

Inicie programando en este lenguaje de programación bastante popular. Puede usarlo tanto en Windows como en Linux. Y tiene el apoyo de Microsoft.

# Iniciando en C#

Variables, si condicional, ciclos y funciones

Rafael Alberto Moreno Parra. Agosto de 2020

---

Contenido

Otros libros del autor ..... 2

Página web del autor y canal en Youtube ..... 2

Sitio en GitHub ..... 2

Licencia del software ..... 3

Marcas registradas ..... 3

Introducción..... 4

Trabajando con Microsoft Visual Studio Community 2019..... 5

Trabajando en Linux con MonoDevelop ..... 6

Iniciando proyectos de consola con Microsoft Visual Studio Community 2019 ..... 7

Directorio o carpeta por defecto..... 11

La diferencia entre Debug y Release..... 12

Comentarios en el código ..... 14

El tradicional “Hola Mundo” ..... 15

Variables ..... 16

Imprimiendo con formato ..... 19

Operaciones matemáticas..... 22

Diferencias de precisión entre float y double ..... 23

Función de potencia..... 24

Orden de evaluación de los operadores..... 25

Problemas al usar cast ..... 26

De formato algebraico a C# ..... 28

Diferencias entre usar el cast y la conversión..... 30

Conversión de números reales con el punto decimal ..... 31

Fallos en la conversión ..... 32

Constantes matemáticas ..... 33

Funciones trigonométricas ..... 34

Funciones hiperbólicas ..... 35

Otras funciones matemáticas ..... 36

Manejo del NaN (Not a Number) y el infinito ..... 38

Leer un número por consola..... 39

Si condicional ..... 40

Uso del if...else if... else ..... 41

Uso del AND && ..... 42

Uso del OR || ..... 43

Uso del switch ..... 44

Operadores booleanos ..... 45

Precedencia de los operadores ..... 49

El operador “?”, un si condicional..... 50

Captura de error con try...catch ..... 51

Uso de TryParse para conversión ..... 52

Ciclo for..... 53

Ciclo while ..... 55

Ciclo do...while..... 57

Romper ciclo con break ..... 59

Uso del continue en ciclos..... 60

Ciclos anidados ..... 61

Rompiendo ciclos anidados ..... 62

    Uso del goto ..... 63

Diferencias de precisión entre float y double ..... 64

Generando números aleatorios ..... 65

Funciones ..... 66

Funciones con doble salida ..... 67

Funciones iterativas y recursivas ..... 68

Ámbito de las variables ..... 69

Azar y ámbito de la variable aleatoria ..... 70

## Otros libros del autor

Libro 13: "Algoritmos Genéticos". En Colombia 2020. Págs. 62. Libro y código fuente descargable en: <https://github.com/ramsoftware/LibroAlgoritmoGenetico2020>

Libro 12: "Redes Neuronales. Segunda Edición". En Colombia 2020. Págs. 108. Libro y código fuente descargable en: <https://github.com/ramsoftware/LibroRedNeuronal2020>

Libro 11: "Capacitándose en JavaScript". En Colombia 2020. Págs. 317. Libro y código fuente descargable en: <https://github.com/ramsoftware/JavaScript>

Libro 10: "Desarrollo de aplicaciones para Android usando MIT App Inventor 2". En Colombia 2016. Págs. 102. Ubicado en: <https://openlibra.com/es/book/desarrollo-de-aplicaciones-para-android-usando-mit-app-inventor-2>

Libro 9: "Redes Neuronales. Parte 1.". En Colombia 2016. Págs. 90. Libro descargable en: <https://openlibra.com/es/book/redes-neuronales-parte-1>

Libro 8: "Segunda parte de uso de algoritmos genéticos para la búsqueda de patrones". En Colombia 2015. Págs. 303. En publicación por la Universidad Libre – Cali.

Libro 7: "Desarrollo de un evaluador de expresiones algebraicas. **Versión 2.0.** C++, C#, Visual Basic .NET, Java, PHP, JavaScript y Object Pascal (Delphi)". En: Colombia 2013. Págs. 308. Ubicado en: <https://openlibra.com/es/book/evaluador-de-expresiones-algebraicas-ii>

Libro 6: "Un uso de algoritmos genéticos para la búsqueda de patrones". En Colombia 2013. En publicación por la Universidad Libre – Cali.

Libro 5: Desarrollo fácil y paso a paso de aplicaciones para Android usando MIT App Inventor. En Colombia 2013. Págs. 104. Estado: Obsoleto (No hay enlace).

Libro 4: "Desarrollo de un evaluador de expresiones algebraicas. C++, C#, Visual Basic .NET, Java, PHP, JavaScript y Object Pascal (Delphi)". En Colombia 2012. Págs. 308. Ubicado en: <https://openlibra.com/es/book/evaluador-de-expresiones-algebraicas>

Libro 3: "Simulación: Conceptos y Programación" En Colombia 2012. Págs. 81. Ubicado en: <https://openlibra.com/es/book/simulacion-conceptos-y-programacion>

Libro 2: "Desarrollo de videojuegos en 2D con Java y Microsoft XNA". En Colombia 2011. Págs. 260. Ubicado en: <https://openlibra.com/es/book/desarrollo-de-juegos-en-2d-usando-java-y-microsoft-xna> . ISBN: 978-958-8630-45-8

Libro 1: "Desarrollo de gráficos para PC, Web y dispositivos móviles" En Colombia 2009. ed.: Artes Gráficas Del Valle Editores Impresores Ltda. ISBN: 978-958-8308-95-1 v. 1 págs. 317

Artículo: "Programación Genética: La regresión simbólica".  
Entramado ISSN: 1900-3803 ed.: Universidad Libre Seccional Cali  
v.3 fasc.1 p.76 - 85, 2007

## Página web del autor y canal en Youtube

Investigación sobre Vida Artificial: <http://darwin.50webs.com>

Canal en Youtube: <http://www.youtube.com/user/RafaelMorenoP> (dedicado al desarrollo en C#)

## Sitio en GitHub

El código fuente se puede descargar en <https://github.com/ramsoftware/C-Sharp-Iniciando>

## Licencia del software

Todo el software desarrollado aquí tiene licencia LGPL “Lesser General Public License” [1]



## Marcas registradas

En este libro se hace uso de las siguientes tecnologías registradas:

Microsoft ® Windows ® Enlace: <http://windows.microsoft.com/en-US/windows/home>

Microsoft ® Visual Studio 2019 ® Enlace: <http://windows.microsoft.com/en-US/windows/home>

# Introducción

Este libro está dirigido al primer curso de programación que cubre los temas de instalación del entorno de desarrollo para programar en C#, tipos de variables, asignación, si condicional, ciclos y funciones.

El complemento de este libro es una serie de vídeos en Youtube donde se muestra la instalación de diversos entornos de desarrollo en Windows y en Linux.

El código fuente se puede descargar de GitHub en esta dirección: <https://github.com/ramsoftware/C-Sharp-Iniciando>

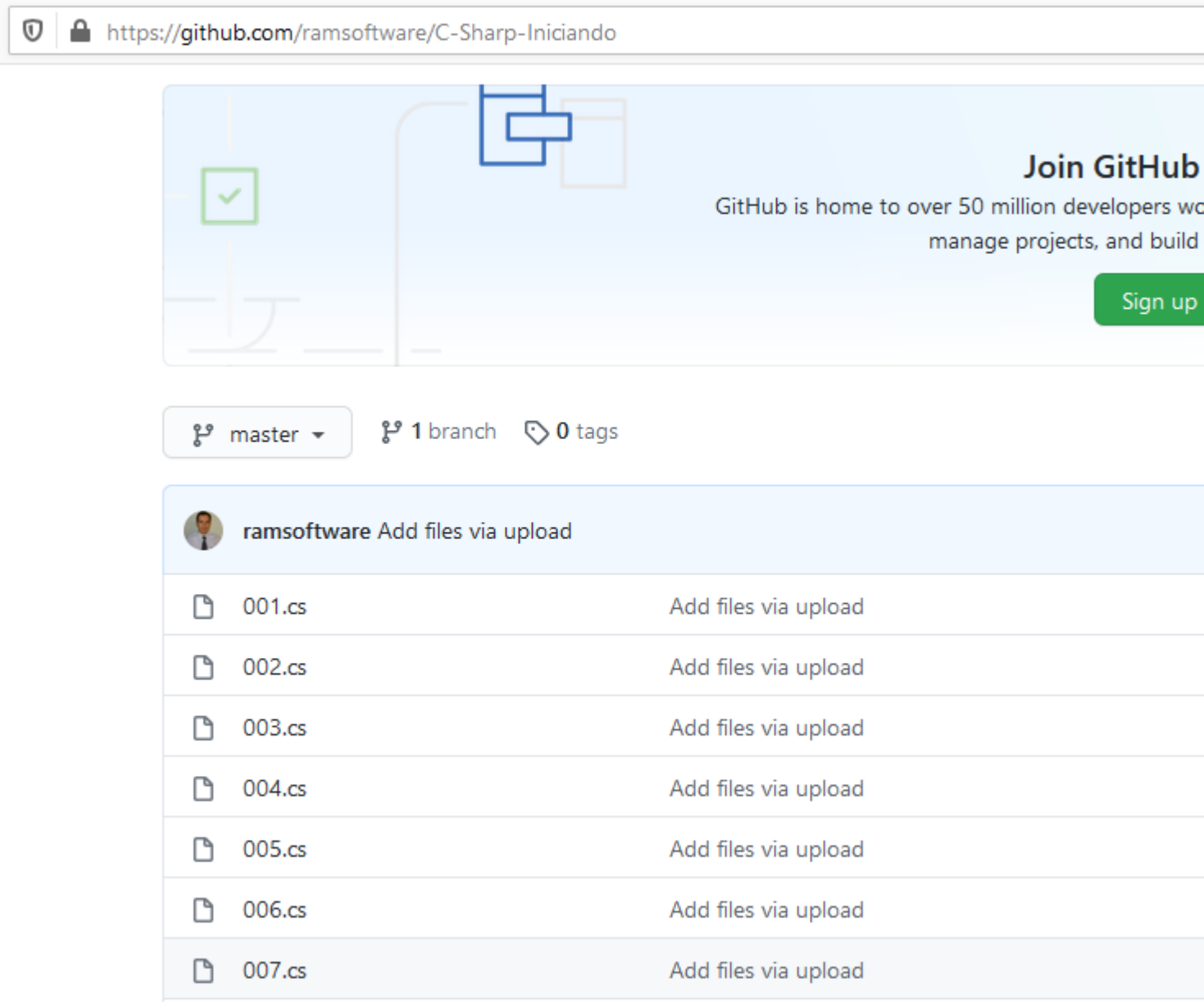


Ilustración 1: Contenido en GitHub

## Trabajando con Microsoft Visual Studio Community 2019

El entorno oficial para trabajar C# en Windows es Microsoft Visual Studio, la versión más reciente es la 2019 y para iniciar recomiendo la edición Community que es gratuita.

Instalando el entorno: <https://www.youtube.com/watch?v=e4-bfinmVo>

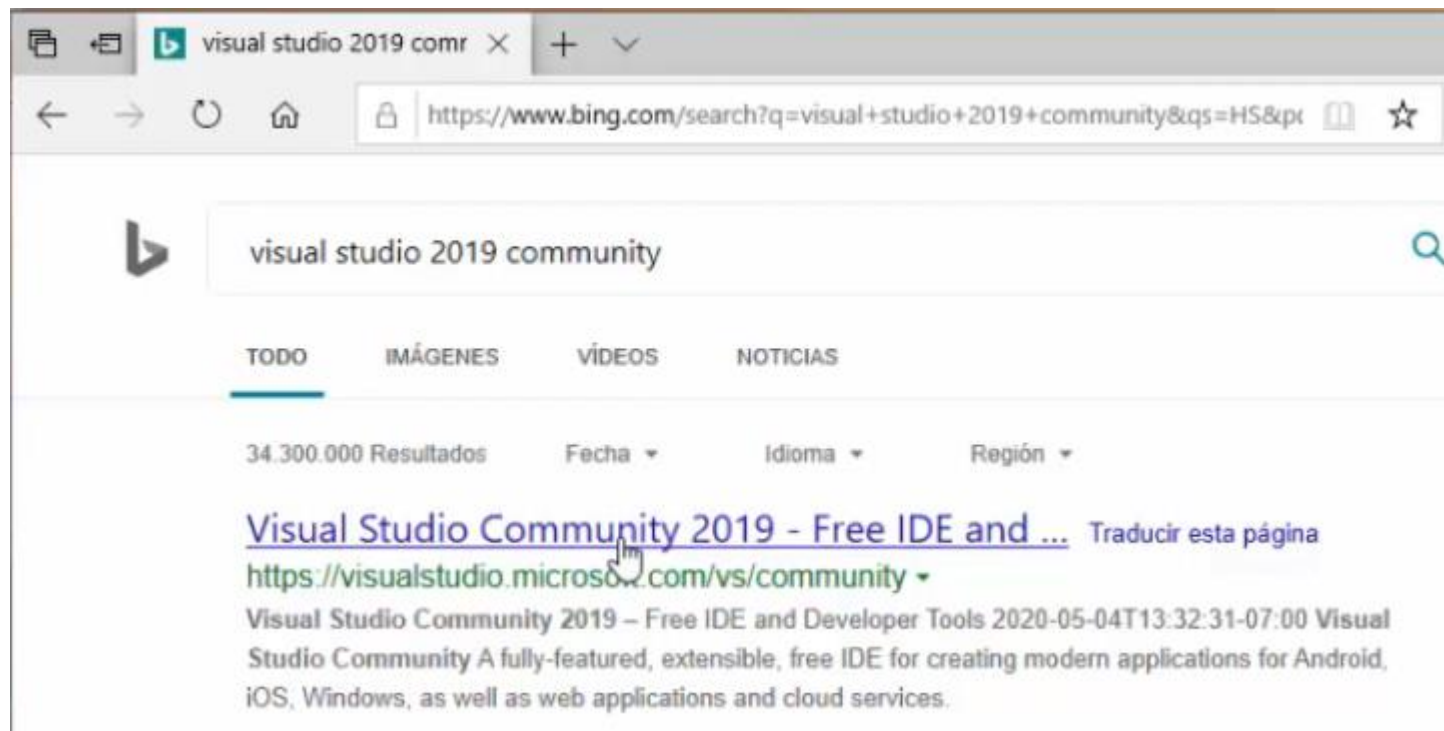


Ilustración 2: Descargando e instalando Visual Studio Community 2019

## Trabajando en Linux con MonoDevelop

Si su sistema operativo de trabajo es alguna distribución de Linux, entonces los siguientes enlaces muestran cómo es instalar el entorno de desarrollo MonoDevelop que es gratuito para programar en C#.

Instalando el entorno en MX Linux: <https://www.youtube.com/watch?v=jbpRj1bK4Qw>

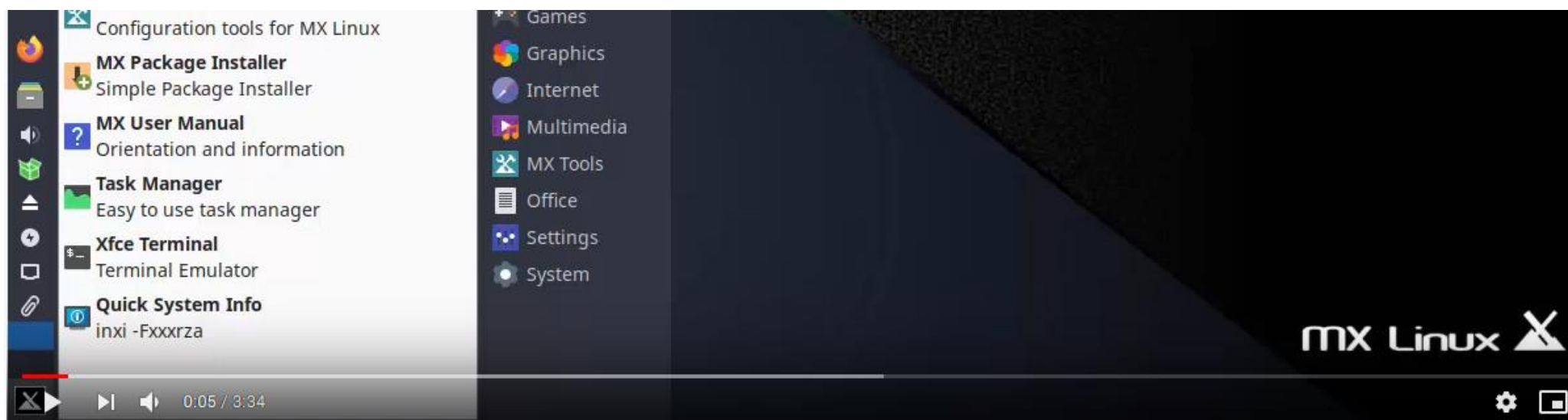


Ilustración 3: Descargando e instalando MonoDevelop en MX Linux

Instalando el entorno en Ubuntu: <https://www.youtube.com/watch?v=fpfstKLQLn0>

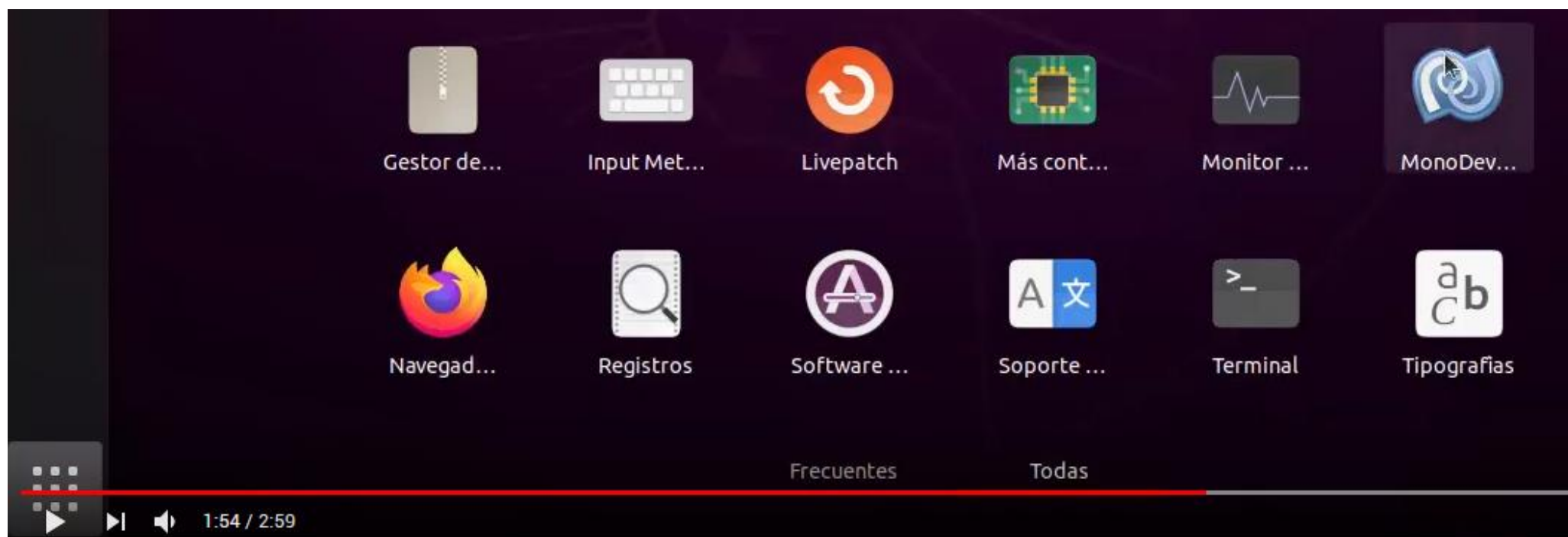


Ilustración 4: Instalando MonoDevelop en Ubuntu

Instalando el entorno en Linux Mint: [https://www.youtube.com/watch?v=vaRwAUq\\_3-w](https://www.youtube.com/watch?v=vaRwAUq_3-w)

