

# C# Y .NET 8

## Parte 01. Variables, si condicional, ciclos, funciones

2024-07

Rafael Alberto Moreno Parra  
ramsoftware@gmail.com

# Contenido

Tabla de ilustraciones.....	4
Acerca del autor.....	6
Licencia de este libro .....	6
Licencia del software .....	6
Marcas registradas .....	7
Dedicatoria .....	8
Comentarios en el código .....	9
El tradicional "Hola Mundo" .....	10
Variables .....	12
Imprimiendo con formato .....	18
Operaciones matemáticas.....	23
Diferencias de precisión entre float, double y decimal.....	25
Función de potencia.....	26
Orden de evaluación de los operadores.....	28
Problemas al usar cast .....	29
De formato algebraico a C# .....	31
Diferencias entre usar el cast y la conversión.....	33
Conversión de números reales con el punto decimal .....	34
Fallos en la conversión .....	35
Constantes matemáticas .....	36
Funciones trigonométricas .....	38
Funciones hiperbólicas .....	40
Otras funciones matemáticas .....	41
Manejo del NaN (Not a Number) y el infinito.....	43
Leer un número por consola.....	45
Si condicional .....	46
Uso del if...else if... else.....	48
Uso del AND &&.....	50
Uso del OR    .....	52
Uso del switch .....	53
Operadores booleanos .....	54
Precedencia de los operadores.....	60
El operador "?", un si condicional .....	61

Captura de error con try...catch.....	62
Uso de TryParse para conversión.....	63
Ciclo for.....	65
Ciclo while .....	67
Ciclo do...while .....	69
Romper ciclo con break .....	71
Uso del continue en ciclos.....	72
Ciclos anidados .....	73
Rompiendo ciclos anidados .....	74
Uso del goto.....	75
Diferencias de precisión entre float y double .....	76
Funciones .....	77
Funciones con doble salida .....	79
Funciones iterativas y recursivas.....	81
Ámbito de las variables .....	83
Manejo de cifras.....	85

## Tabla de ilustraciones

Ilustración 1: Ejecución del programa "Hola Mundo". Impresión en consola.....	10
Ilustración 2: Diferencia entre Write y Writeln.....	11
Ilustración 3: Variables de tipo entero .....	13
Ilustración 4: Valores de tipo "double" .....	14
Ilustración 5: Valores de tipo "char" .....	15
Ilustración 6: Valores de tipo "string" .....	16
Ilustración 7: Valores almacenados por variables de tipo "float" .....	17
Ilustración 8: Números impresos con determinados formatos .....	19
Ilustración 9: Impresión de números de tipo double con formato .....	20
Ilustración 10: Valores tipo "double" con formato .....	21
Ilustración 11: Valores tipo "double" con formato de redondeo .....	22
Ilustración 12: Resultados de operaciones .....	24
Ilustración 13: Diferencias entre float, double y decimal .....	25
Ilustración 14: Función de potencia .....	26
Ilustración 15: Orden de evaluación de los operadores .....	28
Ilustración 16: Problemas del uso del "cast" .....	29
Ilustración 17: Problemas del uso del "cast" .....	30
Ilustración 18: Formato algebraico a C# .....	32
Ilustración 19: Diferencias entre usar el cast y la conversión .....	33
Ilustración 20: Conversión de números reales con el punto decimal.....	34
Ilustración 21: Mensaje de error .....	35
Ilustración 22: Constantes matemáticas .....	37
Ilustración 23: Funciones trigonométricas .....	39
Ilustración 24: Funciones hiperbólicas .....	40
Ilustración 25: Otras funciones matemáticas .....	42
Ilustración 26: Manejo del NaN (Not a Number) y el infinito .....	44
Ilustración 27: Leer un número por consola.....	45
Ilustración 28: Si condicional .....	47
Ilustración 29: Uso de if...else .....	48
Ilustración 30: Uso de if... else if ... else .....	49
Ilustración 31: Uso del AND && .....	51
Ilustración 32: Uso del OR   .....	52
Ilustración 33: Uso del switch .....	53
Ilustración 34: Operadores booleanos.....	54
Ilustración 35: Tablas de verdad .....	56
Ilustración 36: Tablas de verdad usando && y    .....	58
Ilustración 37: Expresión booleana .....	59
Ilustración 38: Precedencia de los operadores.....	60
Ilustración 39: El operador "?", un si condicional .....	61
Ilustración 40: Captura de error con try...catch.....	62
Ilustración 41: Uso de TryParse para conversión.....	64
Ilustración 42: Ciclo for.....	66
Ilustración 43: Ciclo while.....	68
Ilustración 44: Ciclo do...while .....	70
Ilustración 45: Romper ciclo con break .....	71

