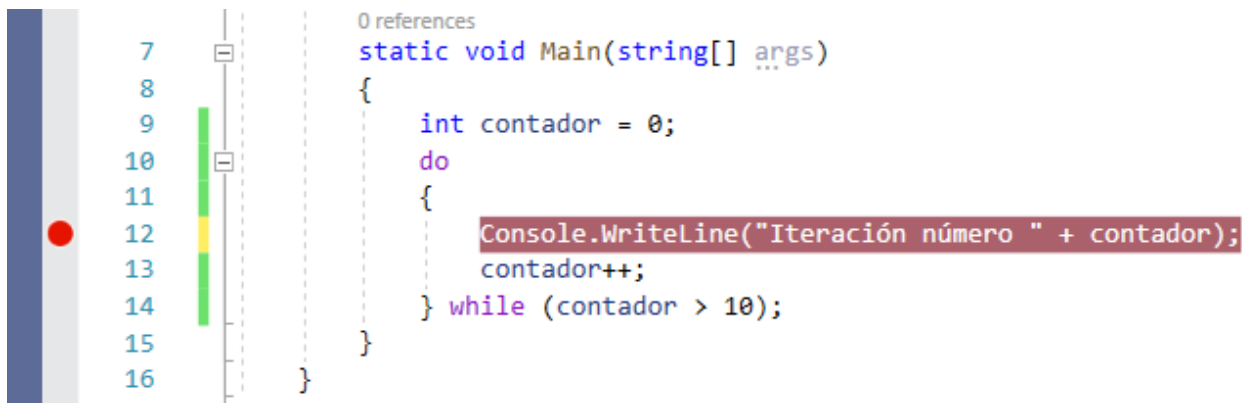
Pero para poder debuguear el código como tal necesitamos añadir `breakpoints` , o puntos de ruptura en castellano, y para añadirlos tenemos 3 opciones.

Colocar el cursor sobre la línea y pulsar **F9**

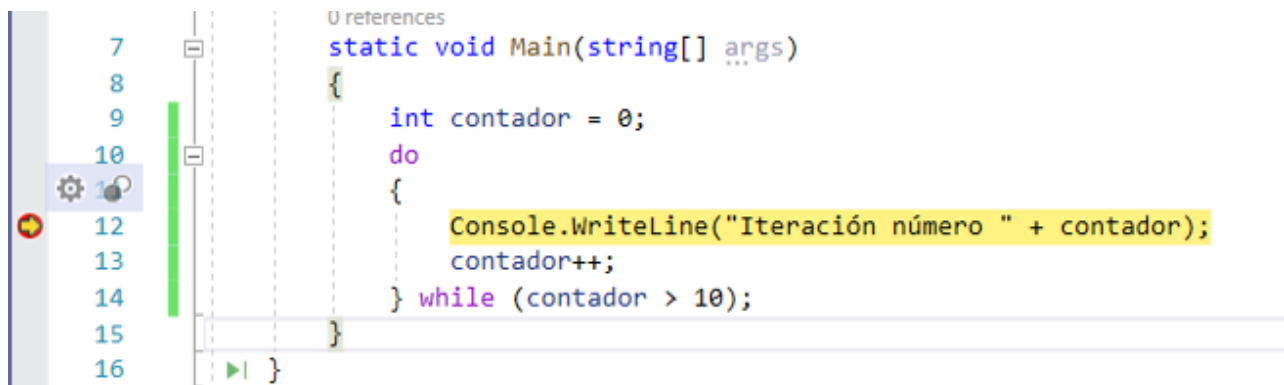
Boton secundario sobre la línea > Breakpoint > insertar breakpoint

En el lateral del fichero pulsar con el botón izquierdo del ratón, y nos añadira un breakpoint

Cuando hemos añadido un breakpoint nos saldrá un punto rojo, el cual indica que es un breakpoint (es el mismo punto que pinchamos utilizamos la opción 3) además la línea se nos pondrá de color rojo.



Una vez el programa está en ejecución, parará automáticamente al llegar al breakpoint, y este se nos pondrá amarillo, junto con una flecha en el lateral del fichero, para indicar que es ahí el punto en el que está en este momento.