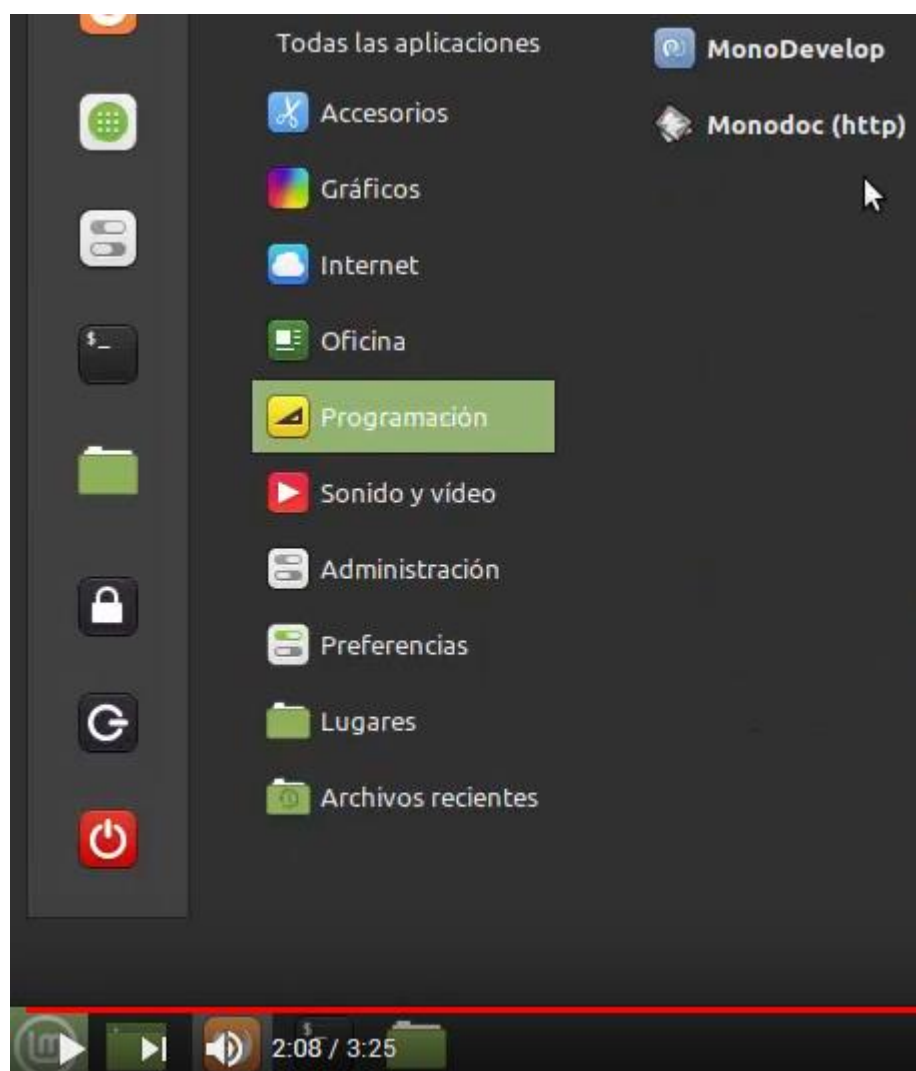
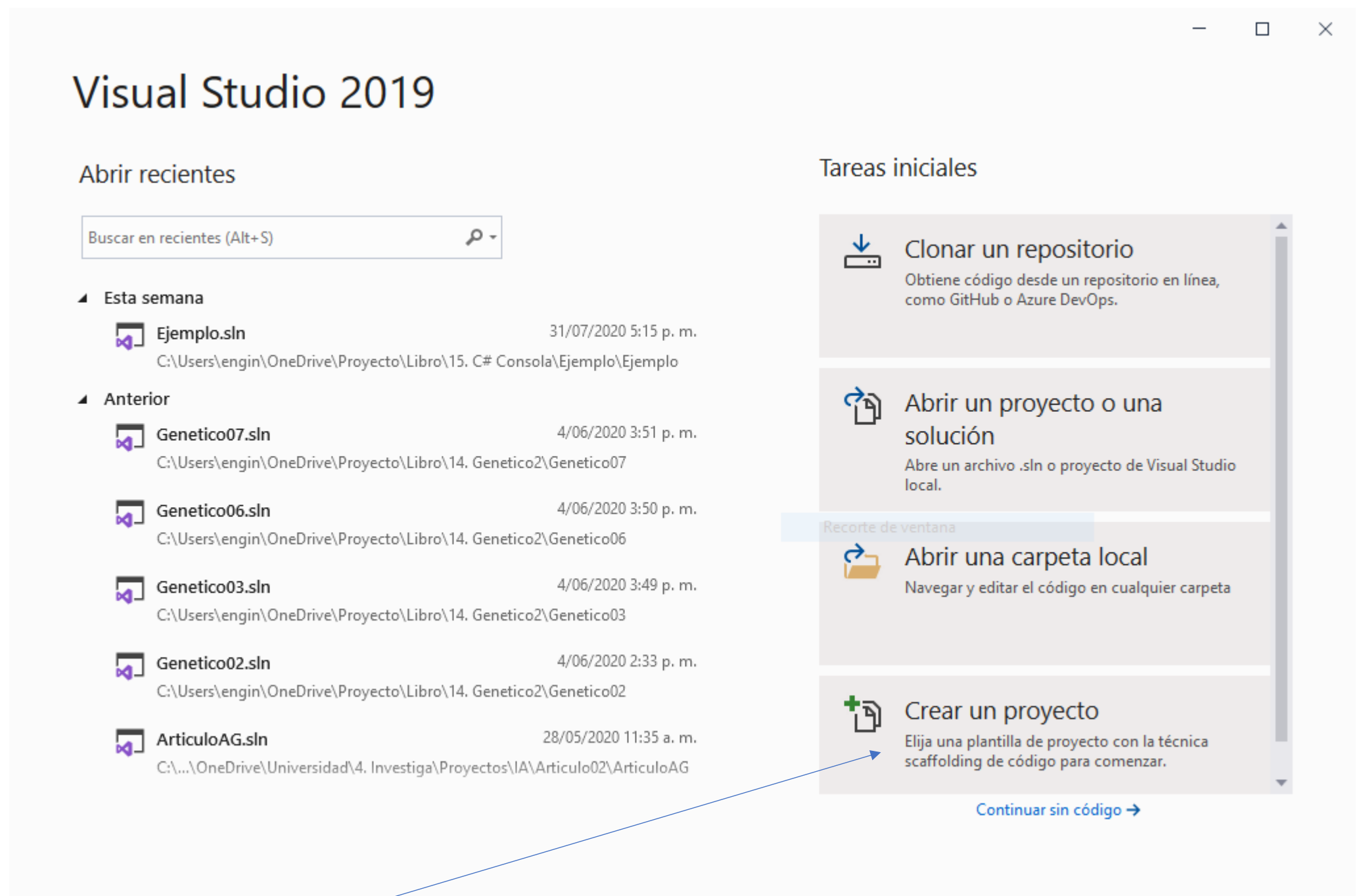


Ilustración 5: Instalando MonoDevelop en Linux Mint



# Iniciando proyectos de consola con Microsoft Visual Studio Community 2019

Al iniciar este entorno de desarrollo tenemos la pantalla siguiente:



Seleccionamos "Crear un proyecto"



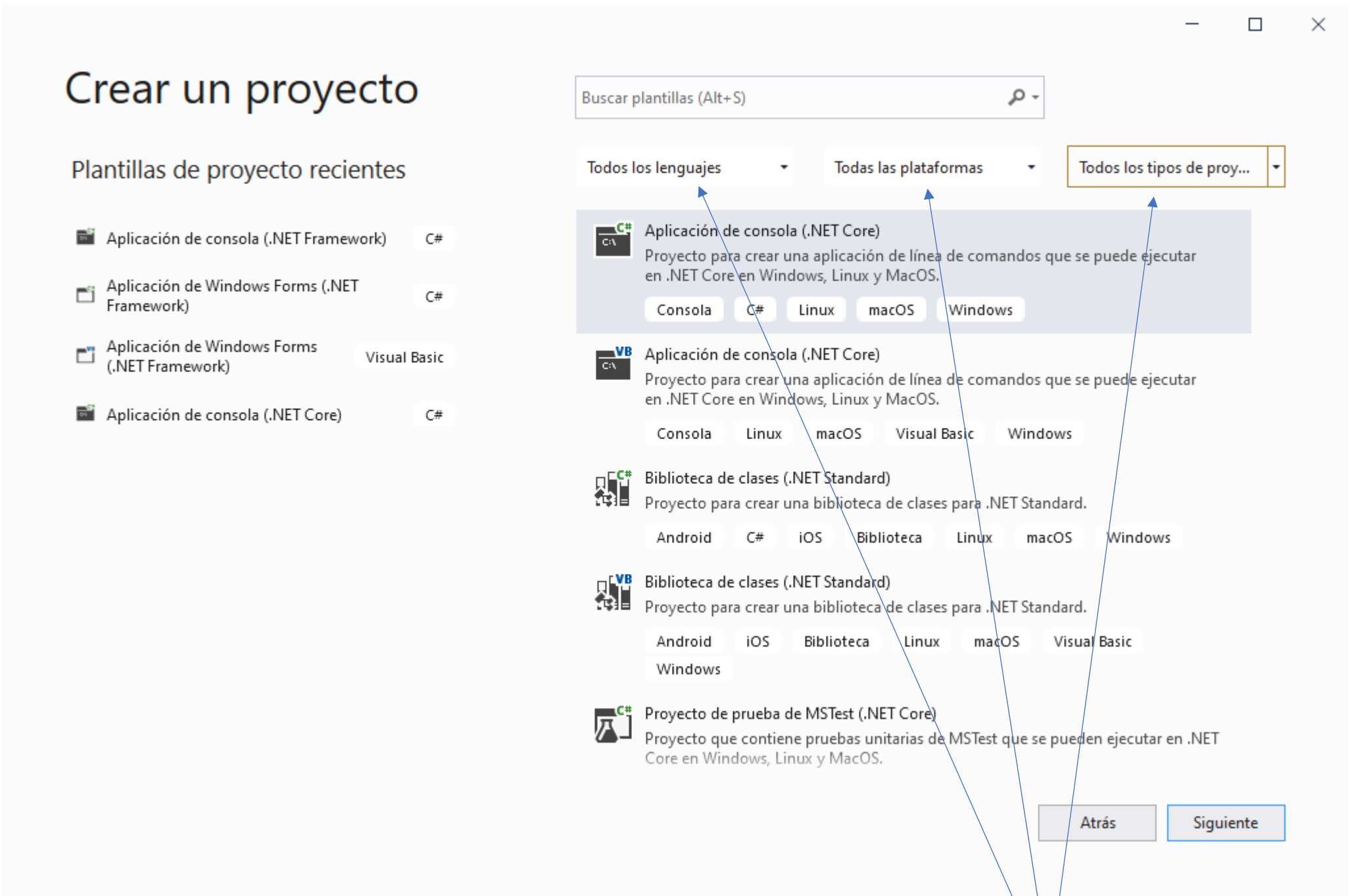


Ilustración 6: Aparecen todos los tipos de proyectos que se pueden hacer con este entorno. Se debe entonces filtrar con estas opciones

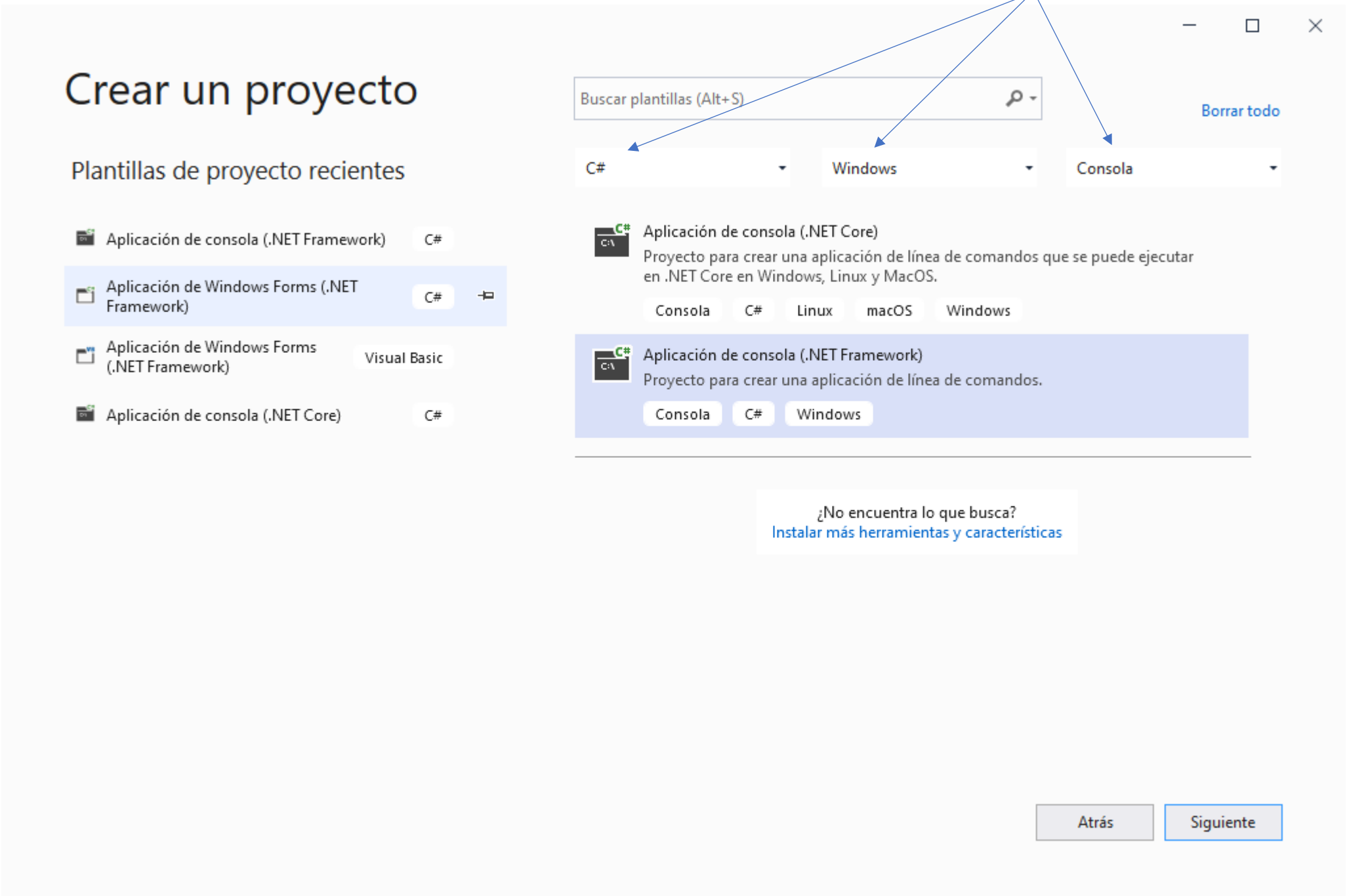
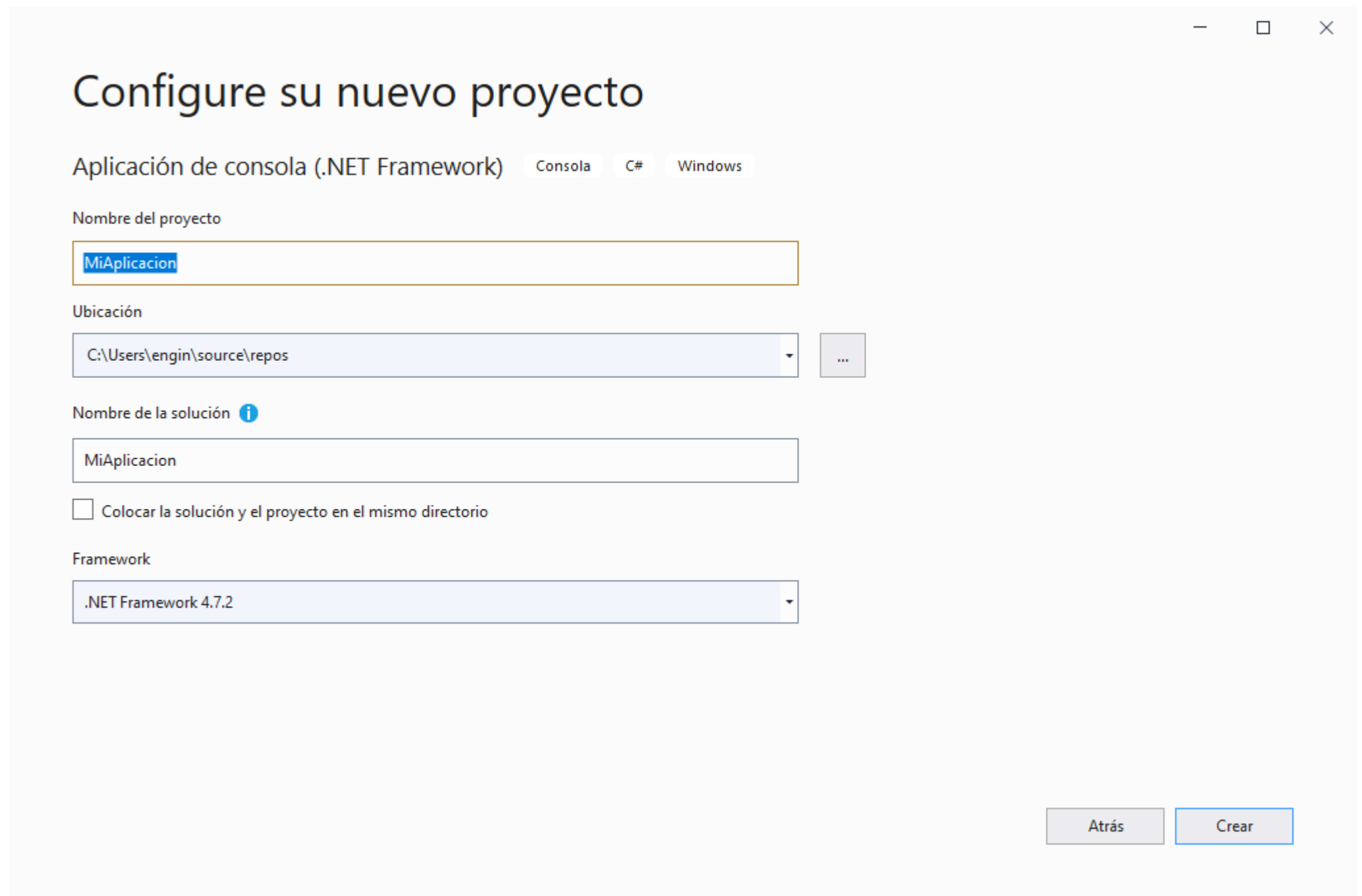


Ilustración 7: El filtro es C#, Windows, Consola

De allí se escoge "Aplicación de consola (.NET Framework).



Configure su nuevo proyecto

Aplicación de consola (.NET Framework) Consola C# Windows

Nombre del proyecto

MiAplicacion

Ubicación

C:\Users\engin\source\repos

Nombre de la solución ⓘ

MiAplicacion

☐ Colocar la solución y el proyecto en el mismo directorio

Framework

.NET Framework 4.7.2

Atrás Crear

Ilustración 8: Escriba el "Nombre del Proyecto", se recomienda sin espacios, ni tildes, ni Ñs, caracteres de A hasta Z

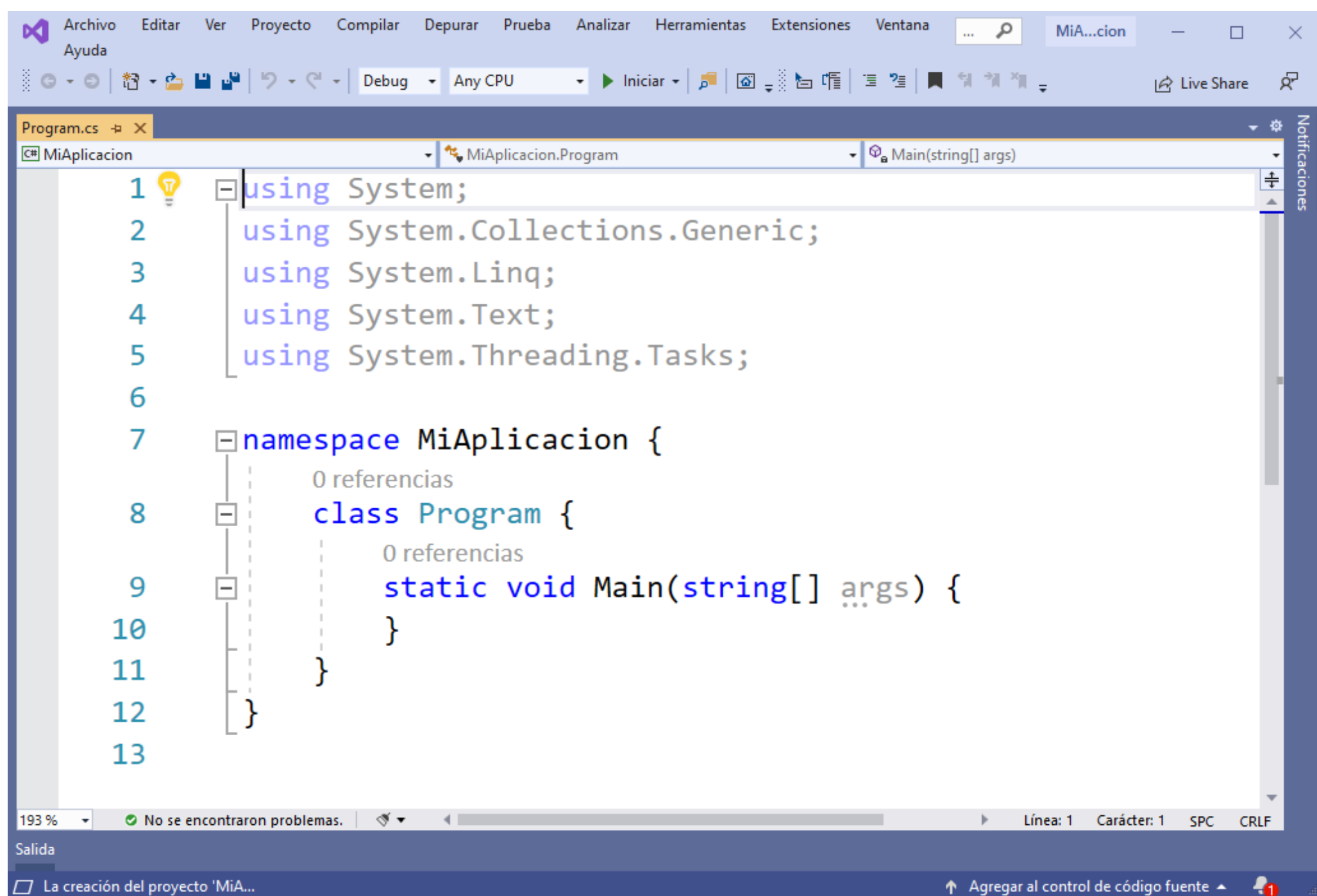


Ilustración 9: Plantilla por defecto de una aplicación de consola en C#

En la plantilla, se hace uso de varias librerías (se pueden ver porque empieza con “using”), se pueden borrar mientras tanto.