Ilustración 46: Uso del continue en ciclos .....	72
Ilustración 47: Ciclos anidados .....	73
Ilustración 48: Rompiendo ciclos anidados .....	74
Ilustración 49: Uso del goto .....	75
Ilustración 50: Diferencias de precisión entre float y double .....	76
Ilustración 51: Función que retorna dato de tipo bool.....	77
Ilustración 52: Funciones .....	78
Ilustración 53: Funciones con doble salida.....	79
Ilustración 54: Funciones con doble salida.....	80
Ilustración 55: Funciones iterativas y recursivas.....	82
Ilustración 56: Ámbito de las variables .....	83
Ilustración 57: Ámbito de las variables .....	84
Ilustración 58: Manejo de cifras.....	95

## Acerca del autor

Rafael Alberto Moreno Parra

[ramsoftware@gmail.com](mailto:ramsoftware@gmail.com) o [enginelif@hotmai.com](mailto:enginelif@hotmai.com)

Sitio Web: <http://darwin.50webs.com> (dedicado a la investigación de algoritmos evolutivos y vida artificial).

Github: <https://github.com/ramsoftware>

Youtube: <https://www.youtube.com/@RafaelMorenoP>

## Licencia de este libro



## 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 2022 ® Enlace: <https://visualstudio.microsoft.com/es/vs/>

## Dedicatoria

A mis padres, a mi hermana....

Y a mi tropa gatuna: Sally, Suini, Grisú, Capuchina, Milú,  
Arián, Frac y mis recordados Tinita, Tammy, Vikingo y  
Michu.



## 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 \*/

A/001.cs

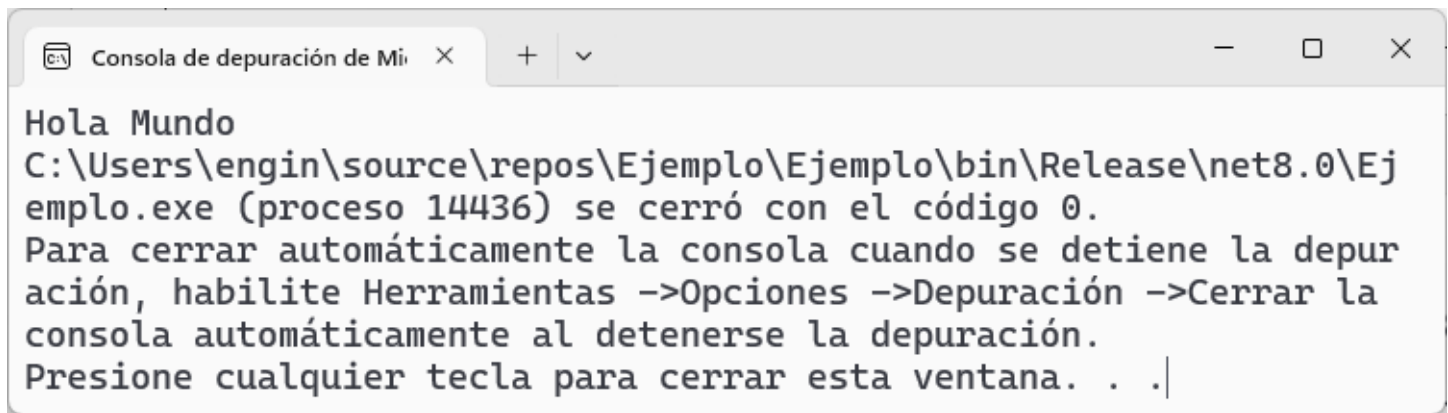
```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Este es un comentario de una sola línea  
  
            /* Este es un  
               comentario de  
               varias líneas */  
        }  
    }  
}
```

