PRACTICA 2. ENTORNO DE DESARROLLO VISUAL STUDIO. PARTE II.

INTRODUCCIÓN TEÓRICA

Esta práctica tiene como objeto aprender a depurar código (debugear) como un profesional: Debugging en Visual Studio

Las tres actividades en la que más tiempo estarás ocupado como desarrollador de software son: Haciendo un research, es decir buscando información online, depurando código, ya sea porque quieres encontrar un bichito, o bug, o porque estás intentando entender alguna porción de código y la tercera es programando.

Dicho esto, entonces es muy importante aprender a depurar código como debe ser, sin embargo hay que tener algo en claro:

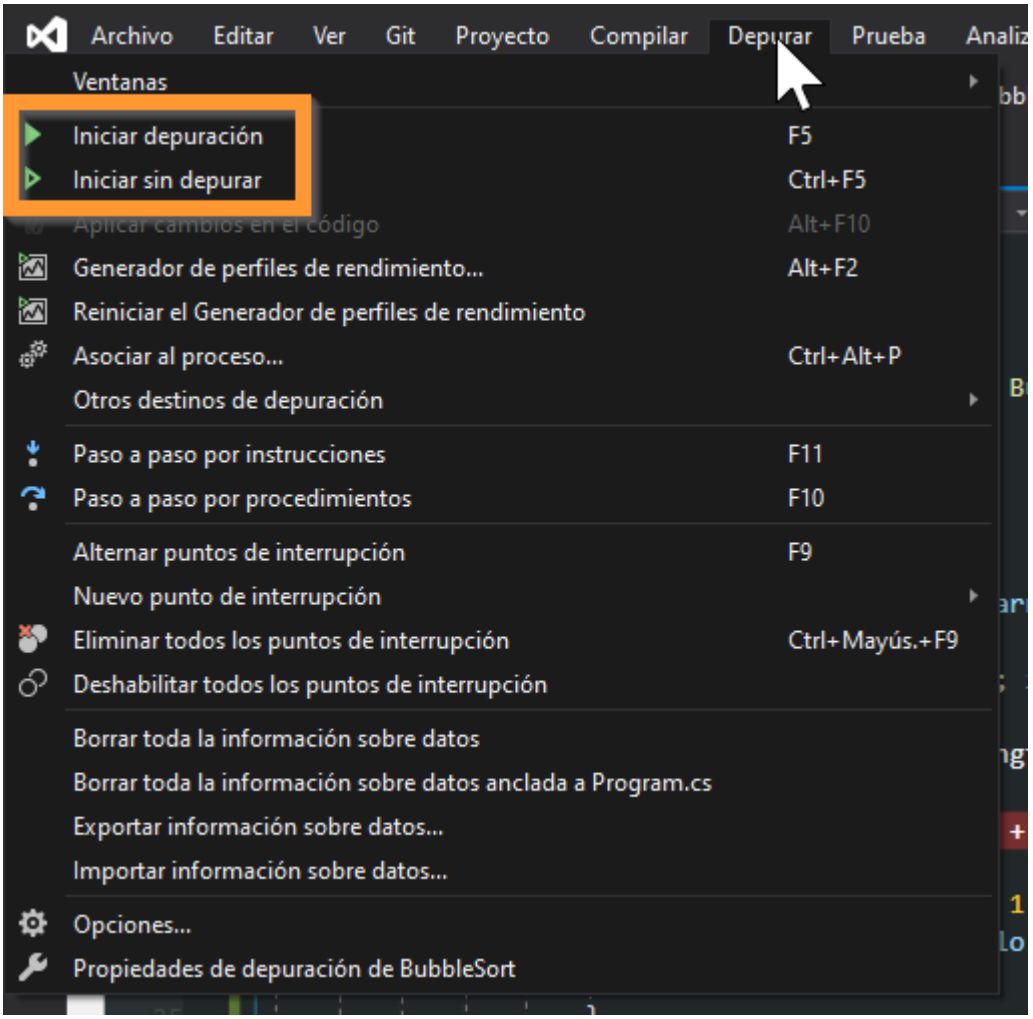
La actividad de depurar que hacemos los desarrolladores de software va más allá de utilizar una herramienta de depuración en un IDE, sino que abarca toda una serie de estrategias y técnicas que incluyen: pensar en cómo es el ciclo de ejecución de un programa, entender el porqué de las cosas, plantear hipótesis, recrear escenarios, realizar pruebas y valerse de herramientas de depuración.

Modos de inicio de un programa

Depurar significa *ejecutar el código paso a paso* en una herramienta de depuración como Visual Studio, para buscar el punto exacto donde ha cometido un error de programación.