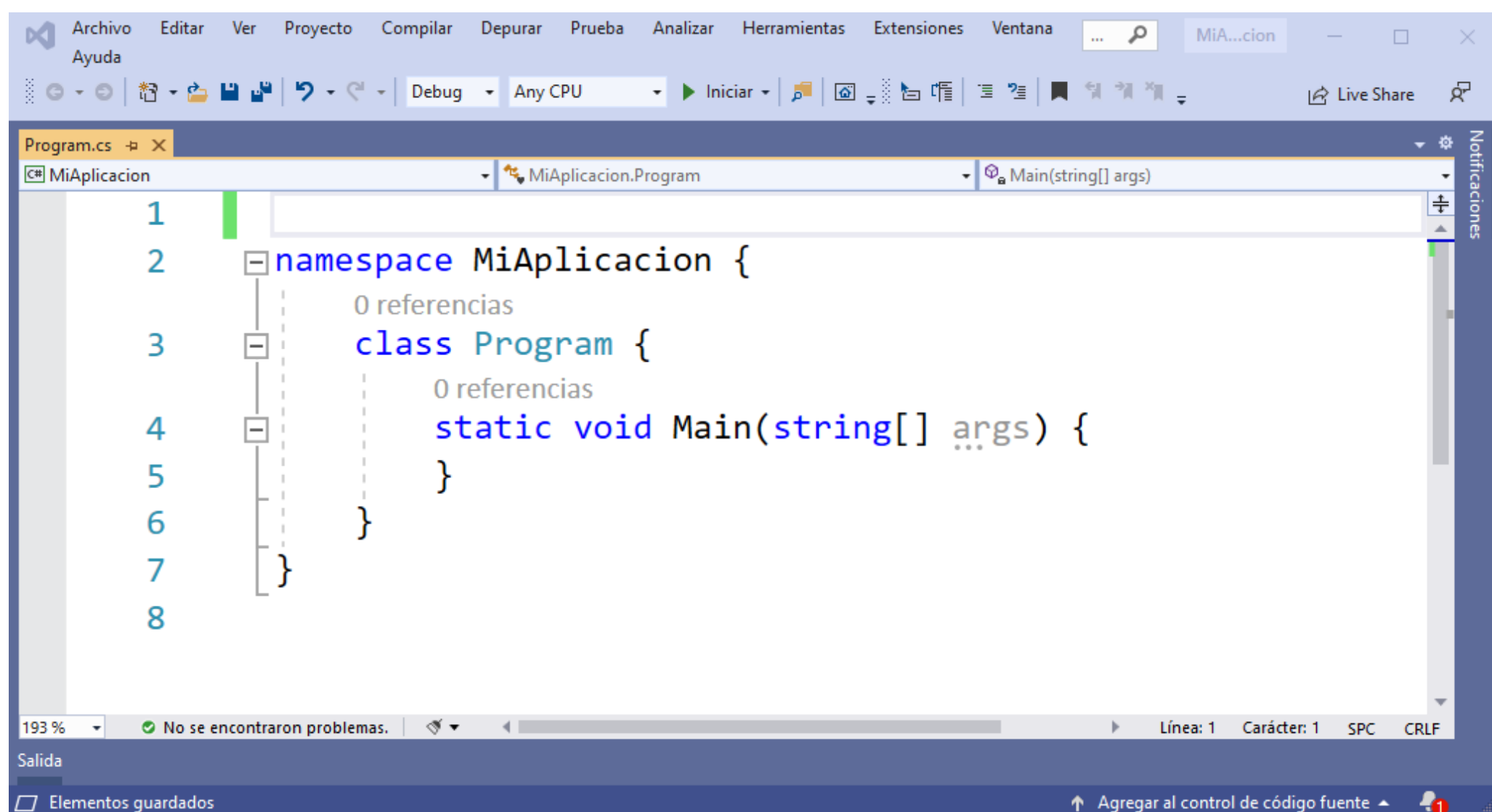


Ilustración 10: Se eliminan las librerías innecesarias por el momento

## Directorio o carpeta por defecto

El directorio o carpeta por defecto donde se crean los proyectos en Microsoft Visual Studio Community 2019 es:  
C:\users\<su usuario en Windows>\source\repos

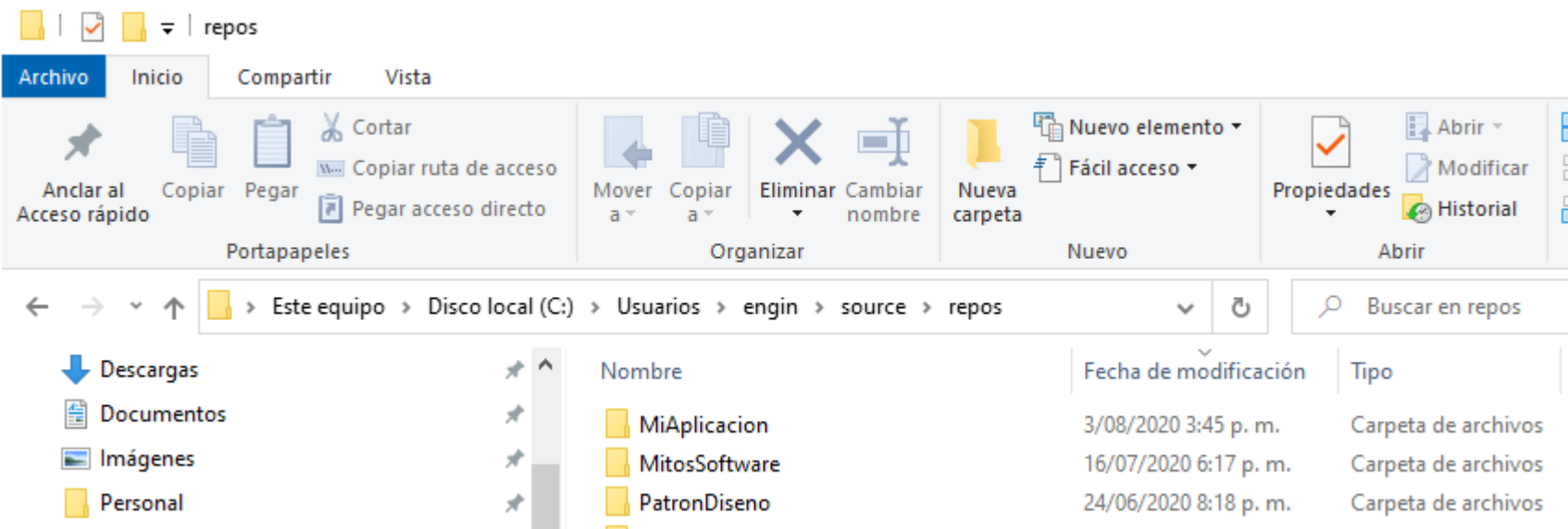


Ilustración 11: Directorio o carpeta por defecto

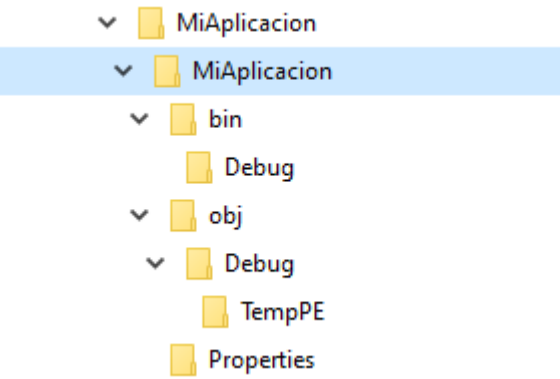


Ilustración 12: Árbol de subdirectorios creado

Nombre	Fecha de modificación	Tipo	Tamaño
MiAplicacion	3/08/2020 3:48 p. m.	Carpeta de archivos	
MiAplicacion.sln	3/08/2020 3:45 p. m.	Visual Studio Solu...	2 KB

Ilustración 13: Archivos en el directorio principal de la solución

Se recomienda ver las extensiones de los archivos en Windows, de esa forma vemos el primer archivo que tiene extensión (.sln) y es con esta extensión con la que se carga la solución en Visual Studio 2019

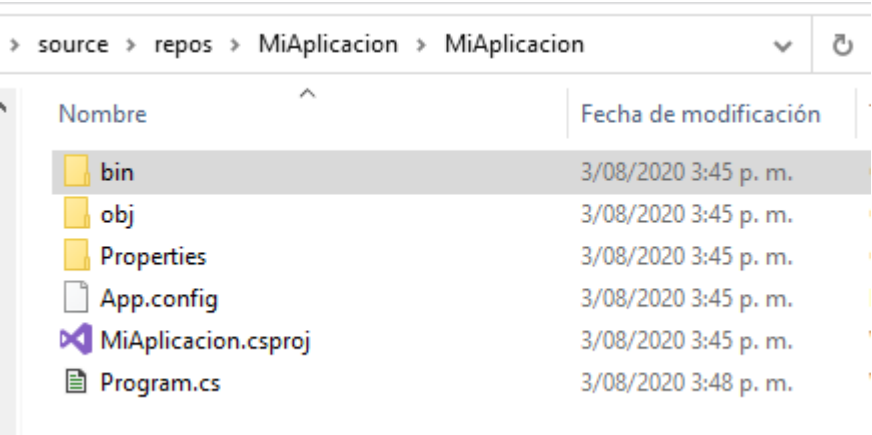


Ilustración 14: Los archivos fuente .cs se encuentran en este subdirectorio

Los archivos fuentes de C# tienen extensión (.cs).

Hay que guardar toda la carpeta de la solución, la que se encuentra en C:\users\<su usuario en Windows>\source\repos para poder abrirla desde Visual Studio Community 2019.

# La diferencia entre Debug y Release

En el IDE hay dos opciones para compilar y ejecutar el programa: Debug y Release

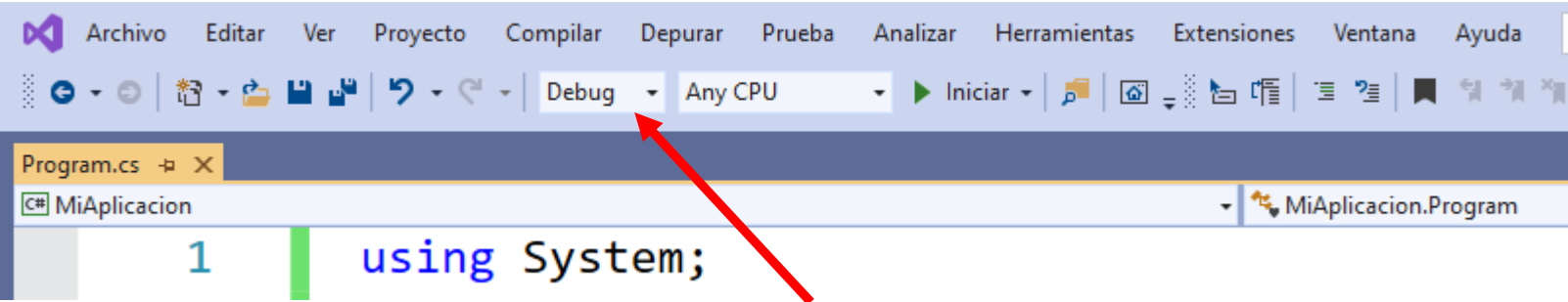


Ilustración 15: Por defecto, el programa compila y ejecuta con "Debug"

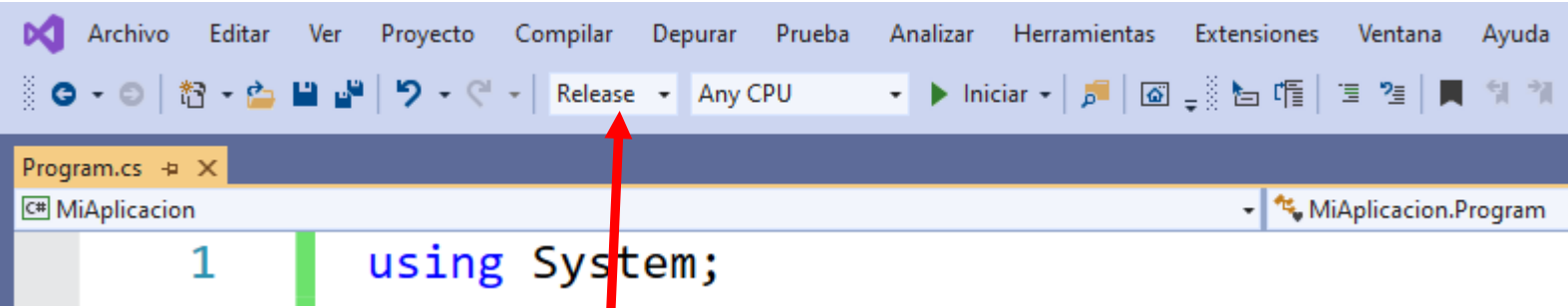


Ilustración 16: Compilación y ejecución por "Release"

En "Debug", el compilador agrega código para poder depurar, la ejecución será más lenta. En "Release", el compilador genera el ejecutable (\*.exe) listo para ser instalado en un cliente, no hay código de depuración y, por lo tanto, será más rápida la ejecución.

En el árbol de directorios se observa que hay uno para "Debug" y otro para "Release"

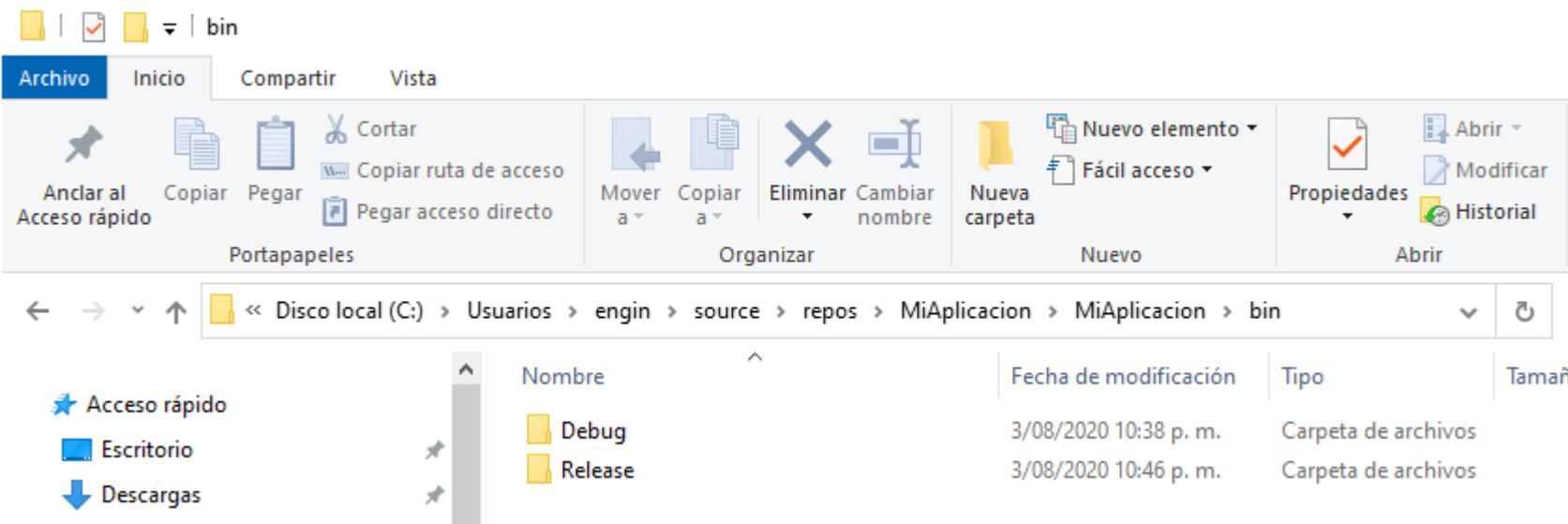


Ilustración 17: Directorios para la versión "debug" y "release"

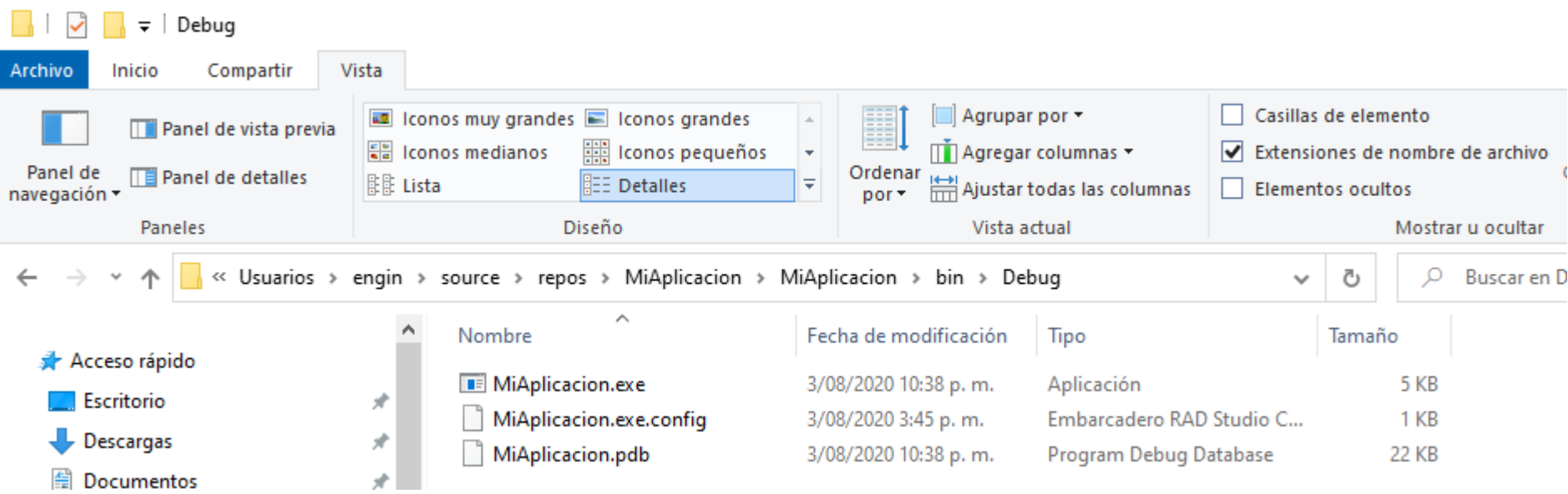


Ilustración 18: Ejecutable en el directorio "Debug"

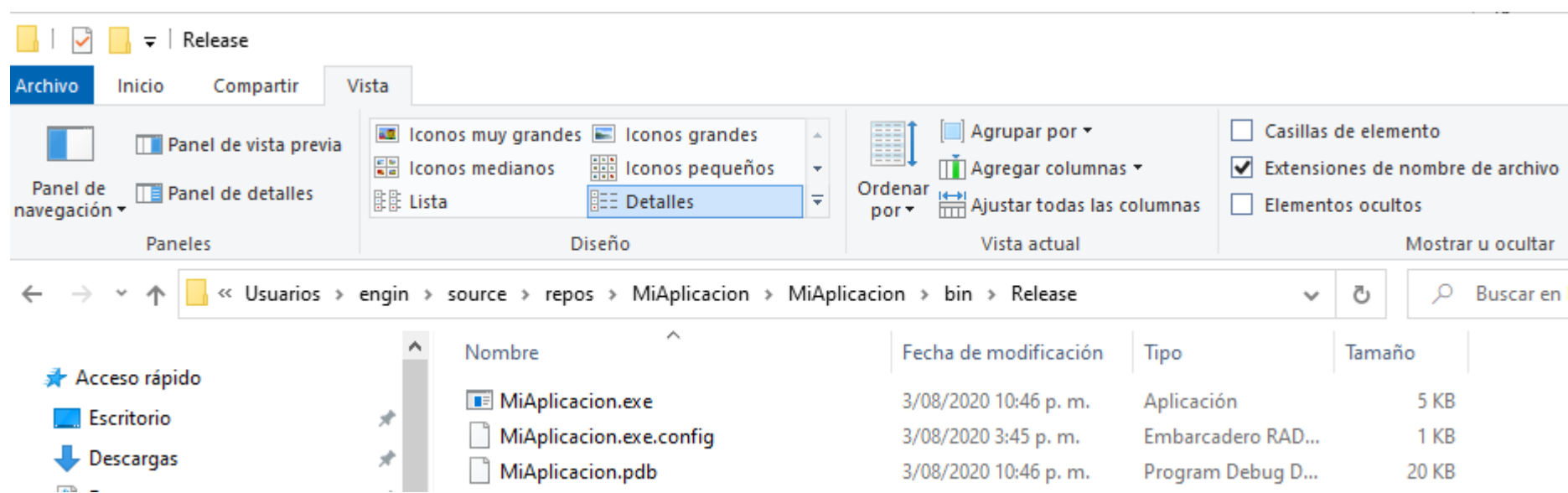


Ilustración 19: Ejecutable en el directorio "Release"

## Comentarios en el código

En C#, similar a lenguajes como C, C++ o Java, los comentarios pueden ser de una línea usando los caracteres `//` o de varias líneas iniciando con `/*` y finalizando con `*/`

001.cs

```
namespace Ejemplo {  
    class Program {  
        static void Main(string[] args) {  
            //Este es un comentario de una sola línea  
  
            /* Este es un  
               comentario de  
               varias líneas */  
        }  
    }  
}
```



## El tradicional “Hola Mundo”

Para mostrar datos en la consola se utiliza la instrucción Console.WriteLine o Console.Write, la diferencia entre ambas instrucciones es que la primera imprime y hace un salto de renglón, la segunda no hace ese salto de renglón.

Es necesario poner la instrucción Console.ReadKey(); al final del programa para evitar que cuando se ejecute, se cierre la ventana de consola inmediatamente.

Aquí es necesario usar una librería, por eso se escribe al inicio `using System;`

002.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Imprime en consola
            Console.Write("Hola Mundo");

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

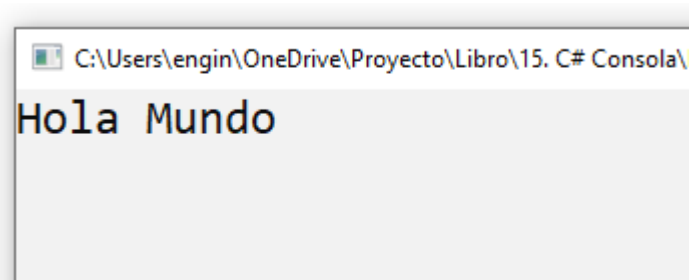


Ilustración 20: Impresión en consola

003.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Imprime en consola en dos líneas
            Console.WriteLine("Primera Línea");
            Console.WriteLine("Segunda Línea");

            //Imprime en la misma línea
            Console.Write("Este es un texto");
            Console.Write(" de una sola línea");

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

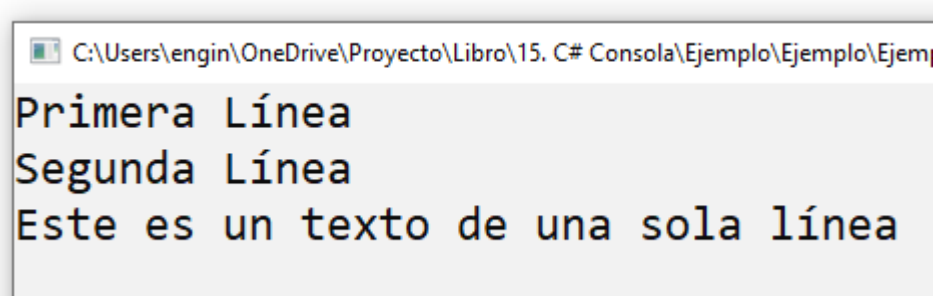


Ilustración 21: Impresión en una y varias líneas

# Variables

C# es un lenguaje de programación fuertemente tipado, luego es necesario declarar las variables con su tipo de dato.