Una vez estamos aquí tenemos varias opciones para continuar:

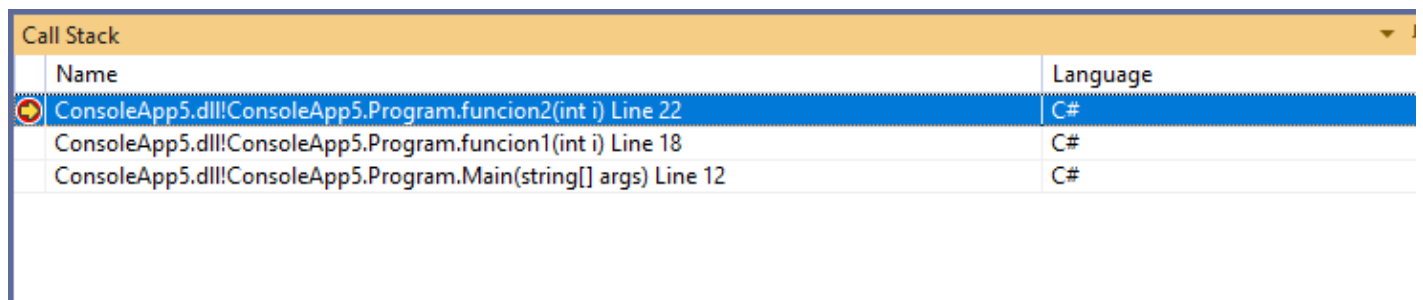
- **F5** continua hasta el siguiente breakpoint
- **F10** Ejecuta esa línea de código
- **F11** Ejecuta la siguiente sentencia de código, esto quiere decir que si por ejemplo estamos llamando a un método dentro de esa línea de código, pulsando F11 entraria dentro del método.

Otra opción que tenemos es arrastrar la flecha hasta la línea que queramos, o volver atrás, pero hay que tener cuidado ya que puede causar algunas excepciones, si algun elemento no ha sido declarado, por ejemplo.

### 3 - Ventajas de Debuguear

Como hemos dicho antes, podemos ver paso por paso la ejecución de un programa, y eso incluye que cuando se rompe o tenemos información corrupta podemos ver donde y porque están esos datos corruptos.

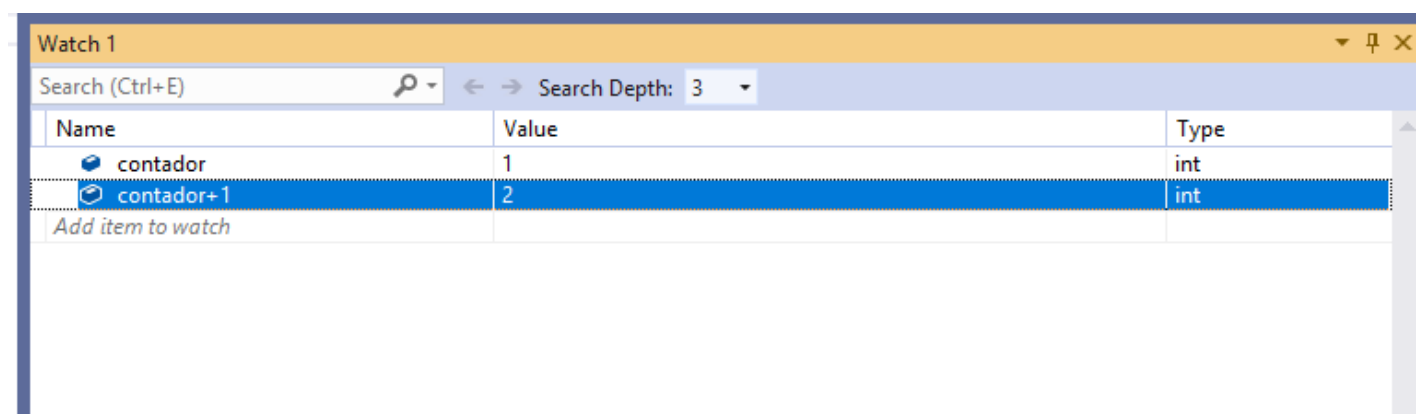
Disponemos de acceso al Stack de la llamada, que incluye cada punto por el que el proceso ha pasado:



Name	Language
ConsoleApp5.dll!ConsoleApp5.Program.funcion2(int i) Line 22	C#
ConsoleApp5.dll!ConsoleApp5.Program.funcion1(int i) Line 18	C#
ConsoleApp5.dll!ConsoleApp5.Program.Main(string[] args) Line 12	C#

Si hacemos doble click en cualquiera de ellos, Visual studio nos llevar a ese punto, pero no solo eso, las variables tendrán el valor que tenían en ese punto. Lo cual facilita mucho a la hora de comprobar errores.

Como hemos dicho cuando estamos debugueando tenemos acceso a todas las variables disponibles en el scope, por lo que podemos consultar su valor o incluso operar con ellas en la caja "watch" de visual stuido, lo cual facilita mucho cuando tenemos grandes bloques de informaión como listas de empleados etc.



Name	Value	Type
contador	1	int
contador+1	2	int

Add item to watch

Como vemos la segunda línea es nuestro contador, que hemos operado con el, esta informacion es temporal y no es la que se va a almacenar en el código cuando volvamos a ejecutarlo.

Finalmente indicar que podemos cambiar valores de las variables cuando estamos parados en un breakpoint, eso nos puede facilitar en ciertos casos a comprobar un "y qué pasaría si..." pero personalmente no lo recomiendo.