Comencemos por definir las bases, en Visual Studio hay 2 modos de iniciar cualquier programa:

- Iniciar en modo depuración (pulsando *F5*)
- Inicio sin depuración, o inicio normal (pulsando *Ctrl + F5*)



Casualmente el modo más común de iniciar un programa en Visual Studio es con F5 aunque esto requiere más recursos y se toma más tiempo para ejecutar el programa, así que se puede ejecutar un programa directamente y de forma más rápida con *CTRL + F5*.

Los puntos de Interrupción:

Son aquellos puntos donde estaremos diciéndole al IDE, cuando mi programa pase por este punto detente, quiero analizar algo.

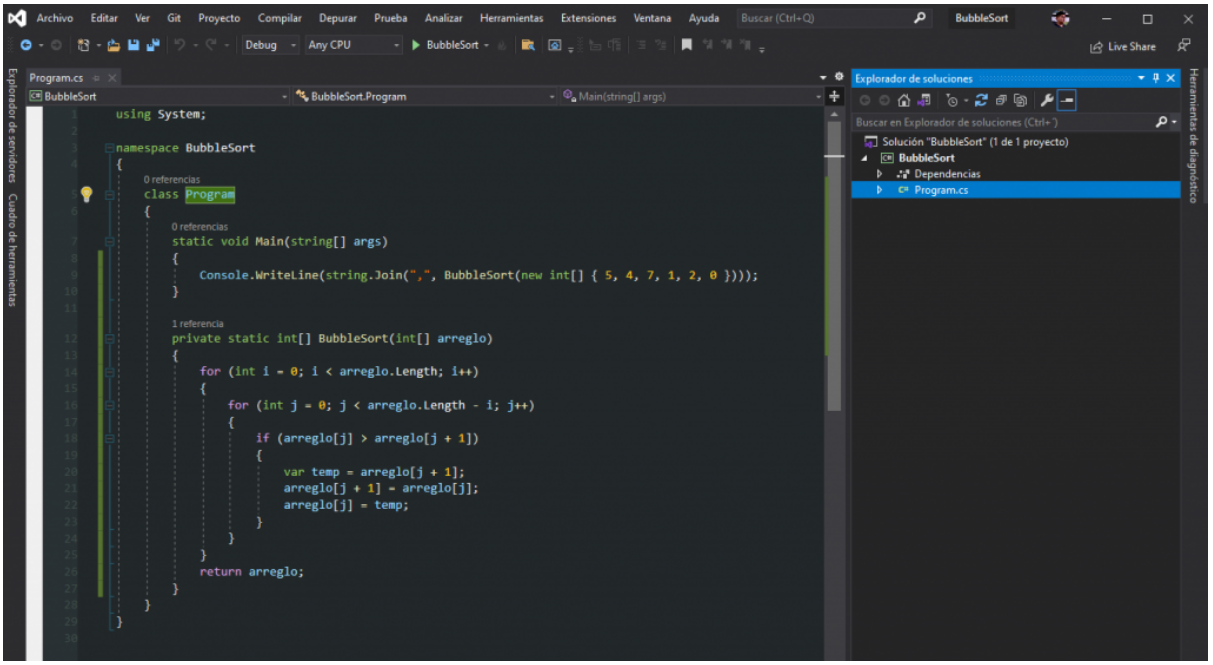
Depurando con una aplicación real

Todo se aprende mejor con ejemplos reales, para hacer unas pruebas voy a depurar un algoritmo de ordenamiento llamado *BubbleSort*, que me está dando problemas, ayúdame a resolver esta cuestión y encontrar dónde está el bug.