004.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Declara variables de tipo entero
            int ladoA = 4;
            int ladoB = 8;
            int ladoC = 10;

            //Imprime esas variables
            Console.WriteLine("Lado A es: " + ladoA.ToString());
            Console.WriteLine("Lado B es: " + ladoB.ToString());
            Console.WriteLine("Lado C es: " + ladoC.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

Para imprimir el valor de una variable en consola, se debe poner el ".ToString()" porque la instrucción "Console.WriteLine" sólo acepta cadenas (strings).

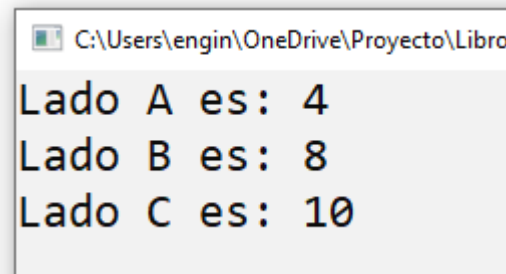


Ilustración 22: Imprime el valor de variables de tipo entero

005.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Declara variables de tipo double
            double valA = 5.718987654321;
            double valB = 3.420321098765;
            double valC = 1.23456789012345678901234567890123456789012345678901234567890123456789;

            //Imprime esas variables
            Console.WriteLine("Valor A es: " + valA.ToString());
            Console.WriteLine("Valor B es: " + valB.ToString());
            Console.WriteLine("Valor C es: " + valC.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

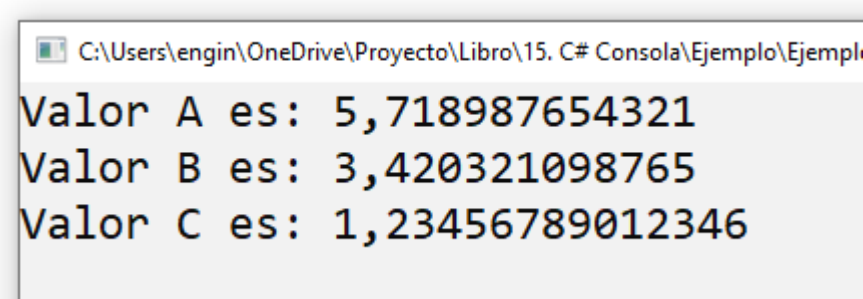


Ilustración 23: Imprime el valor de variables de tipo double, fíjese en el número de decimales de Valor C

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Declara variables de tipo char
            char letraA = 'R';
            char letraB = 'a';
            char letraC = 'f';

            //Imprime esas variables
            Console.WriteLine("Letra A es: " + letraA.ToString());
            Console.WriteLine("Letra B es: " + letraB.ToString());
            Console.WriteLine("Letra C es: " + letraC.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

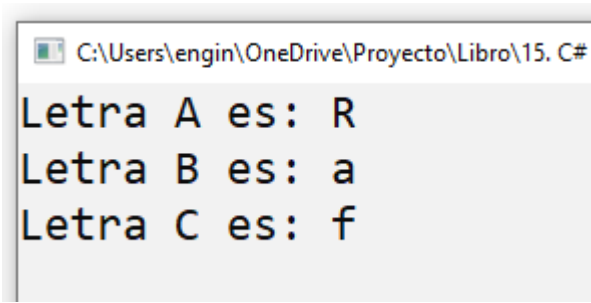


Ilustración 24: Imprimir valores de variables tipo char

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Declara variables de tipo string
            string cadenaA = "Esta es una prueba";
            string cadenaB = "Programando en C#";
            string cadenaC = "Con Visual Studio 2019";

            //Imprime esas variables
            Console.WriteLine("Cadena A es: " + cadenaA);
            Console.WriteLine("Cadena B es: " + cadenaB);
            Console.WriteLine("Cadena C es: " + cadenaC);

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

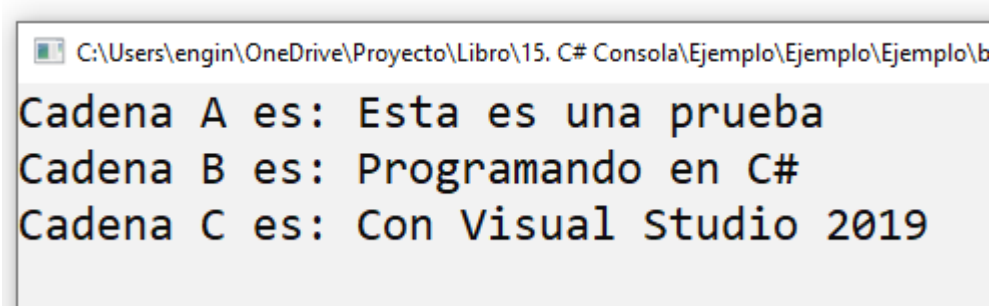


Ilustración 25: Imprimir valores de variables tipo cadena (string). No requiere el uso de .ToString()

Uso de variables tipo float, observe como su precisión es baja en comparación con las variables de tipo double.

008.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Declara variables de tipo float, hay que usar el cast (float)
            float valA = (float) 5.718987654321;
            float valB = (float) 3.420321098765;
            float valC = (float) 1.23456789012345678901234567890123456789012345678901234567890123456789;

            //Imprime esas variables
            Console.WriteLine("Valor A es: " + valA.ToString());
            Console.WriteLine("Valor B es: " + valB.ToString());
            Console.WriteLine("Valor C es: " + valC.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

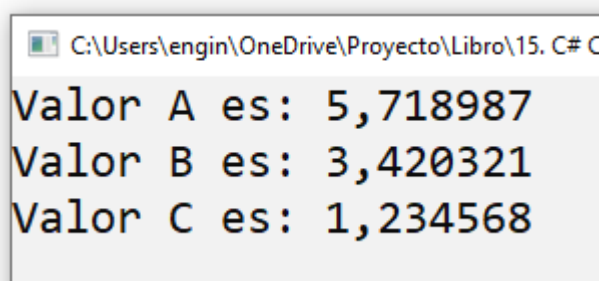


Ilustración 26: Imprimir valores de variables de tipo float. Se pierde precisión.

Por lo tanto, se recomienda hacer uso mejor de variables tipo double.

# Imprimiendo con formato

Se requiere imprimir los números en consola con un determinado formato. Este es el código:

009.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Declara variables de tipo double
            double valA = 5.718987654321;
            double valB = -3.420821098765;
            double valC = 1.234567890123456789;
            double valD = 7.8;

            //Imprime con un formato de al menos dos decimales
            Console.WriteLine("Valor A es: {0:0.00}", valA);

            //Imprime con un formato de al menos tres decimales
            Console.WriteLine("Valor B es: {0:0.000}", valB);

            //Imprime con un formato de al menos cuatro decimales
            Console.WriteLine("Valor C es: {0:0.0000}", valC);

            //Imprime con un formato de al menos cinco decimales
            Console.WriteLine("Valor D es: {0:0.00000}", valD);

            /* Imprime las tres variables. Ver el inicio {N: donde N es la posición de la variable
            * en la instrucción de impresión iniciando en cero. */
            Console.WriteLine("valA: {0:0.00}; valB: {1:0.00}; valC: {2:0.00}", valA, valB, valC);

            /* Máximo tres decimales */
            Console.WriteLine("Valor C con máximo tres decimales es: {0:0.###}", valC);
            Console.WriteLine("Valor D con máximo tres decimales es: {0:0.###}", valD);

            /* Mínimo tres dígitos antes del punto decimal */
            Console.WriteLine("Valor A con tres dígitos antes del punto decimal: {0:000.###}", valA);
            Console.WriteLine("Valor B con tres dígitos antes del punto decimal: {0:000.###}", valB);

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

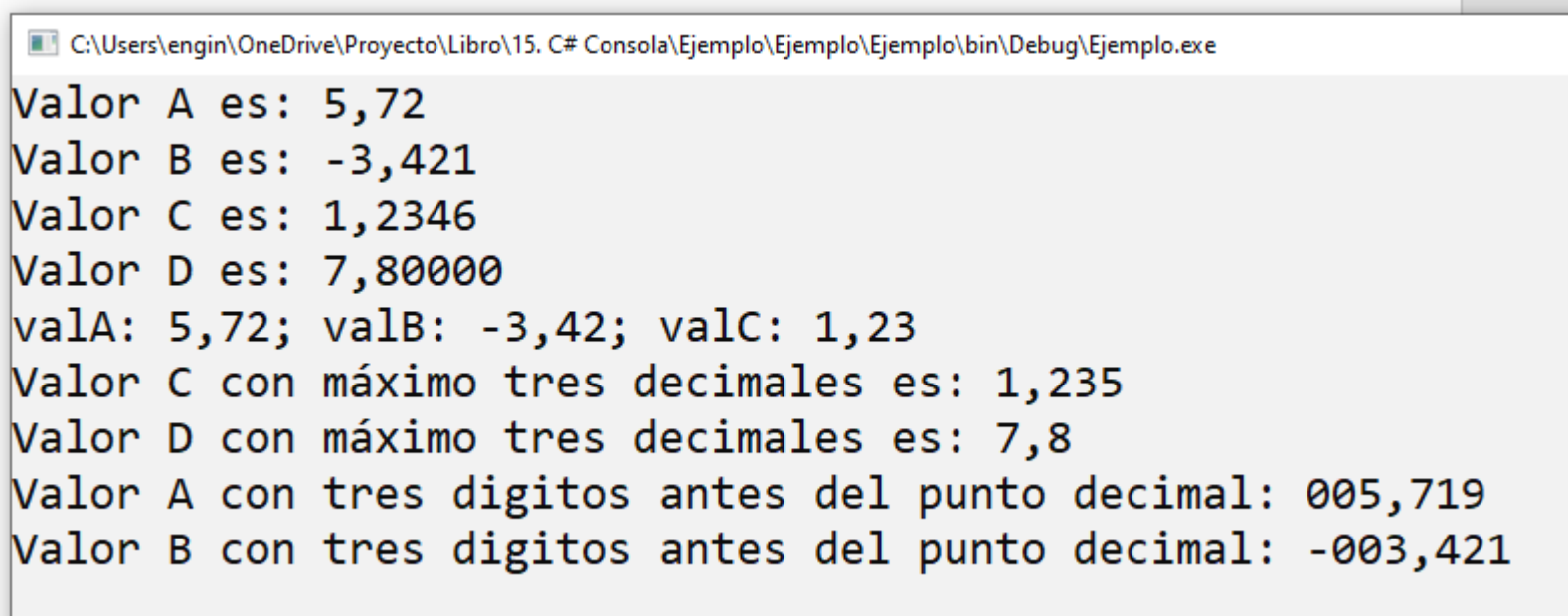


Ilustración 27: Impresión de valores con formato

```

using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Declara variables de tipo double
            double valA = 17902.8421;
            double valB = -871901372.420821098765;
            double valC = 89341759342.1678;

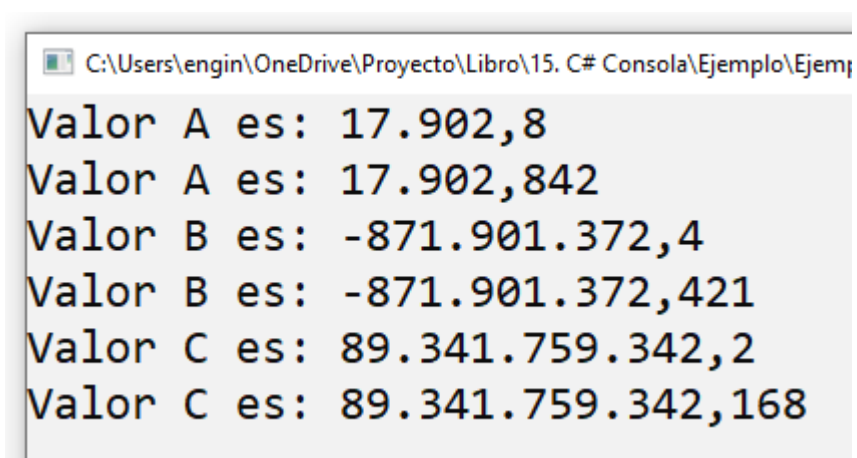
            //Imprime con un formato de miles
            Console.WriteLine("Valor A es: {0:0,0.0}", valA);
            Console.WriteLine("Valor A es: {0:0,0.000}", valA);

            Console.WriteLine("Valor B es: {0:0,0.0}", valB);
            Console.WriteLine("Valor B es: {0:0,0.000}", valB);

            Console.WriteLine("Valor C es: {0:0,0.0}", valC);
            Console.WriteLine("Valor C es: {0:0,0.000}", valC);

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}

```



```

C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejem...
Valor A es: 17.902,8
Valor A es: 17.902,842
Valor B es: -871.901.372,4
Valor B es: -871.901.372,421
Valor C es: 89.341.759.342,2
Valor C es: 89.341.759.342,168

```

Ilustración 28: Imprimir en formato de miles

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Declara variables de tipo double
            double valA = 17902.8421;
            double valB = -871901372.420821098765;
            double valC = 89341759342.1678;

            //Imprime con alineación a la derecha (20 espacios para poner el número)
            Console.WriteLine("Valor A es: {0,20:0.0}", valA);
            Console.WriteLine("Valor B es: {0,20:0.0}", valB);
            Console.WriteLine("Valor C es: {0,20:0.0}", valC);

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

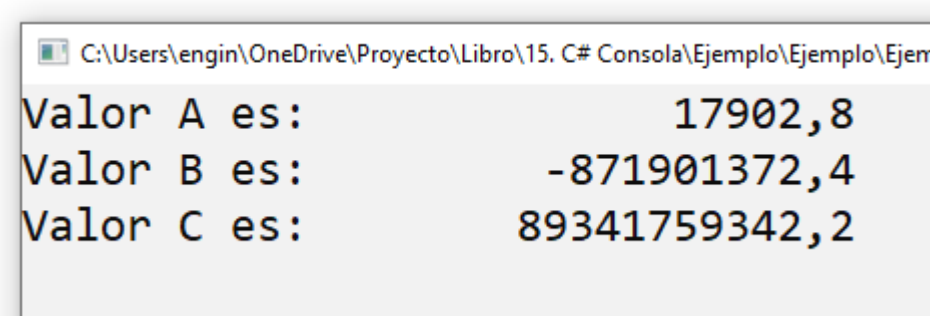


Ilustración 29: Imprimir con autocompletado de espacios a la izquierda

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Declara variables de tipo double
            double valA = 2.76;
            double valB = 2.04;
            double valC = 0.14;

            //Imprime si la cifra es significativa (redondea también)
            Console.WriteLine("Valor A es: {0:0.0}", valA);
            Console.WriteLine("Valor B es: {0:0.0}", valB);
            Console.WriteLine("Valor C es: {0:0.0}", valC);

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

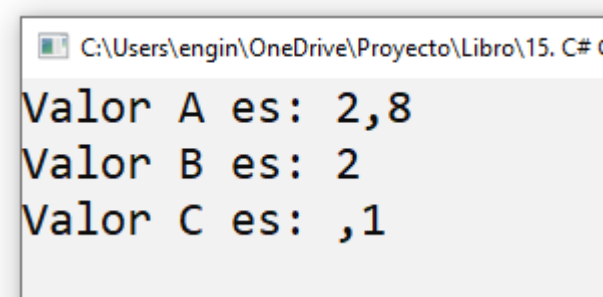


Ilustración 30: Impresión de valores con redondeo



```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            double valA = 1 + 2 + 3; //Operador suma
            double valB = 4 - 5 - 6; //Operador resta
            double valC = 7 * 8 * 9; //Operador multiplicación
            double valD = 178 / 2 / 3; //Operador división usando enteros
            double valE = 178 / 2.0 / 3.0; //Operador división usando números reales
            double valF = (double) 178 / 2 / 3; //Operador división usando números enteros haciendo cast
            double valG = 70 % 6; //Operador división modular

            Console.WriteLine("Valor A es: {0:0.0000}", valA);
            Console.WriteLine("Valor B es: {0:0.0000}", valB);
            Console.WriteLine("Valor C es: {0:0.0000}", valC);
            Console.WriteLine("Valor D es: {0:0.0000}", valD);
            Console.WriteLine("Valor E es: {0:0.0000}", valE);
            Console.WriteLine("Valor F es: {0:0.0000}", valF);
            Console.WriteLine("Valor G es: {0:0.0000}", valG);

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

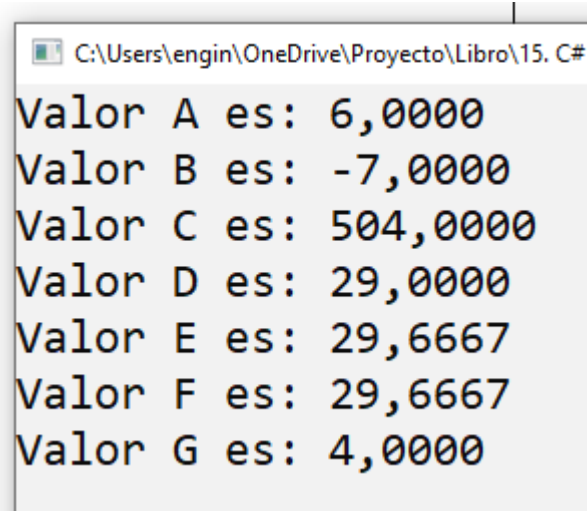


Ilustración 31: Resultados de operaciones matemáticas

Observe los resultados de valD, valE y valF, en valD el resultado es un entero porque C# lo toma como si fuese una división entera de números enteras por lo que el resultado es entero. En cambio, valE y valF retornan el resultado real, el primero por usar notación de número real (el uso del punto) y el otro por usar cast (double).

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Compara la precisión entre float y double
            float valA = (float) (1.7 / 2.8 * 4.8 / 6.7 / 5.3);
            double valB = 1.7 / 2.8 * 4.8 / 6.7 / 5.3;

            Console.WriteLine("Valor A es: " + valA.ToString());
            Console.WriteLine("Valor B es: " + valB.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

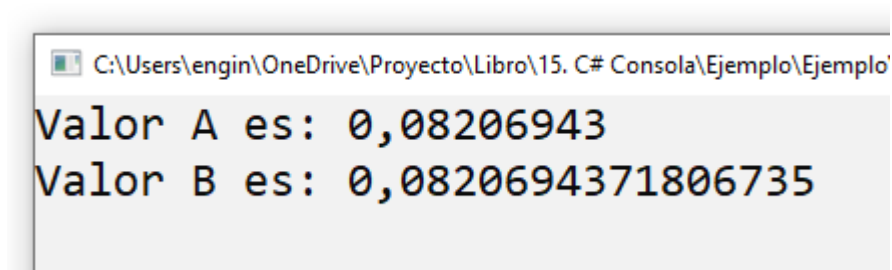


Ilustración 32: Diferencia de precisión entre float y double

Recomendado: Usar siempre double

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Potencia (^)
            double valA = Math.Pow(2, 5); // 2 ^ 5
            double valB = Math.Pow(729, (double)1 / 3); //Raíz cúbica de 729
            double valC = Math.Pow(64, (double)1 / 2); //Raíz cuadrada de 64
            double valD = Math.Pow(4, -2); // 4^(-2) = 1/(4^2) = 1/16
            double valE = Math.Pow(4, 0); // 4^0

            Console.WriteLine("Valor A es: " + valA.ToString());
            Console.WriteLine("Valor B es: " + valB.ToString());
            Console.WriteLine("Valor C es: " + valC.ToString());
            Console.WriteLine("Valor D es: " + valD.ToString());
            Console.WriteLine("Valor E es: " + valE.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

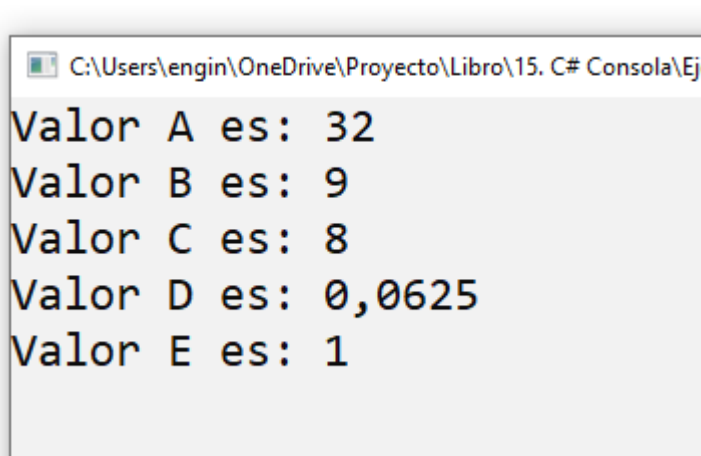


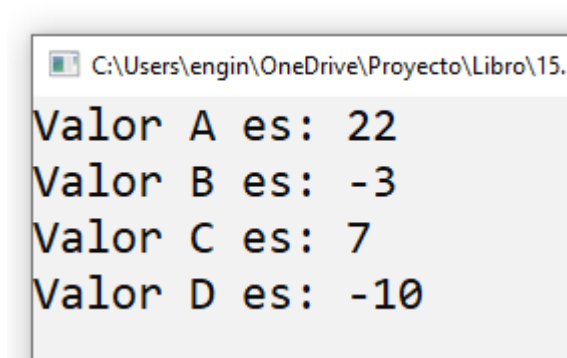
Ilustración 33: Función de potencia

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            /* Orden de evaluación de los operadores:
             * Primero potencia
             * Segundo multiplicación y división
             * Tercero suma y resta
             * */
            double valA = (double)7 - 1 + 2 * Math.Pow(2, 5) / 4;
            double valB = (double)1 + 2 * 3 / 4 - 5;
            double valC = (double)3 + 5 - 2 * 4 / 8;
            double valD = (double)3 * 2 + 5 / 10 - 2 * Math.Pow(3, 2) + 4 * 4 / 8;

            Console.WriteLine("Valor A es: " + valA.ToString());
            Console.WriteLine("Valor B es: " + valB.ToString());
            Console.WriteLine("Valor C es: " + valC.ToString());
            Console.WriteLine("Valor D es: " + valD.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```