## 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.

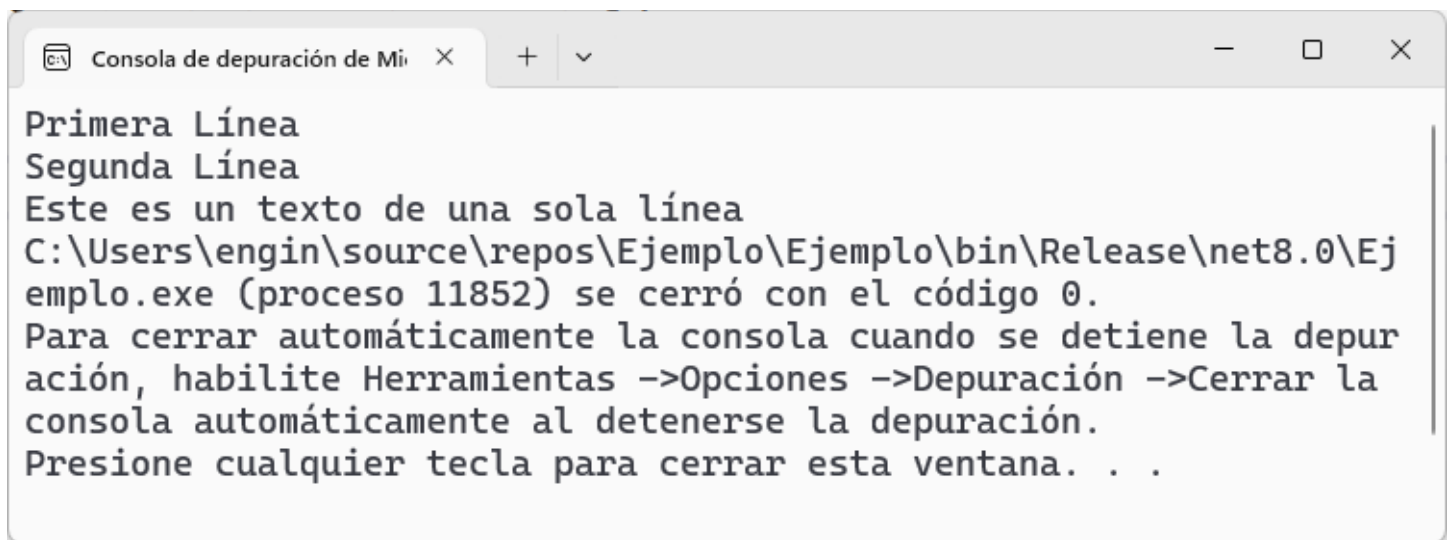
A/002.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Imprime en consola  
            Console.Write("Hola Mundo");  
        }  
    }  
}
```



*Ilustración 1: Ejecución del programa “Hola Mundo”. Impresión en consola.*

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //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");  
        }  
    }  
}
```



Consola de depuración de Mi... X + v - □ X

Primera Línea  
Segunda Línea  
Este es un texto de una sola línea  
C:\Users\engin\source\repos\Ejemplo\Ejemplo\bin\Release\net8.0\Ejemplo.exe (proceso 11852) se cerró con el código 0.  
Para cerrar automáticamente la consola cuando se detiene la depuración, habilite Herramientas ->Opciones ->Depuración ->Cerrar la consola automáticamente al detenerse la depuración.  
Presione cualquier tecla para cerrar esta ventana. . .

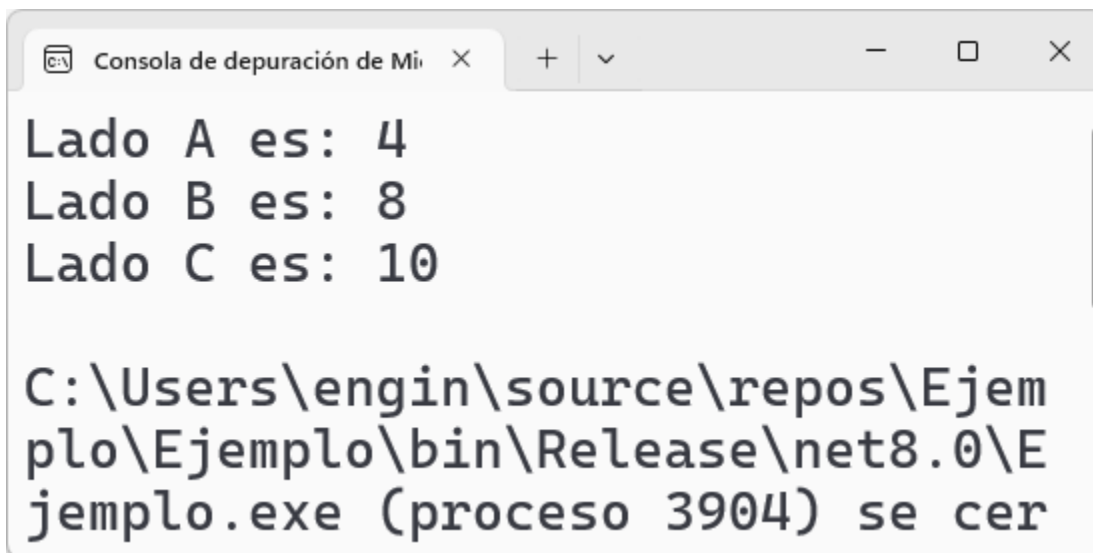
Ilustración 2: Diferencia entre Write y WriteLine