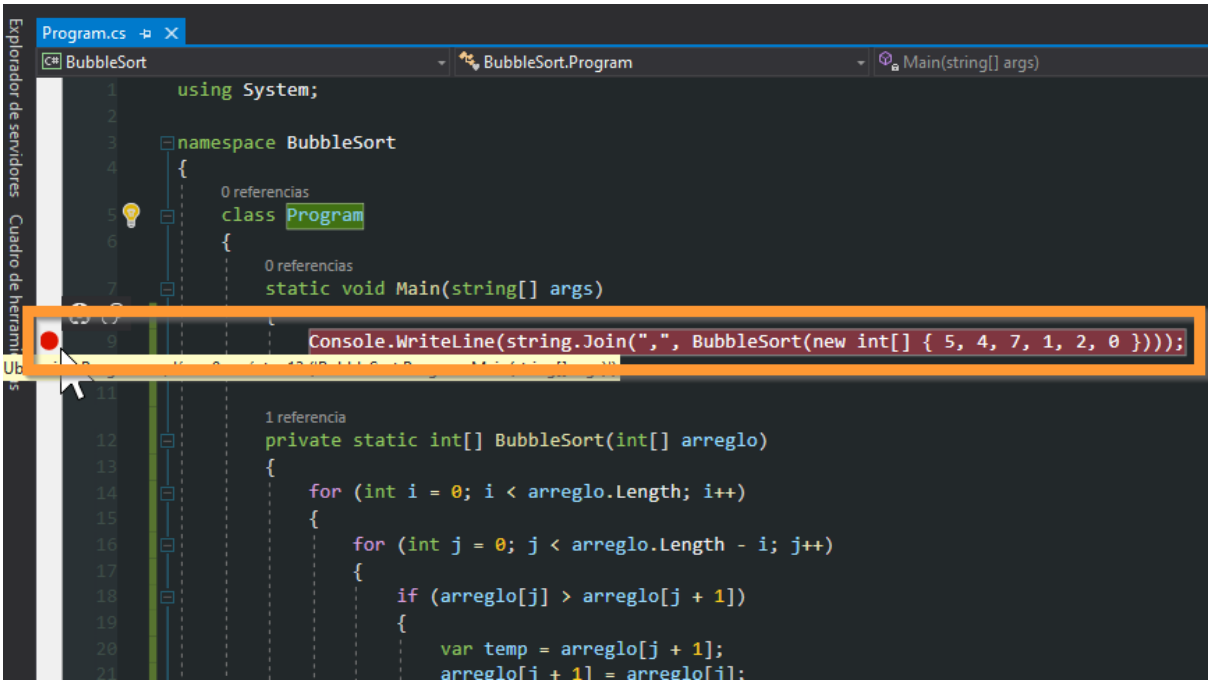
He creado una aplicación de consola y he creado un método llamado *BubbleSort* directamente en la clase *Program*, aquí el código:

```
class Program
{
    static void Main(string[] args)
    {
            Console.WriteLine(string.Join(",", BubbleSort(new
int[] { 5, 4, 7, 1, 2, 0 })));
    }

    private static int[] BubbleSort(int[] arreglo)
    {
        for (int i = 0; i < arreglo.Length; i++)
        {
            for (int j = 0; j < arreglo.Length - i; j++)
            {
                if (arreglo[j] > arreglo[j + 1])
                {
                    var temp = arreglo[j + 1];
                    arreglo[j + 1] = arreglo[j];
                    arreglo[j] = temp;
                }
            }
        }
        return arreglo;
    }
}
```

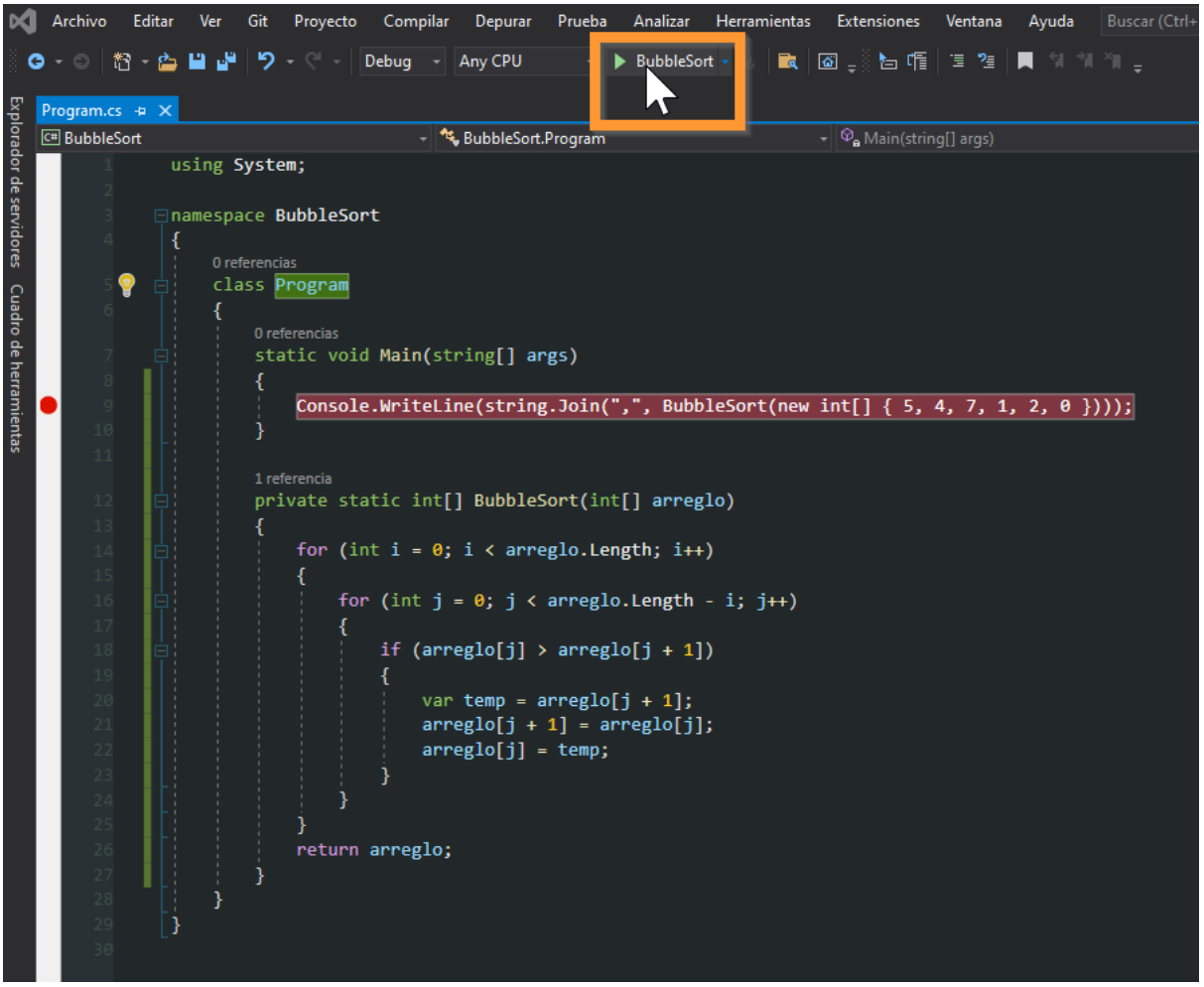
Mi programa tiene esta pinta:



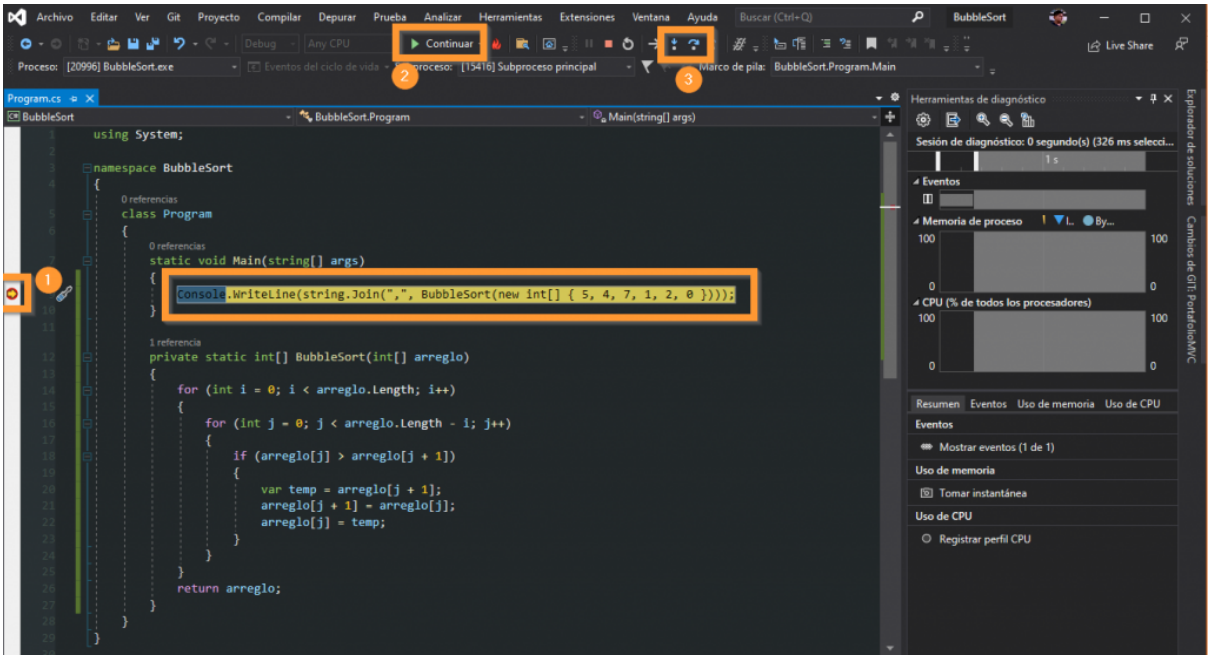
Para crear estos puntos debemos dar clic en la barra lateral de la izquierda justo en la línea donde se desea establecer la interrupción:



Ahora pulsamos *F5* o damos clic en el botón que se muestra a continuación:



Una vez iniciada la depuración se mostrarán algunos cambios en la pantalla, veamos los 3 principales:



1: El IDE nos muestra el punto de interrupción que nosotros hemos definido, la flechita amarilla nos indica que esa línea resaltada de amarillo es la próxima a ejecutarse

2: Ahora el botón no dice Iniciar sino continuar, esto lo pulsaremos sí queremos dejar la depuración y que el programa siga su curso normal

3: Estos botones nos permiten seguir depurando, y hay dos formas de seguir depurando: Paso a paso por procedimientos o *Step Over (F10)* y paso a paso por instrucciones o *Step into (F11)* estos los veremos a continuación

El poder del Step into/Step Over

Sigamos en el ejemplo, el programa se detuvo justo a punto de ejecutarse la línea que le indicamos, genial, ahora como veremos yo no quiero ejecutar la siguiente línea, que en mi código sería la línea de código #10 sino que quiero ingresar al método BubbleSort.