```
C:\Users\engin\OneDrive\Proyecto\Libro\15.
Valor A es: 22
Valor B es: -3
Valor C es: 7
Valor D es: -10
```

Ilustración 34: Orden de evaluación de operadores

```
using System;

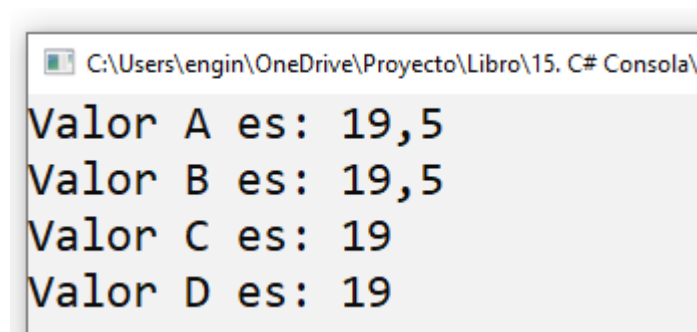
namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Es mejor usar reales que el cast
            double valA = 9.0 / 2.0 - 1.0 + 2.0 * Math.Pow(2.0, 5.0) / 4.0;
            double valB = (double)9 / 2 - 1 + 2 * Math.Pow(2, 5) / 4;

            //Falla porque se interpreta completamente como enteros 9 / 2 es 4 en división entera
            double valC = 9 / 2 - 1 + 2 * Math.Pow(2, 5) / 4;

            //Falla el cast porque primero se interpreta la expresión como entera y luego se pasa a double
            double valD = (double) (9 / 2 - 1 + 2 * Math.Pow(2, 5) / 4);

            Console.WriteLine("Valor A es: " + valA.ToString());
            Console.WriteLine("Valor B es: " + valB.ToString());
            Console.WriteLine("Valor C es: " + valC.ToString());
            Console.WriteLine("Valor D es: " + valD.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```



```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\
Valor A es: 19,5
Valor B es: 19,5
Valor C es: 19
Valor D es: 19
```

Ilustración 35: Cuidado al usar el cast

```

using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Falla porque se interpreta como operación de enteros
            double valA = (4 / 3) + (1 / 2) - (25 / 4);

            //Falla porque el cast solo cubre la primera operación
            double valB = (double)4 / 3 + (1 / 2) - (25 / 4);

            //Operación correcta
            double valC = (4.0 / 3.0) + (1.0 / 2.0) - (25.0 / 4.0);

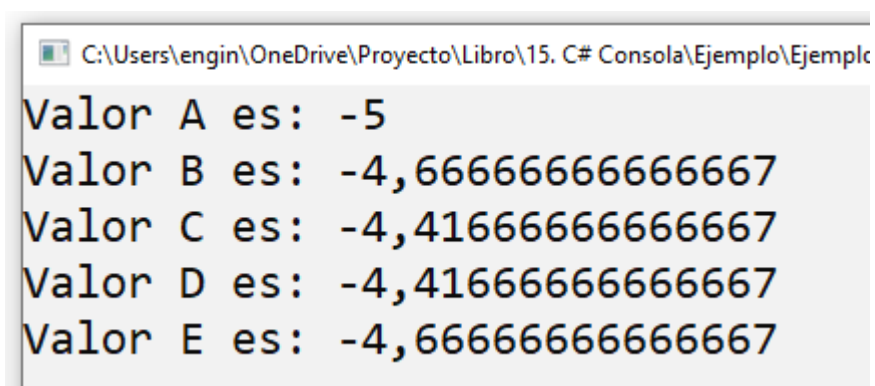
            //Operación correcta pero se deben usar varios cast
            double valD = (double)4 / 3 + (double)1 / 2 - (double)25 / 4;

            //Falla porque el cast solo cubre la primera operación
            double valE = (double)4 / 3 + 1 / 2 - 25 / 4;

            Console.WriteLine("Valor A es: " + valA.ToString());
            Console.WriteLine("Valor B es: " + valB.ToString());
            Console.WriteLine("Valor C es: " + valC.ToString());
            Console.WriteLine("Valor D es: " + valD.ToString());
            Console.WriteLine("Valor E es: " + valE.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}

```



```

C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo
Valor A es: -5
Valor B es: -4,666666666666667
Valor C es: -4,416666666666667
Valor D es: -4,416666666666667
Valor E es: -4,666666666666667

```

Ilustración 36: Cuidado al usar el cast

Formato Algebraico	Formato en horizontal
$Y = \frac{3}{X-1}$	Y=3/(X-1)
$Y = \frac{3}{X-1} + 2$	Y=3/(X-1)+2
$Y = \frac{3}{X} + \frac{X}{7}$	Y=3/X+X/7
$Y = \frac{3}{X^2}$	Y=3/(X*X)
$Y = \frac{3}{2-(X^2+5)}$	Y=3/(2-(X*X+5))
$Y = \frac{5+X}{6-X^3}$	Y=(5+X)/(6-X*X*X)
$Y = 2 + \frac{\frac{4}{X}}{3}$	Y=2+((4/X)/3)
$Y = 5 * \frac{\frac{X^2}{4}}{\frac{3}{X+5}}$	Y=5*((X*X/4)/(3/(X+5)))
$Y = \frac{X^2 + \frac{2X-5}{X+1}}{1-X^2}$	Y=((X*X)+((2*X-5)/(X+1)))/(1-X*X)

019.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Pasar correctamente ecuaciones a formato en C#
            double X = 5.7;
            double Y;

            //Ejemplos
            Y = 3 / (X - 1);
            Console.WriteLine("Valor Y es: " + Y.ToString());

            Y = 3 / (X - 1) + 2;
            Console.WriteLine("Valor Y es: " + Y.ToString());

            Y = 3 / X + X / 7;
            Console.WriteLine("Valor Y es: " + Y.ToString());

            Y = 3 / (X * X);
            Console.WriteLine("Valor Y es: " + Y.ToString());

            Y = 3 / (2 - (X * X + 5));
            Console.WriteLine("Valor Y es: " + Y.ToString());

            Y = (5 + X) / (6 - X * X * X);
            Console.WriteLine("Valor Y es: " + Y.ToString());

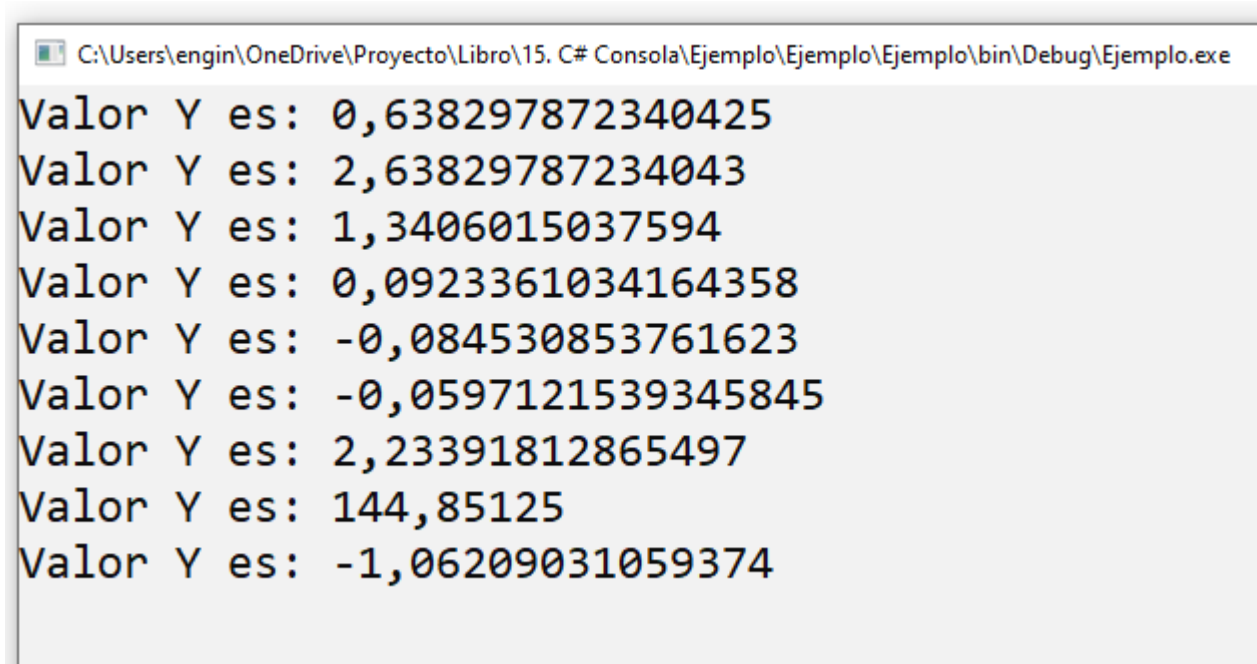
            Y = 2 + ((4 / X) / 3);
            Console.WriteLine("Valor Y es: " + Y.ToString());

            Y = 5 * ((X * X / 4) / (3 / (X + 5)));
            Console.WriteLine("Valor Y es: " + Y.ToString());

            Y = ((X * X) + ((2 * X - 5) / (X + 1))) / (1 - X * X);
            Console.WriteLine("Valor Y es: " + Y.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```





A screenshot of a Windows console window. The title bar shows the file path: C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\bin\Debug\Ejemplo.exe. The console output consists of ten lines, each starting with 'Valor Y es: ' followed by a numerical value in a C#-evaluable format (using commas as decimal separators and scientific notation for large numbers).

```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\bin\Debug\Ejemplo.exe
Valor Y es: 0,638297872340425
Valor Y es: 2,63829787234043
Valor Y es: 1,3406015037594
Valor Y es: 0,0923361034164358
Valor Y es: -0,084530853761623
Valor Y es: -0,0597121539345845
Valor Y es: 2,23391812865497
Valor Y es: 144,85125
Valor Y es: -1,06209031059374
```

*Ilustración 37: De formato algebraico a formato evaluable por C#*

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Conversión vs Cast
            double valA = 15.7;

            int valB = Convert.ToInt32(valA); //Redondea
            int valC = (int)valA; //Trunca

            //Ejemplos
            Console.WriteLine("Valor B es: " + valB.ToString());
            Console.WriteLine("Valor C es: " + valC.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

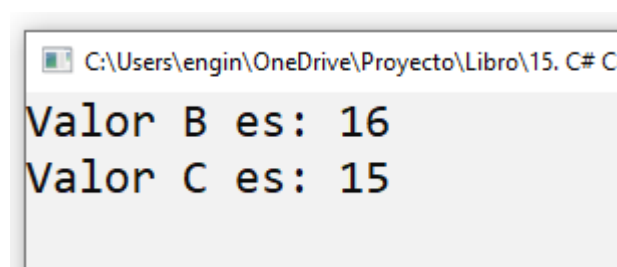


Ilustración 38: Diferencias entre el cast (trunca) y la conversión (redondea)

```
using System;
using System.Globalization; //Nueva librería

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Conversión de reales
            string valA = "4.78";

            //Da a problemas porque se ignora el punto decimal
            double valB = Convert.ToDouble(valA);
            Console.WriteLine("Valor B es: " + valB.ToString());

            //Aquí si funciona la conversión
            string valC = "9,21";
            double valD = Convert.ToDouble(valC);
            Console.WriteLine("Valor D es: " + valD.ToString());

            //Para usar la conversión con punto decimal, se debe hacer uso de CultureInfo
            string valE = "6.8315";
            double valF = double.Parse(valE, CultureInfo.InvariantCulture);
            Console.WriteLine("Valor F es: " + valF.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

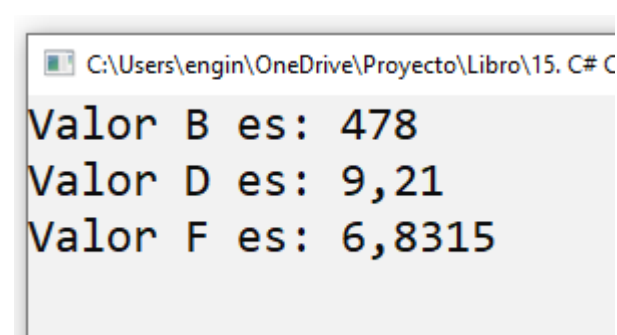


Ilustración 39: Uso de CultureInfo para conversión de números reales

```
using System;
using System.Globalization; //Nueva librería

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Fallos de conversión
            string valA = ""; //Una cadena nula o vacía daña la conversión

            //El programa se cae
            double valB = Convert.ToDouble(valA);
            Console.WriteLine("Valor B es: " + valB.ToString());

            //El programa se cae
            double valF = double.Parse(valA, CultureInfo.InvariantCulture);
            Console.WriteLine("Valor F es: " + valF.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

```
//El programa se cae
double valB = Convert.ToDouble(valA);
Console.WriteLine("Valor B es: " + valB.ToString());

//El programa se cae
double valF = double.Parse(valA, CultureInfo.InvariantCulture);
Console.WriteLine("Valor F es: " + valF.ToString());

//Esta instrucción detiene que se cierre la ventana de consola
Console.ReadKey();
```

Excepción no controlada

**System.FormatException:** 'La cadena de entrada no tiene el formato correcto.'

Esta excepción se generó originalmente en esta pila de llamadas:  
System.Number.ParseDouble(string, System.Globalization.NumberStyles, System.Globalization.NumberFormatInfo, System.Globalization.NumberFormatInfo)  
System.Convert.ToDouble(string)

Ilustración 40: Programa se cae por fallo en la conversión

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Constantes matemáticas
            double valA = Math.PI;
            double valB = Math.E;
            Console.WriteLine("PI es: " + valA.ToString());
            Console.WriteLine("E es: " + valB.ToString());

            //Máximo y mínimo valor entero
            int maximoEntero = int.MaxValue;
            int minimoEntero = int.MinValue;
            Console.WriteLine("Máximo entero: " + maximoEntero.ToString());
            Console.WriteLine("Mínimo entero: " + minimoEntero.ToString());

            //Máximo y mínimo valor float
            float maximofloat = float.MaxValue;
            float minimofloat = float.MinValue;
            float minimoCero = float.Epsilon; //El mínimo valor antes de ser cero
            Console.WriteLine("Máximo float: " + maximofloat.ToString());
            Console.WriteLine("Mínimo float: " + minimofloat.ToString());
            Console.WriteLine("El mínimo float antes de ser cero: " + minimoCero.ToString());

            //Máximo y mínimo valor double
            double maximodouble = double.MaxValue;
            double minimodouble = double.MinValue;
            double minimoCerodouble = double.Epsilon; //El mínimo valor antes de ser cero
            Console.WriteLine("Máximo double: " + maximodouble.ToString());
            Console.WriteLine("Mínimo double: " + minimodouble.ToString());
            Console.WriteLine("El mínimo double antes de ser cero: " + minimoCerodouble.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

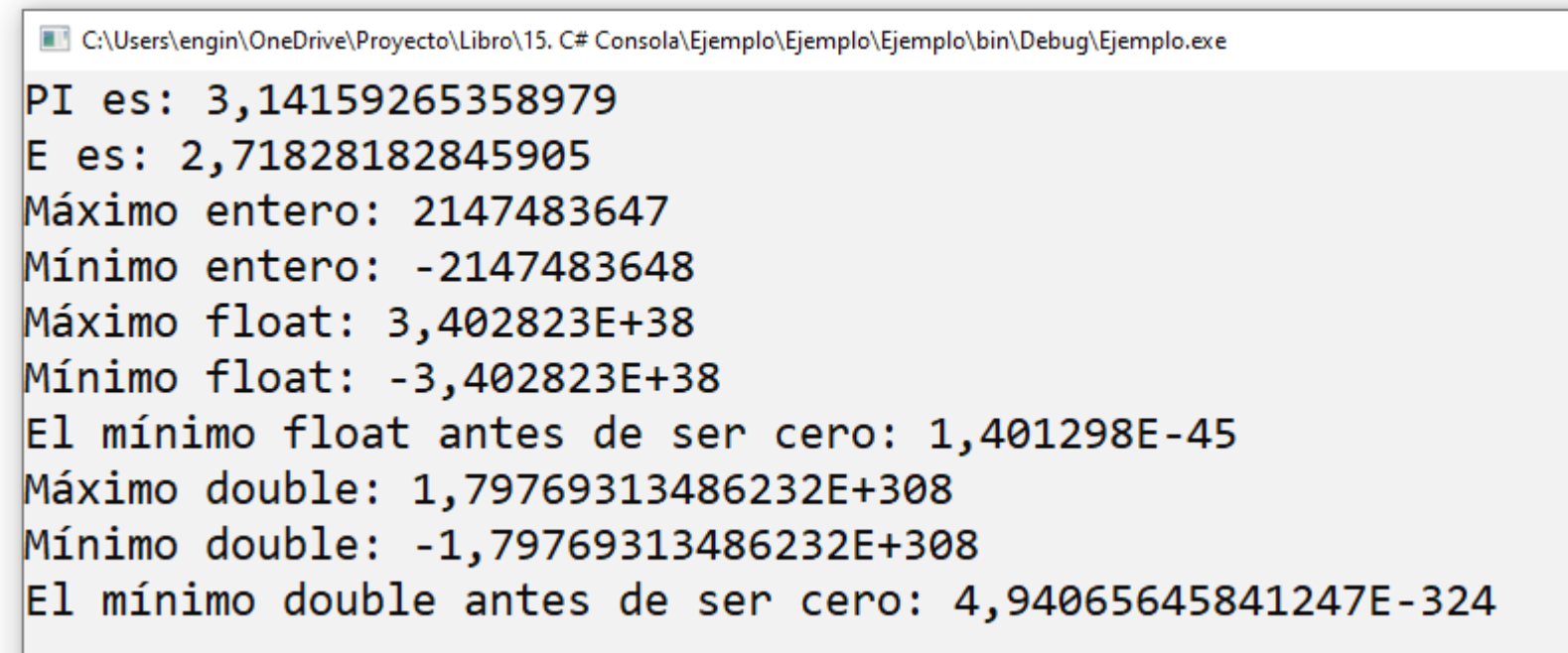


Ilustración 41: Constantes matemáticas

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Funciones trigonométricas
            double anguloGrados = 60;
            double anguloRadian = anguloGrados * Math.PI / 180;

            double valSeno = Math.Sin(anguloRadian);
            double valCoseno = Math.Cos(anguloRadian);
            double valTangente = Math.Tan(anguloRadian);

            double arcoSeno = Math.Asin(valSeno);
            double arcoCoseno = Math.Acos(valCoseno);
            double arcoTangente = Math.Atan(valTangente);

            Console.WriteLine("Ángulo en grados es: " + anguloGrados.ToString());
            Console.WriteLine("Ángulo en radianes: " + anguloRadian.ToString());
            Console.WriteLine("Seno es: " + valSeno.ToString());
            Console.WriteLine("Coseno es: " + valCoseno.ToString());
            Console.WriteLine("Tangente es: " + valTangente.ToString());
            Console.WriteLine("arcoSeno es: " + arcoSeno.ToString());
            Console.WriteLine("arcoCoseno es: " + arcoCoseno.ToString());
            Console.WriteLine("arcoTangente es: " + arcoTangente.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

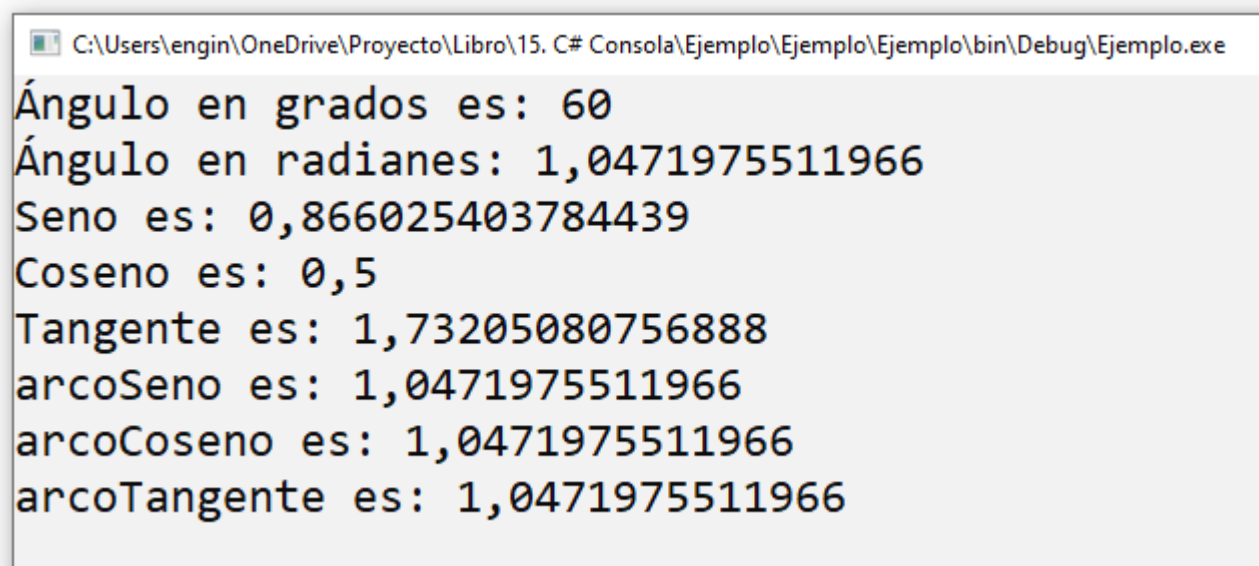


Ilustración 42: Funciones trigonométricas

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Funciones hiperbólicas
            double valorReal = -5.4713;

            double valSenoH = Math.Sinh(valorReal);
            double valCosenoH = Math.Cosh(valorReal);
            double valTangenteH = Math.Tanh(valorReal);

            Console.WriteLine("Valor: " + valorReal.ToString());
            Console.WriteLine("Seno hiperbólico es: " + valSenoH.ToString());
            Console.WriteLine("Coseno hiperbólico es: " + valCosenoH.ToString());
            Console.WriteLine("Tangente hiperbólico es: " + valTangenteH.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

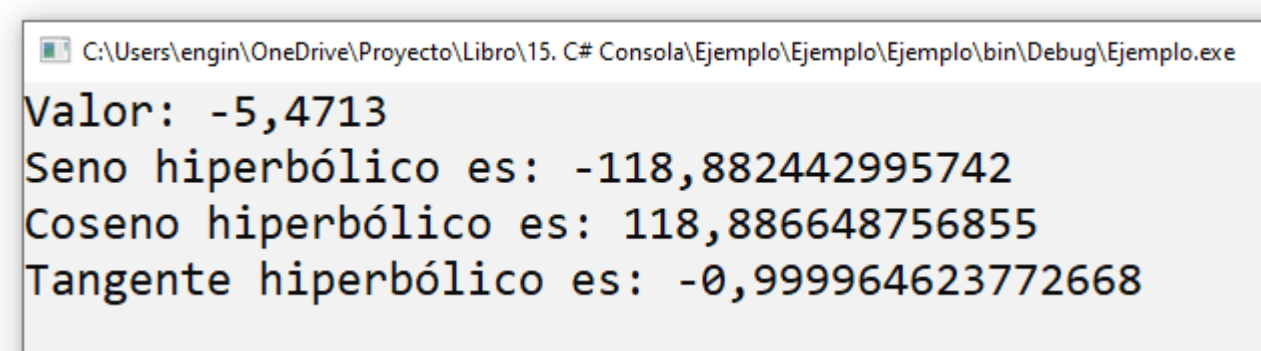


Ilustración 43: Funciones hiperbólicas



```

using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Otras funciones matemáticas
            double valorReal = 5.0713;
            double valAbsoluto = Math.Abs(valorReal); //valor absoluto
            double valTecho = Math.Ceiling(valorReal); //entero superior
            double valExponencial = Math.Exp(valorReal);
            double valPiso = Math.Floor(valorReal);
            double valLogaritmoNatural = Math.Log(valorReal);
            double valLogaritmoBase10 = Math.Log10(valorReal);
            double valMaximoEntero = Math.Max(7, 19);
            double valMinimoEntero = Math.Min(7, 19);
            double valRedondea = Math.Round(valorReal);
            double valSigno = Math.Sign(valorReal);
            double valRaizC = Math.Sqrt(valorReal); //raiz cuadrada
            double valTrunca = Math.Truncate(valorReal);

            Console.WriteLine("Valor: " + valorReal.ToString());
            Console.WriteLine("Valor absoluto es: " + valAbsoluto.ToString());
            Console.WriteLine("Valor techo es: " + valTecho.ToString());
            Console.WriteLine("Exponencial es: " + valExponencial.ToString());
            Console.WriteLine("Valor piso es: " + valPiso.ToString());
            Console.WriteLine("Logaritmo Natural es: " + valLogaritmoNatural.ToString());
            Console.WriteLine("Logaritmo Base 10 es: " + valLogaritmoBase10.ToString());
            Console.WriteLine("Máximo entero es: " + valMaximoEntero.ToString());
            Console.WriteLine("Mínimo entero es: " + valMinimoEntero.ToString());
            Console.WriteLine("Redondea: " + valRedondea.ToString());
            Console.WriteLine("Signo es: " + valSigno.ToString());
            Console.WriteLine("Raiz cuadrada es: " + valRaizC.ToString());
            Console.WriteLine("Trunca: " + valTrunca.ToString());

            //Funciones con dos salidas
            int residuo;
            int cociente = Math.DivRem(29, 4, out residuo);
            Console.WriteLine("29/4 cociente: " + cociente.ToString() + " residuo: " +
residuo.ToString());

            //Multiplicación enorme
            long valMultiplica = Math.BigMul(123456789, 987654321);
            Console.WriteLine("Multiplicación enorme: " + valMultiplica.ToString());

            //División modular, operación y funciones
            decimal dividendo = 100, divisor = 34;
            decimal residuoA = dividendo % divisor;
            decimal residuoB = (Math.Abs(dividendo) - (Math.Abs(divisor) * (Math.Floor(Math.Abs(dividendo)
/ Math.Abs(divisor)))) * Math.Sign(dividendo);
            Console.WriteLine("ResiduoA: " + residuoA.ToString() + " ResiduoB: " + residuoB.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}

```

```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\Ejemplo\bin\Debug\Ej  
Valor: 5,0713  
Valor absoluto es: 5,0713  
Valor techo es: 6  
Exponencial es: 159,381388529453  
Valor piso es: 5  
Logaritmo Natural es: 1,62359719499201  
Logaritmo Base 10 es: 0,705119302618628  
Máximo entero es: 19  
Mínimo entero es: 7  
Redondea: 5  
Signo es: 1  
Raiz cuadrada es: 2,25195470647169  
Trunca: 5  
29/4 cociente: 7 residuo: 1  
Multiplicación enorme: 121932631112635269  
ResiduoA: 32 ResiduoB: 32
```

Ilustración 44: Otras funciones matemáticas

# Manejo del NaN (Not a Number) y el infinito

Cuando algunas operaciones matemáticas generan error (por ejemplo, raíz cuadrada de un número negativo), la función retorna NaN (Not a Number). También hay operaciones que dan infinito como la división entre cero, en ese caso en consola se muestra como si fuese el número 8 (ocho), y es más bien un infinito girado 90 grados.

027.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            // Genera un NaN: Not a Number
            double valorReal = -70;
            double valLogaritmoNatural = Math.Log(valorReal);
            double valLogaritmoBase10 = Math.Log10(valorReal);
            double valRaizC = Math.Sqrt(valorReal); //raiz cuadrada
            double arcoSeno = Math.Asin(valorReal);
            double arcoCoseno = Math.Acos(valorReal);

            Console.WriteLine("Valor: " + valorReal.ToString());
            Console.WriteLine("Logaritmo Natural es: " + valLogaritmoNatural.ToString());
            Console.WriteLine("Logaritmo Base 10 es: " + valLogaritmoBase10.ToString());
            Console.WriteLine("Raiz cuadrada es: " + valRaizC.ToString());
            Console.WriteLine("Arcoseno es: " + arcoSeno.ToString());
            Console.WriteLine("Arcocoseno es: " + arcoCoseno.ToString());

            //Genera infinito positivo
            float valA = 10;
            float valB = 0;
            float valC = valA / valB;
            Console.WriteLine("Valor C es: " + valC);

            //Genera infinito negativo
            valA = -10;
            valB = 0;
            valC = valA / valB;
            Console.WriteLine("Valor C es: " + valC);

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

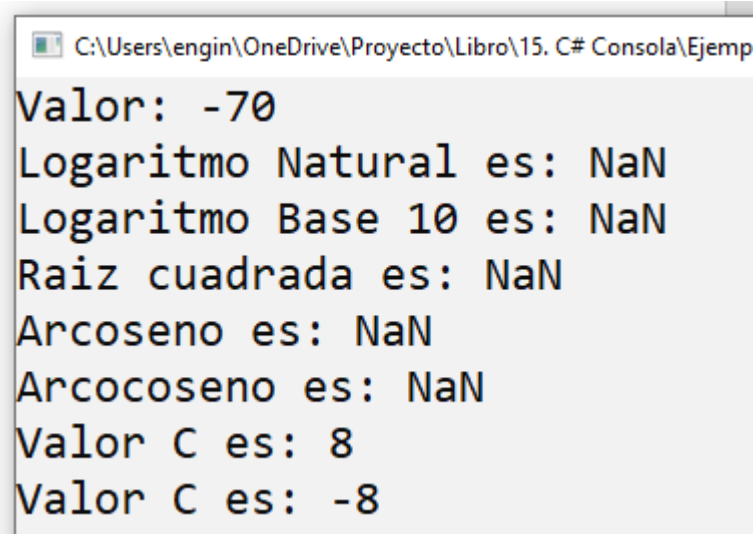


Ilustración 45: Manejo del NaN y el infinito (girado 90 grados)

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Leer un número por consola
            Console.Write("Escriba un número: ");
            double valorReal = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("Escribió: " + valorReal.ToString());

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

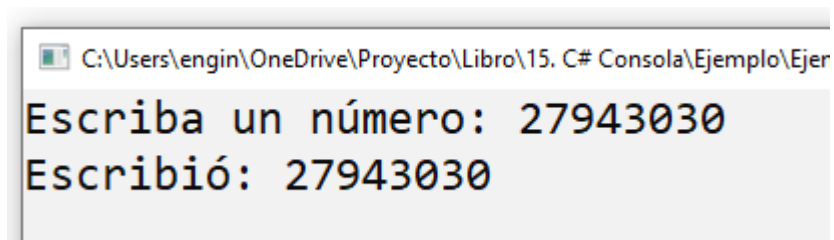


Ilustración 46: Leer un número del teclado

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Lee dos números por consola
            Console.Write("Escriba un primer número: ");
            double valorA = Convert.ToDouble(Console.ReadLine());

            Console.Write("Escriba un segundo número: ");
            double valorB = Convert.ToDouble(Console.ReadLine());

            //Si condicional
            if (valorA > valorB) {
                Console.WriteLine(valorA.ToString() + " es mayor que " + valorB.ToString());
            }

            if (valorA >= valorB) {
                Console.WriteLine(valorA.ToString() + " es mayor o igual que " + valorB.ToString());
            }

            if (valorA < valorB) {
                Console.WriteLine(valorA.ToString() + " es menor que " + valorB.ToString());
            }

            if (valorA <= valorB) {
                Console.WriteLine(valorA.ToString() + " es menor o igual que " + valorB.ToString());
            }

            if (valorA == valorB) {
                Console.WriteLine(valorA.ToString() + " es igual a " + valorB.ToString());
            }

            if (valorA != valorB) {
                Console.WriteLine(valorA.ToString() + " es diferente de " + valorB.ToString());
            }

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

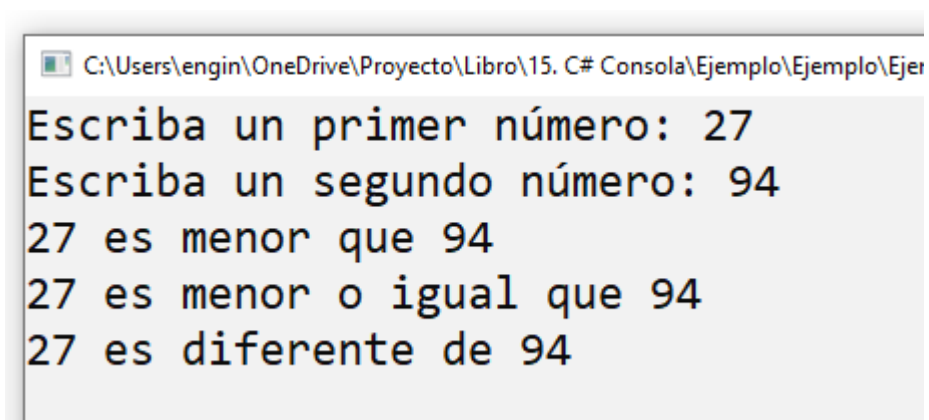


Ilustración 47: Si condicional

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Lee dos números por consola
            Console.Write("Escriba un primer número: ");
            double valorA = Convert.ToDouble(Console.ReadLine());

            Console.Write("Escriba un segundo número: ");
            double valorB = Convert.ToDouble(Console.ReadLine());

            //Si condicional
            if (valorA > valorB) {
                Console.WriteLine(valorA.ToString() + " es mayor que " + valorB.ToString());
            }
            else { //de lo contrario
                Console.WriteLine(valorA.ToString() + " es menor o igual que " + valorB.ToString());
            }

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

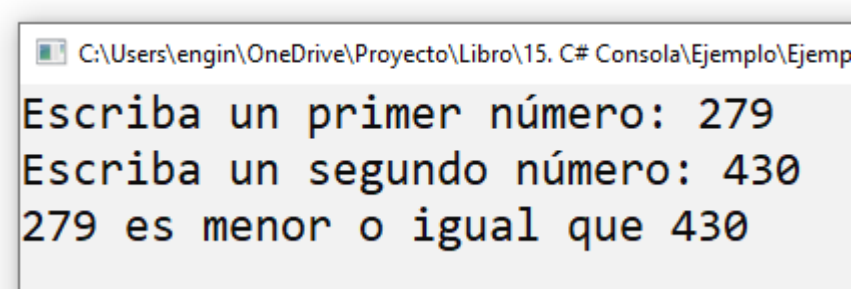


Ilustración 48: Uso del if...else

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Lee dos números por consola
            Console.Write("Escriba un primer número: ");
            double valorA = Convert.ToDouble(Console.ReadLine());

            Console.Write("Escriba un segundo número: ");
            double valorB = Convert.ToDouble(Console.ReadLine());

            //Si condicional
            if (valorA > valorB) {
                Console.WriteLine(valorA.ToString() + " es mayor que " + valorB.ToString());
            }
            else if (valorA < valorB) {
                Console.WriteLine(valorA.ToString() + " es menor que " + valorB.ToString());
            }
            else {
                Console.WriteLine(valorA.ToString() + " es igual a " + valorB.ToString());
            }

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

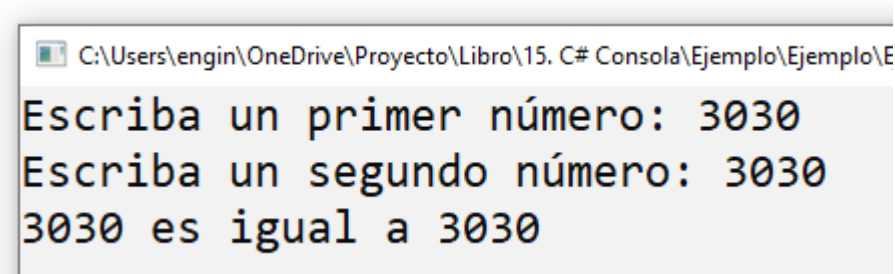


Ilustración 49: if..else if... else

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Lee los lados de un triángulo
            Console.Write("Escriba valor lado A: ");
            double ladoA = Convert.ToDouble(Console.ReadLine());

            Console.Write("Escriba valor lado B: ");
            double ladoB = Convert.ToDouble(Console.ReadLine());

            Console.Write("Escriba valor lado C: ");
            double ladoC = Convert.ToDouble(Console.ReadLine());

            //Si condicional, uso del AND &&
            if (ladoA == ladoB && ladoA == ladoC) {
                Console.WriteLine("Triángulo equilátero");
            }
            else if (ladoA != ladoB && ladoA != ladoC && ladoB != ladoC) {
                Console.WriteLine("Triángulo escaleno");
            }
            else {
                Console.WriteLine("Triángulo isósceles");
            }

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

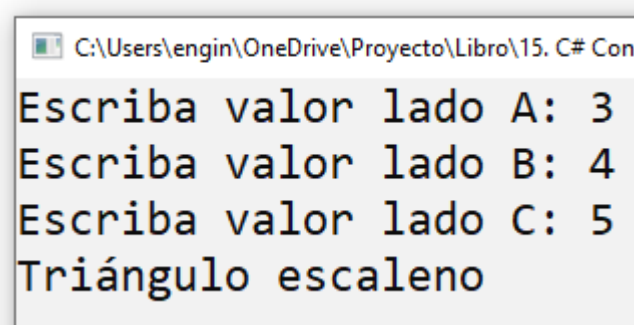


Ilustración 50: Triángulo escaleno

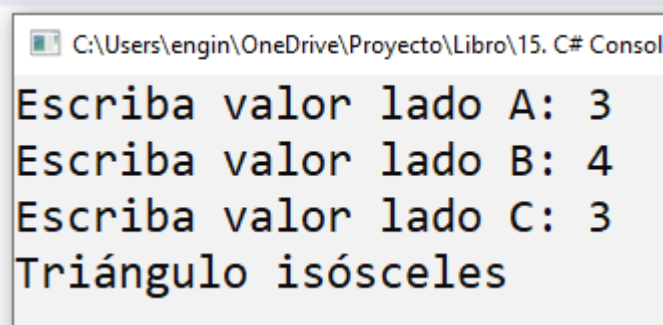


Ilustración 51: Triángulo isósceles

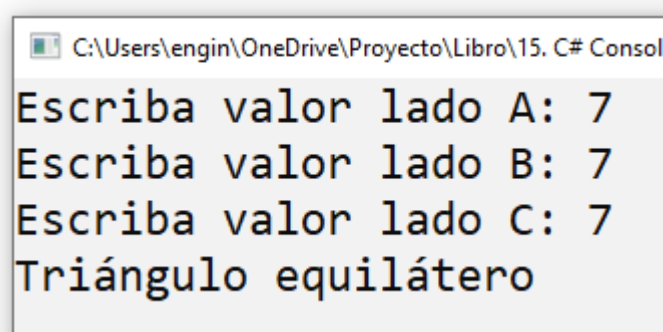


Ilustración 52: Triángulo equilátero

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Lee los lados de un triángulo
            Console.Write("Escriba valor lado A: ");
            double ladoA = Convert.ToDouble(Console.ReadLine());

            Console.Write("Escriba valor lado B: ");
            double ladoB = Convert.ToDouble(Console.ReadLine());

            Console.Write("Escriba valor lado C: ");
            double ladoC = Convert.ToDouble(Console.ReadLine());

            //Si condicional, uso del OR ||
            if (ladoA == ladoB || ladoA == ladoC || ladoB == ladoC) {
                Console.WriteLine("Triángulo equilátero o isósceles");
            }
            else {
                Console.WriteLine("Triángulo escaleno");
            }

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

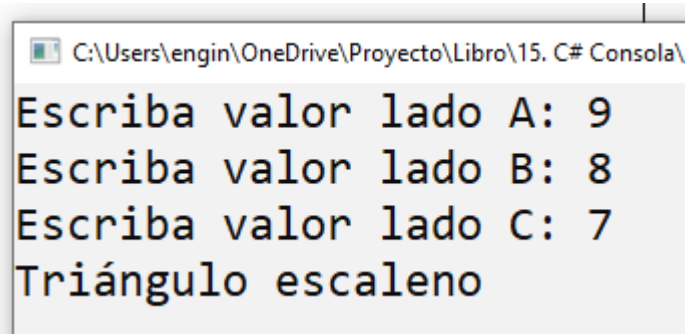


Ilustración 53: Triángulo escaleno

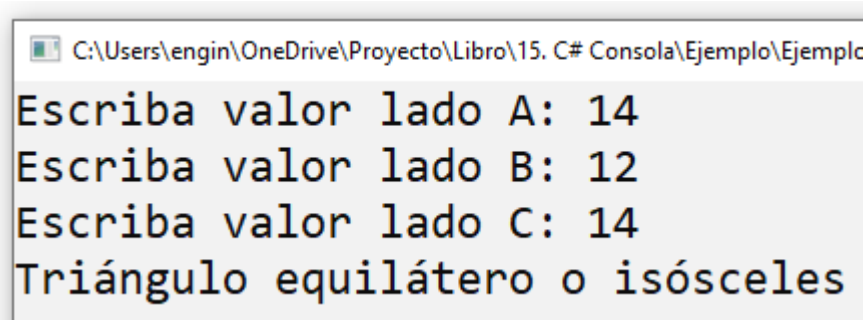


Ilustración 54: Triángulo isósceles



```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Lee un valor entero
            Console.Write("Escriba un valor entero: ");
            int valor = Convert.ToInt32(Console.ReadLine());

            switch (valor) {
                case 1: Console.WriteLine("Escribió uno"); break;
                case 2: Console.WriteLine("Escribió dos, ");
                    Console.WriteLine("que es un número par");
                    break; //Hay que terminar con break
                case 3: //Esto sería el equivalente a un OR
                case 4:
                case 5: Console.WriteLine("Escribió 3 o 4 o 5");
                    break;
                default: Console.WriteLine("Escribió un número por fuera del rango de 1 a 5");
                    break; //Inclusive el default requiere break
            }

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

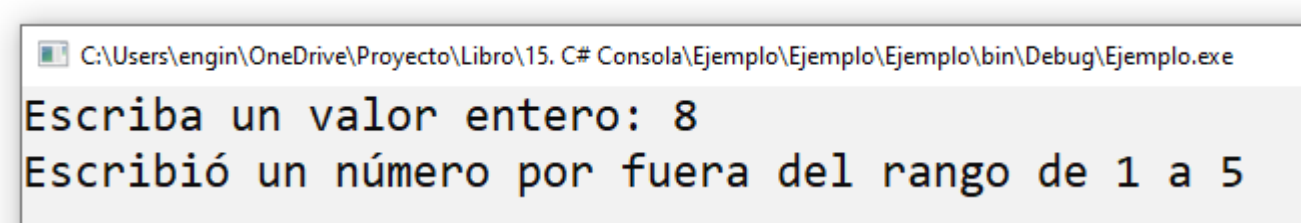


Ilustración 55: Uso del switch

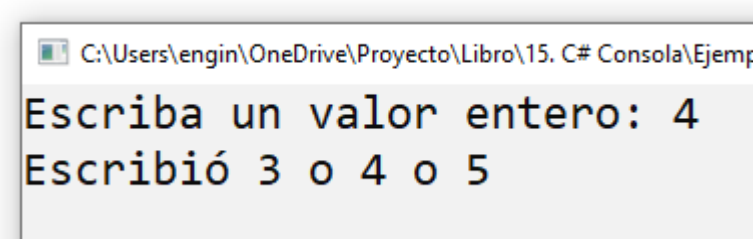


Ilustración 56: Uso del switch

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Operadores booleanos
            bool valA = true;
            bool valB = false;

            bool Resultado1 = valA & valB; //Operador Y
            bool Resultado2 = valA | valB; //Operador O
            bool Resultado3 = valA ^ valB; //Operador XOR
            bool Resultado4 = !valA; //Operador negación

            Console.WriteLine(Resultado1);
            Console.WriteLine(Resultado2);
            Console.WriteLine(Resultado3);
            Console.WriteLine(Resultado4);

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

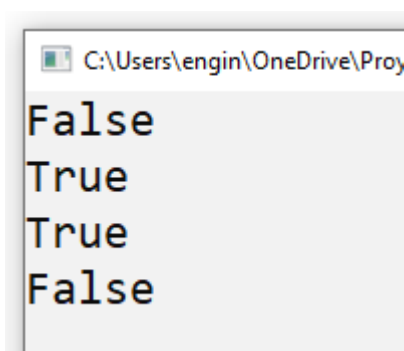


Ilustración 57: Operadores booleanos

```

using System;