Eso lo conseguimos con Step into pulsando la tecla *F11*, así que hagámoslo

Se nos muestra la siguiente pantalla:

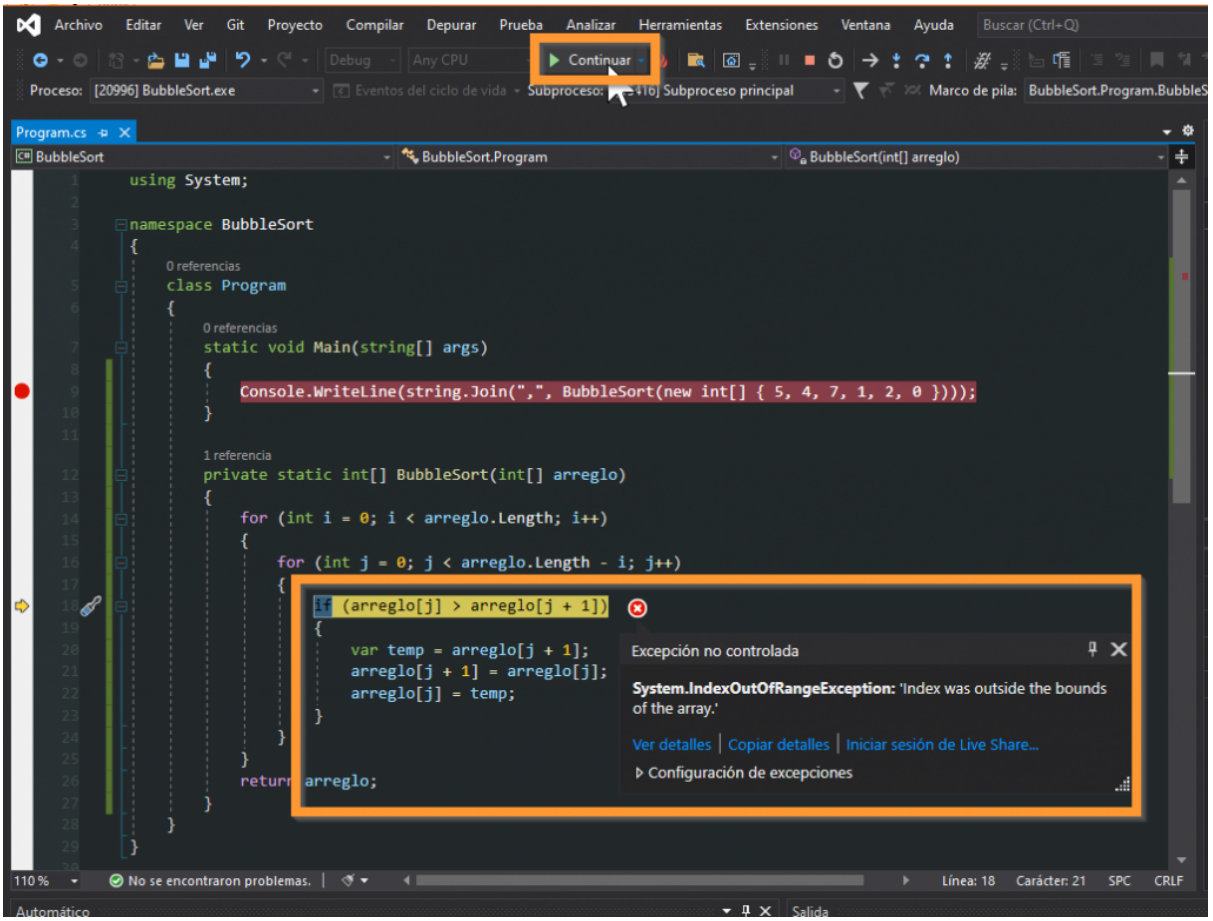
```
class Program
{
    0 referencias
    static void Main(string[] args)
    {
        Console.WriteLine(string.Join(",", BubbleSort(new int[] { 5, 4, 7, 1, 2, 0 })));
    }

    1 referencia
    private static int[] BubbleSort(int[] arreglo)
    {                ≤ 11 ms transcurridos
        for (int i = 0; i < arreglo.Length; i++)
        {
            for (int j = 0; j < arreglo.Length - i; j++)
            {
                if (arreglo[j] > arreglo[j + 1])
                {
                    var temp = arreglo[j + 1];
                    arreglo[j + 1] = arreglo[j];
                    arreglo[j] = temp;
                }
            }
        }
        return arreglo;
    }
}
}
```

Ten en cuenta lo siguiente: Si quieres pasar a la siguiente línea simplemente pulsa *F10*, si quieres ingresar y depurar algún método usado en la línea actual pulsas *F11* y el depurador te llevará hasta donde se encuentre declarado ese método.