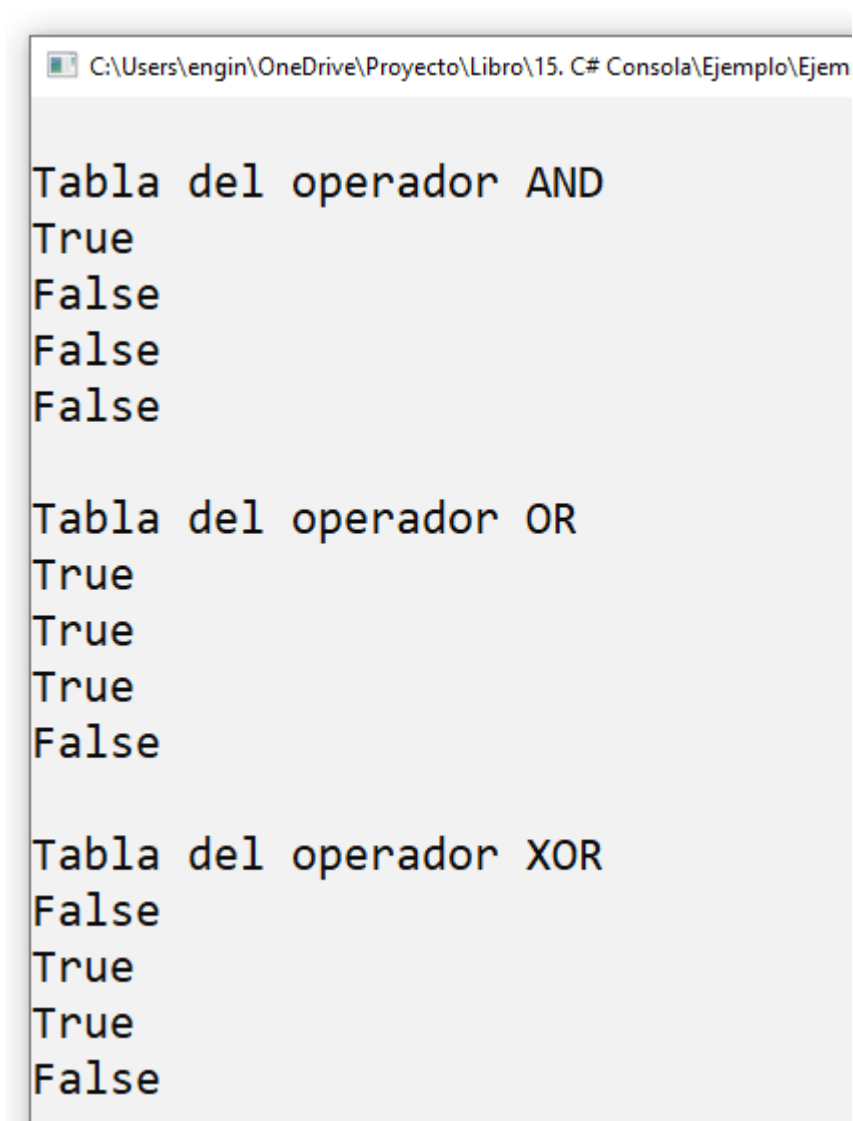
namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Tabla del AND
            Console.WriteLine("\r\nTabla del operador AND");
            Console.WriteLine(true & true);
            Console.WriteLine(true & false);
            Console.WriteLine(false & true);
            Console.WriteLine(false & false);

            //Tabla del OR
            Console.WriteLine("\r\nTabla del operador OR");
            Console.WriteLine(true | true);
            Console.WriteLine(true | false);
            Console.WriteLine(false | true);
            Console.WriteLine(false | false);

            //Tabla del XOR
            Console.WriteLine("\r\nTabla del operador XOR");
            Console.WriteLine(true ^ true);
            Console.WriteLine(true ^ false);
            Console.WriteLine(false ^ true);
            Console.WriteLine(false ^ false);

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}

```



```

C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejem

Tabla del operador AND
True
False
False
False

Tabla del operador OR
True
True
True
False

Tabla del operador XOR
False
True
True
False

```

Ilustración 58: Operadores booleanos

```

using System;

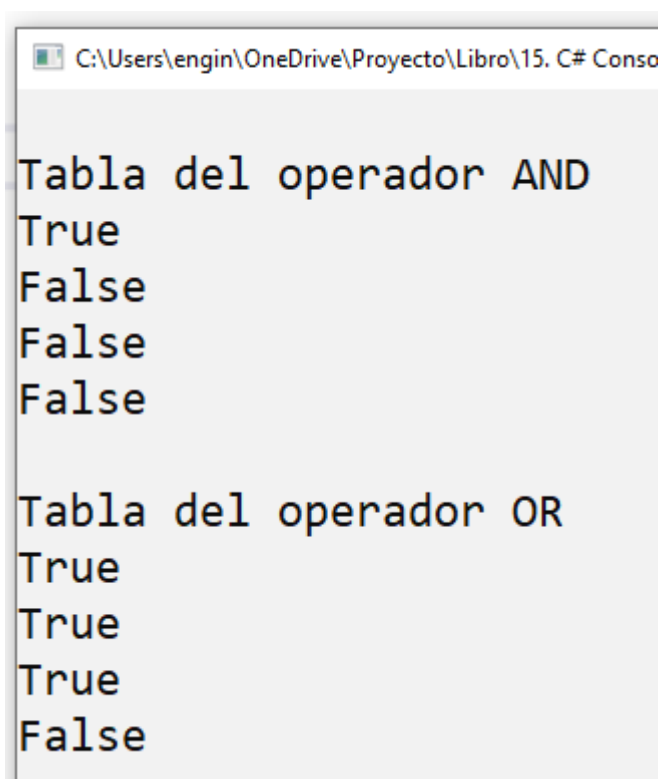
namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Tabla del AND
            Console.WriteLine("\r\nTabla del operador AND");
            Console.WriteLine(true && true);
            Console.WriteLine(true && false);
            Console.WriteLine(false && true);
            Console.WriteLine(false && false);

            //Tabla del OR
            Console.WriteLine("\r\nTabla del operador OR");
            Console.WriteLine(true || true);
            Console.WriteLine(true || false);
            Console.WriteLine(false || true);
            Console.WriteLine(false || false);

            /* La diferencia entre & y &&, | y || es que cuando se
            * usa & se evalúan ambos operandos a la izquierda y derecha del &,
            * en cambio cuando se usa && se evalúa primero el operando de la izquierda
            * y si da falso, ya se sabe que toda la expresión es falsa. Luego en
            * un si condicional, se ejecuta más rápido si se usa && en vez de &.
            * Similar se aplica para || o | , si se usa en un si condicional
            * evalúa el primer operando, si es verdadero, toda la expresión es verdadera. */

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}

```



C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Conso

```

Tabla del operador AND
True
False
False
False

Tabla del operador OR
True
True
True
False

```

Ilustración 59: Operadores booleanos

```

using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            Console.WriteLine("\r\nCompleja expresión booleana");
            Console.WriteLine(true && true || false && true || true || false);
            Console.WriteLine(true || false && true || false && true && true);
            Console.WriteLine(!false && !true || !true && false && !false);

            /* No se recomienda en una expresión lógica poner dos o más operadores
            * lógicos. Llega a confundir.
            * Hacer uso de paréntesis. */

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}

```

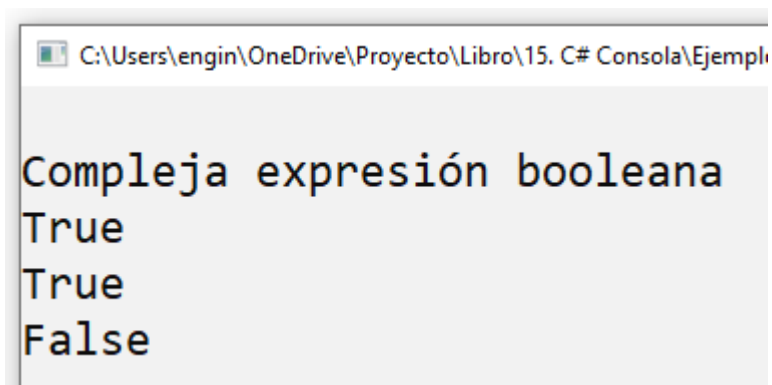


Ilustración 60: Compleja expresión booleana

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            Console.WriteLine("\r\nCompleja expresión booleana");
            Console.WriteLine(!true && true | !false & true ^ true & !false);
            Console.WriteLine(!true | false & true ^ false && !true & true);
            Console.WriteLine(!false | !true ^ !true & false & !false);

            /* Precedencia de los operadores. Tomado de: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/boolean-logical-operators
                Logical negation operator !
                Logical AND operator &
                Logical exclusive OR operator ^
                Logical OR operator |
                Conditional logical AND operator &&
                Conditional logical OR operator ||
            */

            //Esta instrucción detiene que se cierre la ventana de consola
            Console.ReadKey();
        }
    }
}
```

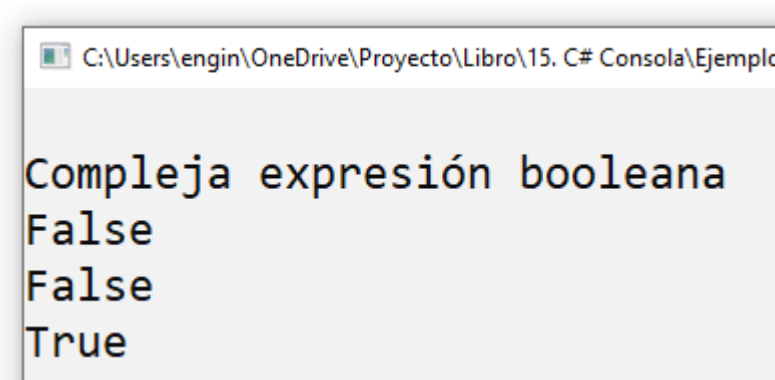


Ilustración 61: Precedencia de los operadores

# El operador “?”, un si condicional

Usado en una asignación, funciona de esta manera:

Variable = condición ? valor si es verdadero: valor si es falso;

040.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Operador ?
            int valA = 10;
            int valB = 13;
            string resultado = valA > valB ? "A es mayor": "B es mayor o igual";
            Console.WriteLine(resultado);

            //Segundo ejemplo
            int valC = 10;
            int valD = 13;
            int valE = 34;
            string imprimir = valE > valD && valC <= valD ? "Primero" : "de lo contrario, Segundo";
            Console.WriteLine(imprimir);

            Console.ReadKey();
        }
    }
}
```

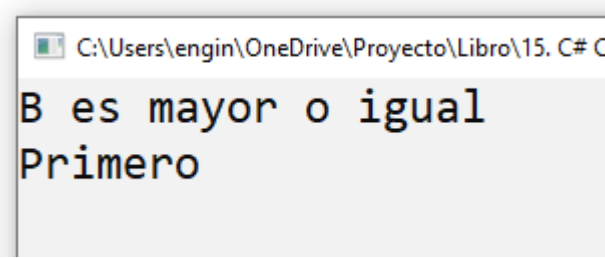


Ilustración 62: Operador ?

# Captura de error con try...catch

Se captura el error y así se evita que el programa colapse.

041.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Uso del try...catch
            int valA = 17;
            int valB = 0;
            int resultado;
            try {
                resultado = valA / valB; //Intenta dividir entre cero
                Console.WriteLine("Resultado es: " + resultado.ToString());
            }
            catch { //Captura el error
                Console.WriteLine("División entre cero");
            }
            Console.ReadKey();
        }
    }
}
```

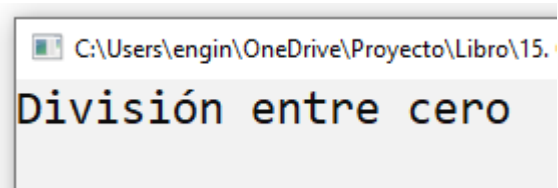


Ilustración 63: Uso de try...catch



```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Uso del TryParse. Trata de convertir un string a entero
            string Numero = "150";
            int valorEntero;
            if (Int32.TryParse(Numero, out valorEntero)) {
                Console.WriteLine("Se pudo convertir. valorEntero: " + valorEntero.ToString());
            }
            else {
                Console.WriteLine("1. No se puede convertir a entero");
            }

            //Segundo ejemplo
            Numero = "4.178"; //Un punto en la cadena
            if (Int32.TryParse(Numero, out valorEntero)) {
                Console.WriteLine("Se pudo convertir. valorEntero: " + valorEntero.ToString());
            }
            else {
                Console.WriteLine("2. No se puede convertir a entero");
            }

            //Tercer ejemplo
            Numero = "    90    "; //espacios en la cadena
            if (Int32.TryParse(Numero, out valorEntero)) {
                Console.WriteLine("Se pudo convertir. valorEntero: " + valorEntero.ToString());
            }
            else {
                Console.WriteLine("3. No se puede convertir a entero");
            }

            //Cuarto ejemplo
            Numero = "-90";
            if (Int32.TryParse(Numero, out valorEntero)) {
                Console.WriteLine("Se pudo convertir. valorEntero: " + valorEntero.ToString());
            }
            else {
                Console.WriteLine("4. No se puede convertir a entero");
            }

            //Quinto ejemplo
            Numero = "- 90"; //Espacio intermedio
            if (Int32.TryParse(Numero, out valorEntero)) {
                Console.WriteLine("Se pudo convertir. valorEntero: " + valorEntero.ToString());
            }
            else {
                Console.WriteLine("5. No se puede convertir a entero");
            }

            Console.ReadKey();
        }
    }
}
```

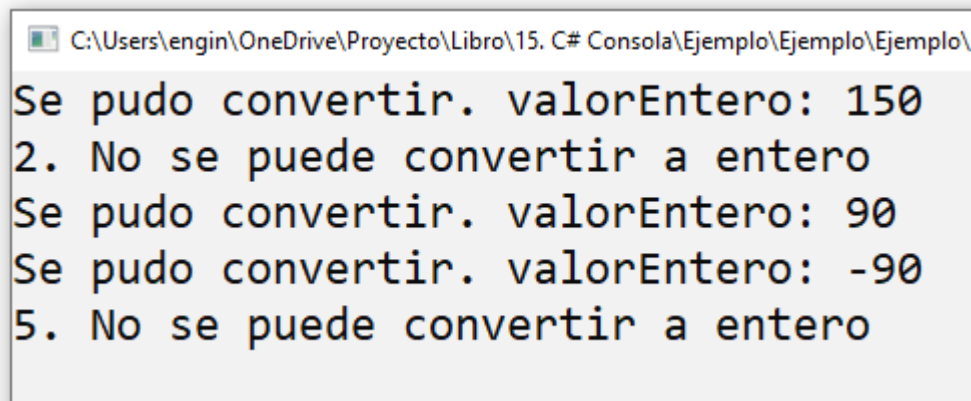


Ilustración 64: Uso de TryParse

## Ciclo for

Su sintaxis es:

```
for (variable=valor inicio; condición; incremento/decremento variable) {  
  
}
```

043.cs

```
using System;  
  
namespace Ejemplo {  
    class Program {  
        static void Main(string[] args) {  
            //Ciclo for ascendente  
            Console.WriteLine("Ciclo ascendente:");  
            for (int cont=1; cont<=20; cont++) {  
                Console.Write(cont.ToString() + ", ");  
            }  
  
            //Ciclo for descendente  
            Console.WriteLine("\r\nCiclo descendente:");  
            for (int cont = 20; cont >= 1; cont--) {  
                Console.Write(cont.ToString() + ", ");  
            }  
  
            //Ciclo for ascendente, avance de 2 en 2  
            Console.WriteLine("\r\nCiclo ascendente (avance de 2 en 2):");  
            for (int cont = 1; cont <= 20; cont+=2) {  
                Console.Write(cont.ToString() + ", ");  
            }  
  
            //Ciclo for descendente, retrocede de 2 en 2  
            Console.WriteLine("\r\nCiclo descendente (retrocede de 2 en 2):");  
            for (int cont = 20; cont >= 1; cont -= 2) {  
                Console.Write(cont.ToString() + ", ");  
            }  
  
            //Ciclo for ascendente, avance doble  
            Console.WriteLine("\r\nCiclo ascendente (avance doble):");  
            for (int cont = 1; cont <= 20; cont *= 2) {  
                Console.Write(cont.ToString() + ", ");  
            }  
  
            //Ciclo for descendente, retrocede la mitad  
            Console.WriteLine("\r\nCiclo descendente (retrocede la mitad):");  
            for (int cont = 20; cont >= 1; cont /= 2) {  
                Console.Write(cont.ToString() + ", ");  
            }  
  
            Console.ReadKey();  
        }  
    }  
}
```

```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\bin\Debug\Ejemplo.exe
Ciclo ascendente:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
Ciclo descendente:
20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
Ciclo ascendente (avance de 2 en 2):
1, 3, 5, 7, 9, 11, 13, 15, 17, 19,
Ciclo descendente (retrocede de 2 en 2):
20, 18, 16, 14, 12, 10, 8, 6, 4, 2,
Ciclo ascendente (avance doble):
1, 2, 4, 8, 16,
Ciclo descendente (retrocede la mitad):
20, 10, 5, 2, 1,
```

Ilustración 65: Ciclo for

# Ciclo while

Su sintaxis es:

```
while (condición) {  
  
}
```

044.cs

```
using System;  
  
namespace Ejemplo {  
    class Program {  
        static void Main(string[] args) {  
            int cont;  
  
            //Ciclo while ascendente  
            Console.WriteLine("Ciclo ascendente:");  
            cont = 1;  
            while(cont<=20) {  
                Console.Write(cont.ToString() + ", ");  
                cont++;  
            }  
  
            //Ciclo while descendente  
            Console.WriteLine("\r\nCiclo descendente:");  
            cont = 20;  
            while (cont >= 1) {  
                Console.Write(cont.ToString() + ", ");  
                cont--;  
            }  
  
            //Ciclo while ascendente, avance de 2 en 2  
            Console.WriteLine("\r\nCiclo ascendente (avance de 2 en 2):");  
            cont = 1;  
            while (cont <= 20) {  
                Console.Write(cont.ToString() + ", ");  
                cont += 2;  
            }  
  
            //Ciclo while descendente, retrocede de 2 en 2  
            Console.WriteLine("\r\nCiclo descendente (retrocede de 2 en 2):");  
            cont = 20;  
            while (cont >= 1) {  
                Console.Write(cont.ToString() + ", ");  
                cont -= 2;  
            }  
  
            //Ciclo while ascendente, avance doble  
            Console.WriteLine("\r\nCiclo ascendente (avance doble):");  
            cont = 1;  
            while (cont <= 20) {  
                Console.Write(cont.ToString() + ", ");  
                cont *= 2;  
            }  
  
            //Ciclo while descendente, retrocede la mitad  
            Console.WriteLine("\r\nCiclo descendente (retrocede la mitad):");  
            cont = 20;  
            while (cont >= 1) {  
                Console.Write(cont.ToString() + ", ");  
                cont /= 2;  
            }  
  
            Console.ReadKey();  
        }  
    }  
}
```

```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\bin\Debug\Ejemplo.exe
Ciclo ascendente:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
Ciclo descendente:
20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,
Ciclo ascendente (avance de 2 en 2):
1, 3, 5, 7, 9, 11, 13, 15, 17, 19,
Ciclo descendente (retrocede de 2 en 2):
20, 18, 16, 14, 12, 10, 8, 6, 4, 2,
Ciclo ascendente (avance doble):
1, 2, 4, 8, 16,
Ciclo descendente (retrocede la mitad):
20, 10, 5, 2, 1,
```

Ilustración 66: Ciclo while

## Ciclo do...while

Su sintaxis es:

```
do {  
  
  
} while (condición)
```

045.cs

```
using System;  
  
namespace Ejemplo {  
    class Program {  
        static void Main(string[] args) {  
            int cont;  
  
            //Ciclo do-while ascendente  
            Console.WriteLine("Ciclo ascendente:");  
            cont = 1;  
            do {  
                Console.Write(cont.ToString() + ", ");  
                cont++;  
            } while (cont <= 20);  
  
            //Ciclo do-while descendente  
            Console.WriteLine("\r\nCiclo descendente:");  
            cont = 20;  
            do {  
                Console.Write(cont.ToString() + ", ");  
                cont--;  
            } while (cont >= 1);  
  
            //Ciclo do-while ascendente, avance de 2 en 2  
            Console.WriteLine("\r\nCiclo ascendente (avance de 2 en 2):");  
            cont = 1;  
            do {  
                Console.Write(cont.ToString() + ", ");  
                cont += 2;  
            } while (cont <= 20);  
  
            //Ciclo do-while descendente, retrocede de 2 en 2  
            Console.WriteLine("\r\nCiclo descendente (retrocede de 2 en 2):");  
            cont = 20;  
            do {  
                Console.Write(cont.ToString() + ", ");  
                cont -= 2;  
            } while (cont >= 1);  
  
            //Ciclo do-while ascendente, avance doble  
            Console.WriteLine("\r\nCiclo ascendente (avance doble):");  
            cont = 1;  
            do {  
                Console.Write(cont.ToString() + ", ");  
                cont *= 2;  
            } while (cont <= 20);  
  
            //Ciclo do-while descendente, retrocede la mitad  
            Console.WriteLine("\r\nCiclo descendente (retrocede la mitad):");  
            cont = 20;  
            do {  
                Console.Write(cont.ToString() + ", ");  
                cont /= 2;  
            } while (cont >= 1);  
  
            Console.ReadKey();  
        }  
    }  
}
```

```
Ciclo ascendente:  
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,  
Ciclo descendente:  
20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1,  
Ciclo ascendente (avance de 2 en 2):  
1, 3, 5, 7, 9, 11, 13, 15, 17, 19,  
Ciclo descendente (retrocede de 2 en 2):  
20, 18, 16, 14, 12, 10, 8, 6, 4, 2,  
Ciclo ascendente (avance doble):  
1, 2, 4, 8, 16,  
Ciclo descendente (retrocede la mitad):  
20, 10, 5, 2, 1,
```

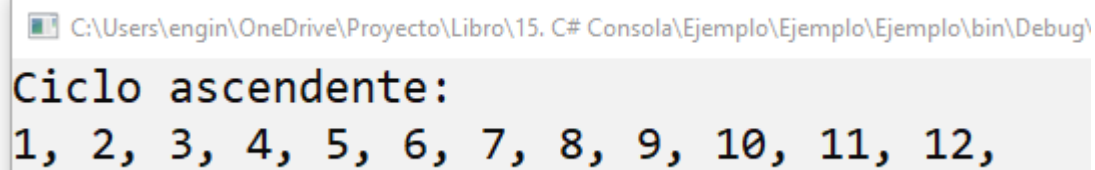
Ilustración 67: Ciclo do...while

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            int cont;

            //Rompe el ciclo con break
            Console.WriteLine("Ciclo ascendente:");
            cont = 1;
            do {
                //Si cont es múltiplo de 13 se rompe el ciclo con break
                if (cont % 13 == 0) break;
                Console.Write(cont.ToString() + ", ");
                cont++;
            } while (cont <= 20);

            Console.ReadKey();
        }
    }
}
```



Ciclo ascendente:  
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,

Ilustración 68: Romper ciclo con break



```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            int cont;

            Console.WriteLine("Ciclo ascendente:");
            cont = 0;
            do {
                cont++;

                //Si cont es par entonces va a la siguiente iteración, no
                //ejecuta lo que está después.
                if (cont % 2 == 0) continue;

                Console.Write(cont.ToString() + ", ");
            } while (cont <= 20);

            Console.ReadKey();
        }
    }
}
```

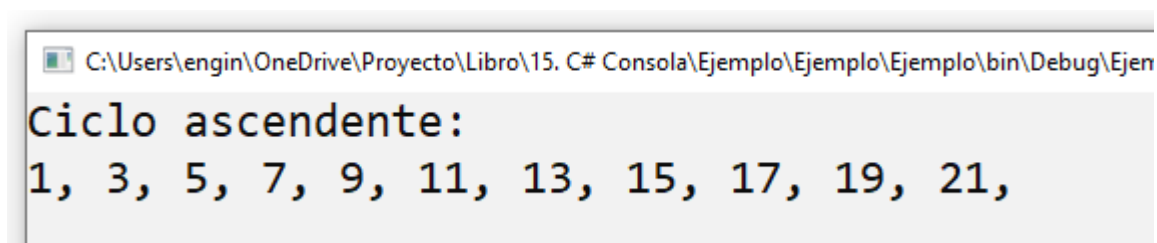


Ilustración 69: Uso del continue

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {

            //Ciclos anidados simulando un minuterero y un segundero
            for (int minuto = 0; minuto <= 10; minuto++) {
                for (int segundo = 0; segundo < 60; segundo++) {
                    Console.WriteLine(minuto.ToString() + ":" + segundo.ToString());
                }
            }

            Console.ReadKey();
        }
    }
}
```

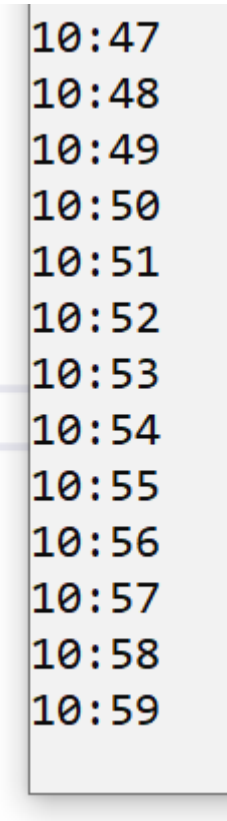


Ilustración 70: Ciclos anidados

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {

            for (int numA = 1; numA <= 3; numA++) {
                for (int numB = 1; numB <= 6; numB++) {
                    for (int numC = 1; numC <= 9; numC++) {
                        Console.WriteLine(numA.ToString() + "|" + numB.ToString() + "|" +
numC.ToString());
                        if (numC == 5) break; //Sólo rompe el ciclo más interno
                    }
                }
            }

            Console.WriteLine("Final");
            Console.ReadKey();
        }
    }
}
```

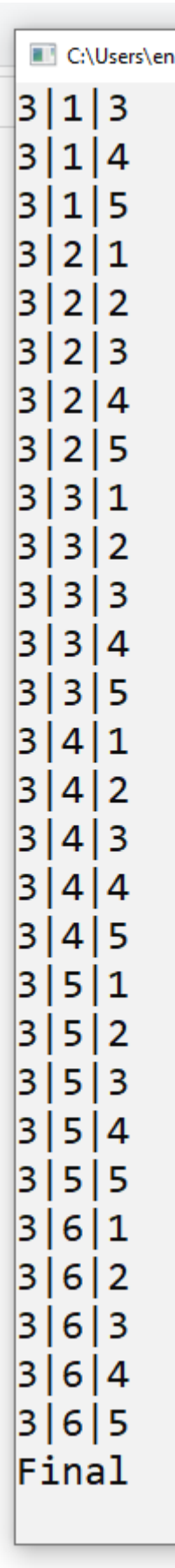


Ilustración 71: Break solo rompe un ciclo

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {

            for (int numA = 1; numA <= 3; numA++) {
                for (int numB = 1; numB <= 6; numB++) {
                    for (int numC = 1; numC <= 9; numC++) {
                        Console.WriteLine(numA.ToString() + "|" + numB.ToString() + "|" +
numC.ToString());
                        if (numC == 5) goto afuera; //Sale de todos los ciclos
                    }
                }
            }

            afuera: Console.WriteLine("Final");
            Console.ReadKey();
        }
    }
}
```

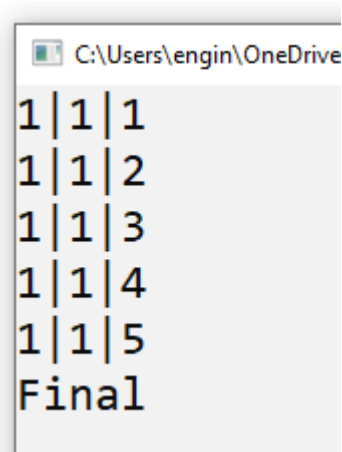


Ilustración 72: Uso del goto para romper todos los ciclos anidados

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Diferencias en precisión entre float y double

            float acumA = 0;
            float divideA = 7;
            for (float numA = 1; numA <= 100000; numA++) {
                acumA += 1 / divideA;
            }

            double acumB = 0;
            double divideB = 7;
            for (double numB = 1; numB <= 100000; numB++) {
                acumB += 1 / divideB;
            }

            Console.WriteLine("Resultados: " + acumA.ToString() + " y " + acumB.ToString());
            Console.ReadKey();
        }
    }
}
```

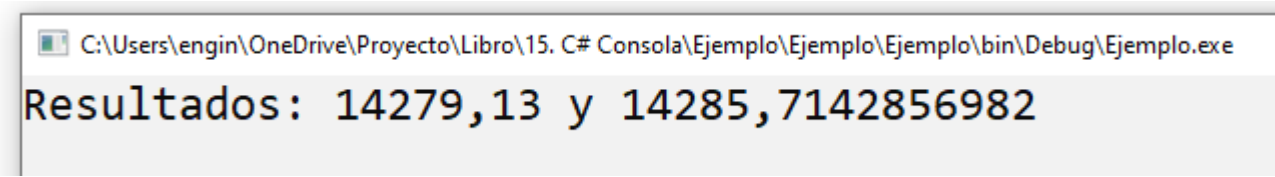


Ilustración 73: Diferencia de precisión entre single y double

Muy recomendado: usar double.

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Generando números al azar.
            Random azar = new Random(); //La semilla del generador la define el .Net framework

            //Números enteros al azar
            Console.Write("Enteros positivos: ");
            for (int cont=1; cont<=10; cont++) {
                int numEntero = azar.Next();
                Console.Write(numEntero.ToString() + " | ");
            }

            //Números entre 0 y 1 al azar
            Console.Write("\r\n\r\nReales: ");
            for (int cont = 1; cont <= 10; cont++) {
                double numReal = azar.NextDouble();
                Console.Write(numReal.ToString() + " | ");
            }

            //Números entre 15 y 44 al azar.
            Console.Write("\r\n\r\nEnteros positivos entre 15 y 44: ");
            for (int cont = 1; cont <= 10; cont++) {
                int numEntero = azar.Next(15, 45); //El segundo parámetro debe ser +1 del rango máximo que
se busca
                Console.Write(numEntero.ToString() + " | ");
            }

            //Generando los mismos valores
            Console.Write("\r\n\r\nGenerando los mismos valores: ");
            Random AleatorioA = new Random(500);
            Random AleatorioB = new Random(500);
            for (int cont=1; cont<=20; cont++) {
                int numA = AleatorioA.Next(55, 95);
                int numB = AleatorioB.Next(55, 95);
                Console.Write(numA.ToString() + " y " + numB.ToString() + " | ");
            }

            Console.ReadKey();
        }
    }
}
```

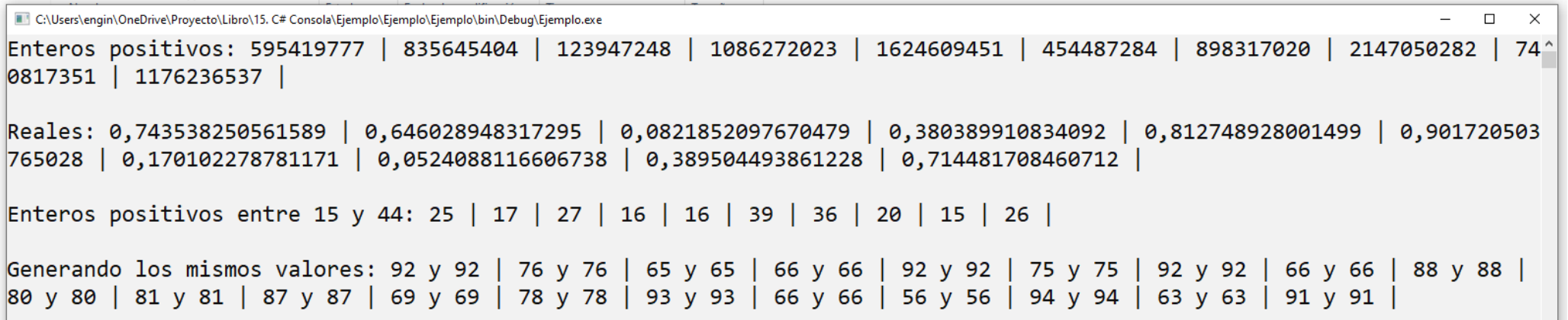


Ilustración 74: Generando números al azar

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Uso de funciones
            double ladoA = 3;
            double ladoB = 4;
            double ladoC = 5;
            double Area = AreaTrianguloHeron(ladoA, ladoB, ladoC);
            Console.WriteLine("Area triángulo es: " + Area.ToString());

            Console.ReadKey();
        }

        //Fórmula de Herón para el cálculo del área de un triángulo dado los lados
        static double AreaTrianguloHeron(double valA, double valB, double valC) {
            double s = (valA + valB + valC) / 2;
            double area = Math.Sqrt(s * (s - valA) * (s - valB) * (s - valC));
            return area;
        }
    }
}
```

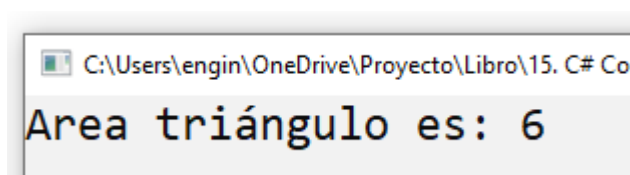


Ilustración 75: Funciones

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Función con doble salida. Nuevo en C# 7.0
            double radio = 1;
            double perimetro, area;
            area = DatosCirculo(radio, out perimetro);
            Console.WriteLine("Area del círculo es: " + area.ToString());
            Console.WriteLine("Perímetro del círculo es: " + perimetro.ToString());
            Console.ReadKey();
        }

        //Retorna dos valores: el área y el perímetro del círculo
        static double DatosCirculo(double Radio, out double Perimetro) {
            double area = Math.PI * Radio * Radio;
            Perimetro = 2 * Math.PI * Radio;
            return area;
        }
    }
}
```

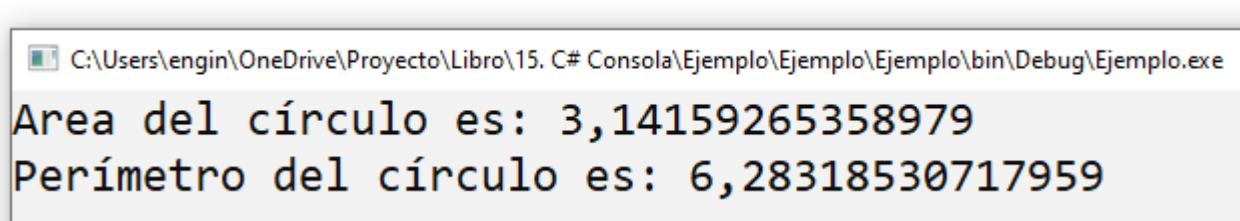


Ilustración 76: Función con doble salida

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Función con doble salida. Nuevo en C# 7.0
            double radio = 1;
            double perimetro, area;
            DatosCirculo(radio, out perimetro, out area);
            Console.WriteLine("Area del círculo es: " + area.ToString());
            Console.WriteLine("Perímetro del círculo es: " + perimetro.ToString());
            Console.ReadKey();
        }

        //Retorna dos valores: el área y el perímetro del círculo
        static void DatosCirculo(double Radio, out double Perimetro, out double Area) {
            Area = Math.PI * Radio * Radio;
            Perimetro = 2 * Math.PI * Radio;
        }
    }
}
```

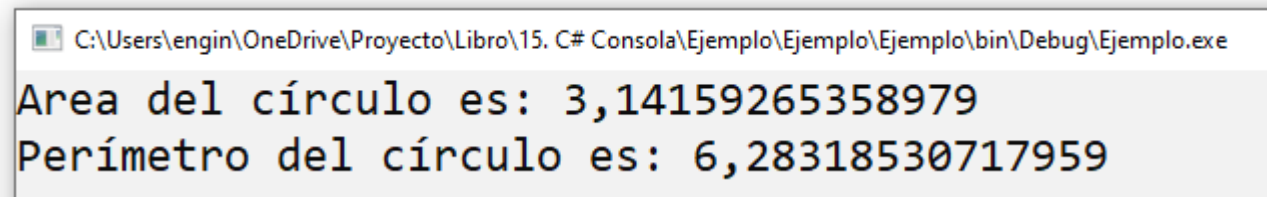


Ilustración 77: Función con triple salida



```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Función iterativa y recursiva
            long valor = 10;
            long factorialA, factorialB;

            factorialA = CalculaFactorialIterativa(valor);
            factorialB = CalculaFactorialRecursivo(valor);

            Console.WriteLine("factorialA es: " + factorialA.ToString());
            Console.WriteLine("factorialB es: " + factorialB.ToString());
            Console.ReadKey();
        }

        //Retorna el factorial de un número, de forma iterativa
        static long CalculaFactorialIterativa(long numero) {
            long resultado = 1;
            for (long num=2; num <= numero; num++) {
                resultado *= num;
            }
            return resultado;
        }

        //Retorna el factorial de un número, de forma recursiva
        static long CalculaFactorialRecursivo(long numero) {
            if (numero == 1) return 1;
            return numero*CalculaFactorialRecursivo(numero-1);
        }
    }
}
```

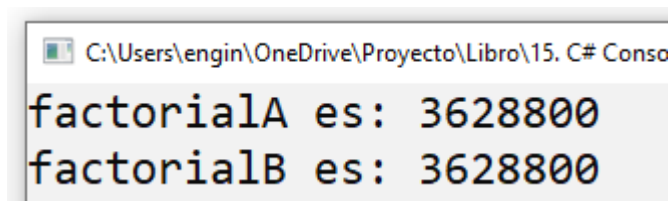


Ilustración 78: Función iterativa y recursiva

```
using System;

namespace Ejemplo {
    class Program {
        static int valor; //Una variable que es conocida por todas las funciones

        static void Main(string[] args) {
            //Ámbito de las variables
            valor = 17;
            Console.WriteLine("Valor es: " + valor.ToString());
            UnProcedimiento();
            Console.WriteLine("Valor es: " + valor.ToString());
            OtroProcedimiento();
            Console.WriteLine("Valor es: " + valor.ToString());
            Console.ReadKey();
        }

        //Un procedimiento
        static void UnProcedimiento() {
            valor = 853;
        }

        //Otro procedimiento
        static void OtroProcedimiento() {
            valor = 987654321;
        }
    }
}
```

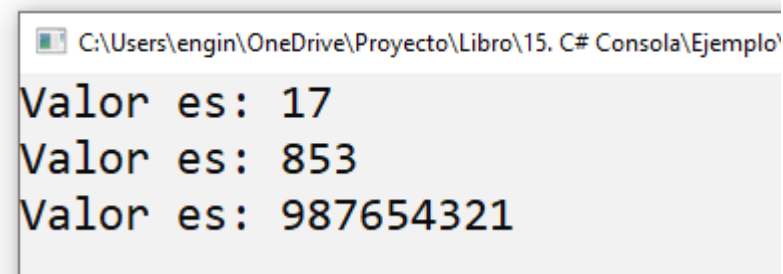


Ilustración 79: Variable global

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Ámbito de las variables. Variable local
            int valor = 17;
            Console.WriteLine("Valor es: " + valor.ToString());
            UnProcedimiento();
            Console.WriteLine("Valor es: " + valor.ToString());
            OtroProcedimiento();
            Console.WriteLine("Valor es: " + valor.ToString());
            Console.ReadKey();
        }

        //Un procedimiento. La variable "valor" es otra distinta a la del "Main"
        static void UnProcedimiento() {
            int valor = 853;
        }

        //Otro procedimiento. La variable "valor" es otra distinta a la del "Main"
        static void OtroProcedimiento() {
            int valor = 987654321;
        }
    }
}
```

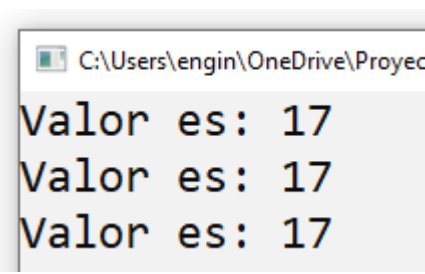


Ilustración 80: Variable local

```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Valores al azar PERO que dan la misma sucesión
            ProcedimientoA();
            Console.WriteLine(" ");
            ProcedimientoB();
            Console.ReadKey();
        }

        static void ProcedimientoA() {
            Random azar = new Random();
            for (int num=1; num<=10; num++) {
                Console.Write(azar.Next(1, 30).ToString() + " , ");
            }
        }

        static void ProcedimientoB() {
            Random azar = new Random();
            for (int num = 1; num <= 10; num++) {
                Console.Write(azar.Next(1, 30).ToString() + " , ");
            }
        }
    }
}
```

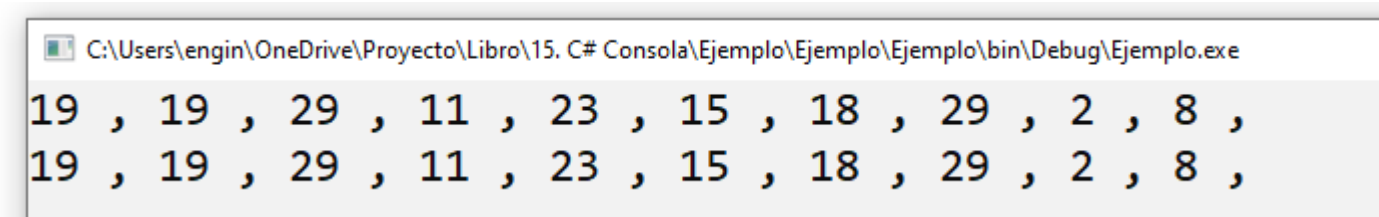


Ilustración 81: Dos sucesiones iguales de números aleatorios

La semilla usada por el algoritmo generador de números aleatorios .NET Framework es el reloj de la máquina. Cuando se crea el generador de números aleatorios se inicializa con el reloj de la máquina y como ese reloj no ha variado entre el llamado de una función y otra, entonces se generan los mismos valores porque la semilla es igual.

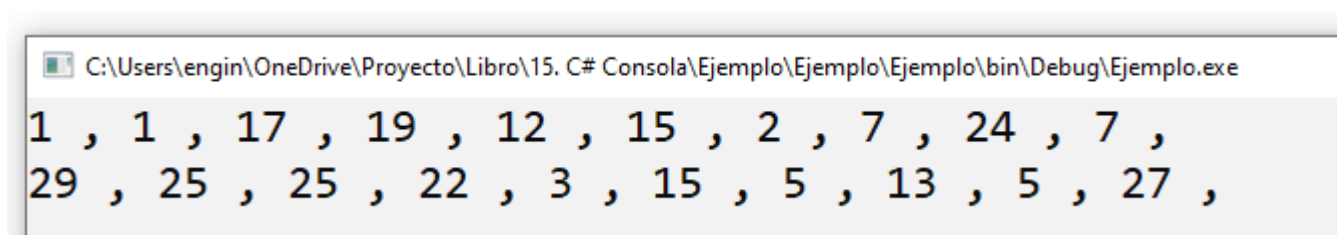
```
using System;

namespace Ejemplo {
    class Program {
        static void Main(string[] args) {
            //Enviando el objeto aleatorio como parámetro
            Random azar = new Random();

            ProcedimientoA(azar);
            Console.WriteLine(" ");
            ProcedimientoB(azar);
            Console.ReadKey();
        }

        static void ProcedimientoA(Random azar) {
            for (int num=1; num<=10; num++) {
                Console.Write(azar.Next(1, 30).ToString() + " , ");
            }
        }

        static void ProcedimientoB(Random azar) {
            for (int num = 1; num <= 10; num++) {
                Console.Write(azar.Next(1, 30).ToString() + " , ");
            }
        }
    }
}
```



```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\bin\Debug\Ejemplo.exe  
1 , 1 , 17 , 19 , 12 , 15 , 2 , 7 , 24 , 7 ,  
29 , 25 , 25 , 22 , 3 , 15 , 5 , 13 , 5 , 27 ,
```

*Ilustración 82: Dos secuencias distintas de números aleatorios*

Como sólo hay un solo inicio del generador de números aleatorios, entonces se garantiza que la secuencia de números sea distinta.