Muy bien, la flecha amarilla nos indica la siguiente línea que se va a ejecutar, en otras palabras ingresamos a depurar el método BubbleSort, y todo porque pulsamos F11, de ahora entonces tendremos que ir línea por línea con F10, a menos que queramos ingresar a algún método, en tal caso volveríamos a ingresar en él mediante F11 nuevamente.

Si pulsamos el botón *CONTINUAR* de la parte superior, veamos lo que ocurre:

Esto hace que el programa continúe y se detiene abruptamente debido a un bug, el IDE nos dice exactamente en qué línea ocurrió la excepción y nos da un mensaje de referencia del error, en este caso al parecer estamos accediendo a un arreglo usando un índice no válido.



Entonces para la siguiente depuración ya sabemos dónde ubicar el punto de interrupción, será en la línea #18 de mi programa. Bien, sigamos, por ahora hay que detener el programa con el botón *STOP* de la parte superior y quitar el punto de interrupción actual (el de la línea #9).

Condicionales en las interrupciones

Ponemos el punto de interrupción en la línea #18 pero eso significa que vamos a tener que estar iterando cuantas veces sea necesario hasta desbordar el array, lo cual tomaría tiempo, analicemos.

Y es justo aquí donde entra la tarea de depuración de pensar, y pensar mucho.

Analicemos que hay dos bucles anidados:

Conforme el bucle padre avanza va acortando el límite del hijo ya que el valor máximo de la variable j que es la que itera al hijo se le sustrae el valor de i. Entonces el valor máximo de j es cuando i es 0

Entonces el valor máximo de j será definida en esta porción de código:

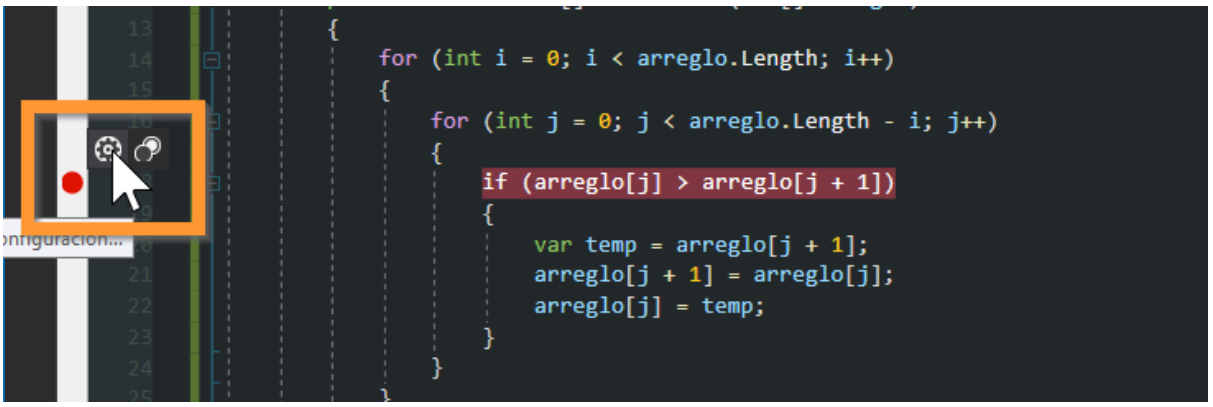
`j < arreglo.Length - i`

esto significa:

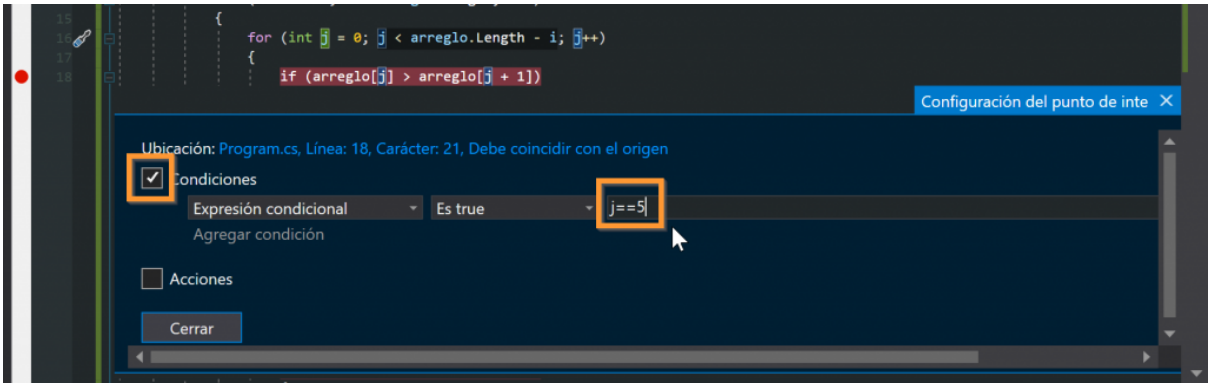
`j < 6 - 0`

Entonces el valor máximo de j es: **j = 5**.

Por lo tanto nos interesa capturar el momento exacto cuando j vale 5 en esa línea de interrupción, aquí es donde entra a tallar las condicionales en los puntos de interrupción, y ponemos condicionales así:

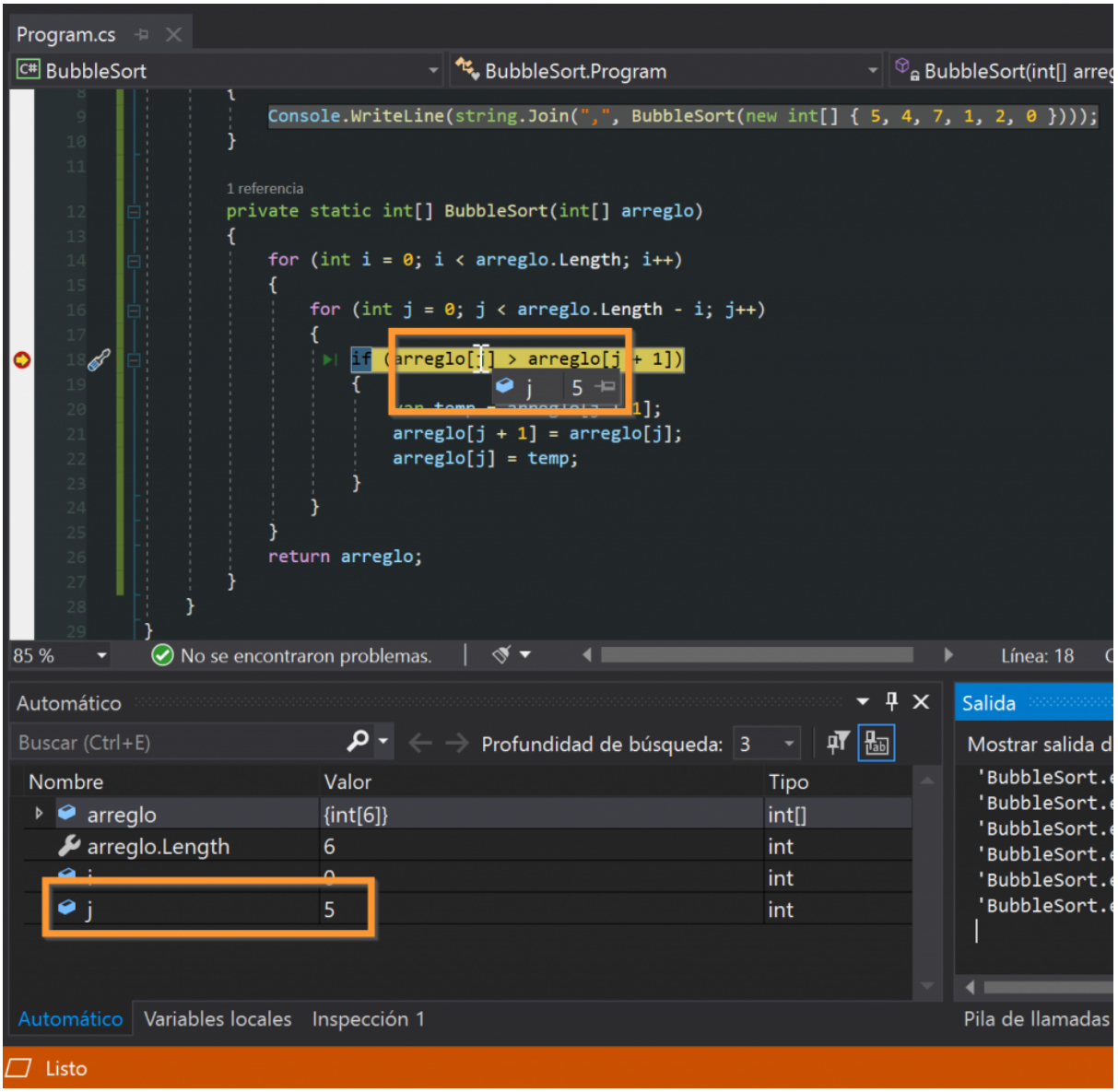


Pulsamos en la tuerquita y luego clic en *Condicionales* y escribir la condicional deseada, veamos:



Cierra la ventanita y ahora pulsa F5

Ahora la depuración se interrumpió no sólo en la línea indicada sino cuando se cumple la condicional, y podemos comprobar el valor actual de **j** posamos el cursor sobre la variable y como vemos en la imagen se muestra el valor actual, también podemos ver los valores actuales en la ventana de la parte inferior, para poder comprobar los valores de cualquier variable en el punto de interrupción actual:



Esto de posar el cursor sobre cualquier variable es válida para variables de cualquier tipo, sean listas, arreglos, tipos primitivos como enteros, strings, etc

Estamos entonces en el punto exacto donde se podría estar dando el problema.

Si nos posamos sobre el índice *[j+1]* podremos notar que estamos intentando acceder al índice 6 en un array de longitud 6 y eso no es posible ya que los índices empiezan en 0, así que el índice máximo es 5, seguro ese es el problema, eureka!

Pulsamos *F10* para seguir con la depuración de todas maneras:

Tal como lo sospechamos, esa era la causa del problema, así que ya sabemos que tenemos que hacer, veamos:

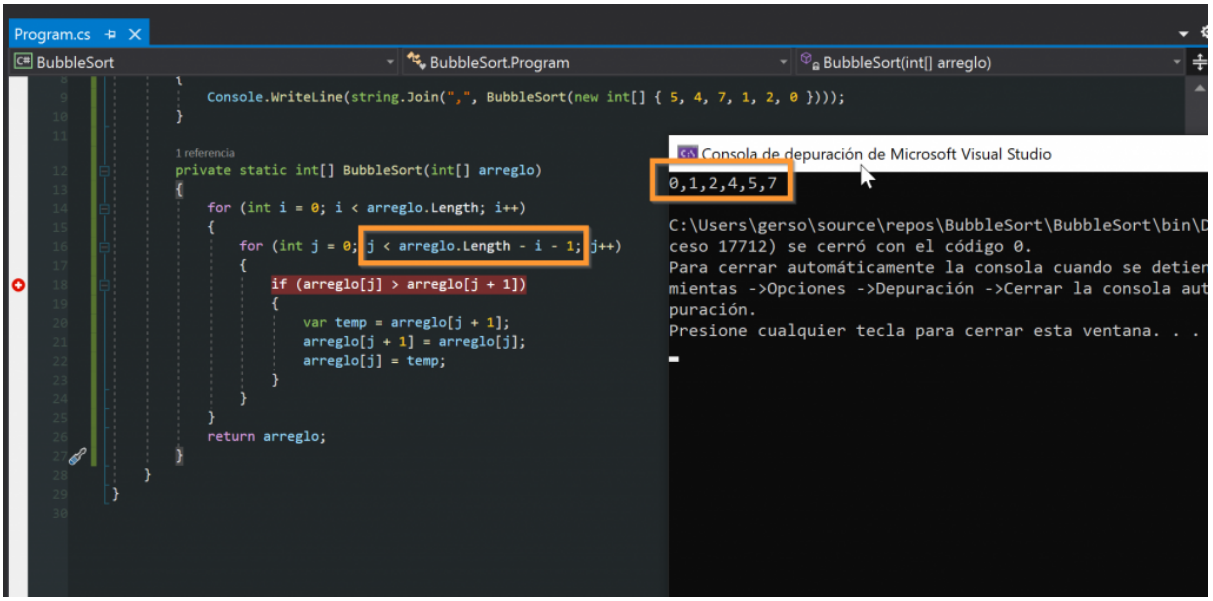
Tenemos que evitar que j llegue a 5 ya que estamos haciendo una comparación con un índice j+1, así que hagamos que el límite máximo sea en vez de :

`j < arreglo.Length - i`

sea este:

`j < arreglo.Length - i - 1`

Ahora probemos de nuevo! Pulsar F5



Genial! ahora sí, y ni siquiera se detuvo en el punto de interrupción, ¿por qué?

Fácil, porque j nunca llega hasta 5 ahora sino hasta 4

Entonces ahora sí, mi algoritmo de ordenamiento BubbleSort funciona!

Desarrollo de habilidades

1. Investigar qué hacen y cómo funciona
   - Ventanas automático y variables locales.
   - Ventana inspección
   - Pila de llamada.

Haz un ejemplo real donde hagas uso de cada una de ellas.