## Variables

C# es un lenguaje de programación tipado, luego al crear las variables se debe poner su tipo:

Tipo	Descripción	Valor mínimo	Valor máximo
byte	Número entero sin signo	0	255
sbyte	Número entero con signo	-128	127
short	Número entero con signo	-32768	32767
ushort	Número entero sin signo	0	65535
int	Número entero con signo (32 bits)	-2147483648	2147483647
uint	Número entero sin signo (32 bits)	0	4294967295
long	Número entero con signo (64 bits)	-9223372036854775808	9223372036854775807
ulong	Número entero sin signo (64 bits)	0	18446744073709551615
float	Número real de precisión simple (32 bits)		
double	Número real de precisión doble (64 bits)		
decimal	Número real de precisión más alta (128 bits)		
char	Carácter (1 byte)		
bool	Booleano (1 bit)		
string	Almacenamiento de caracteres alfanuméricos		
var	Es una forma de crear una variable local que no requiere especificar el tipo, ya que el compilador de C# lo deduce a partir de la expresión que se asigna.		

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Declara variables de tipo entero  
            int ladoA = 4;  
            int ladoB = 8;  
            int ladoC = 10;  
  
            //Imprime esas variables  
            Console.WriteLine("Lado A es: " + ladoA);  
            Console.WriteLine("Lado B es: " + ladoB);  
            Console.WriteLine("Lado C es: " + ladoC);  
        }  
    }  
}
```



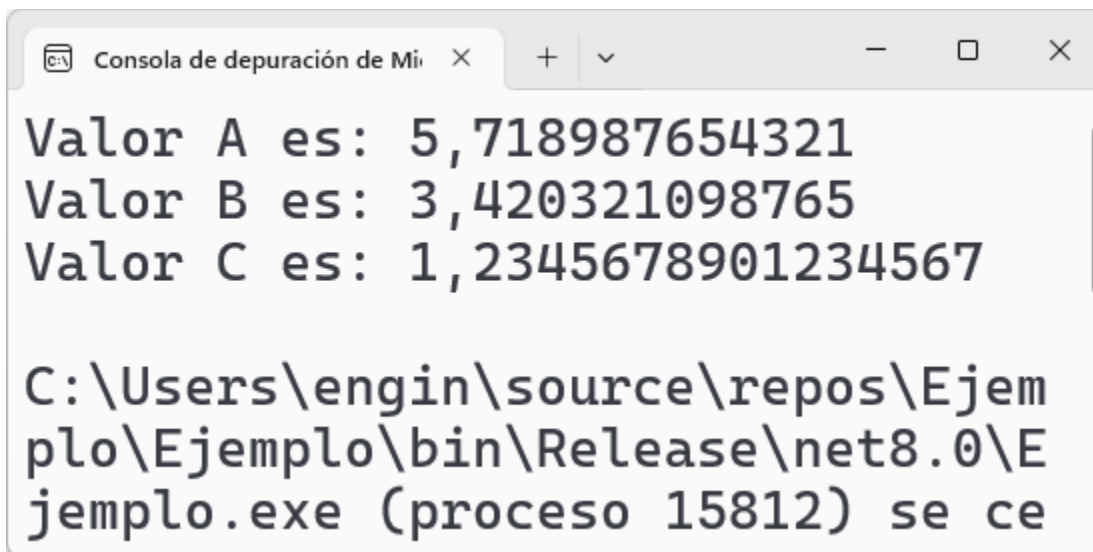
Consola de depuración de Mi... x + v - □ ×

Lado A es: 4  
Lado B es: 8  
Lado C es: 10

C:\Users\engin\source\repos\Ejemplo\Ejemplo\bin\Release\net8.0\Ejemplo.exe (proceso 3904) se cer

Ilustración 3: Variables de tipo entero

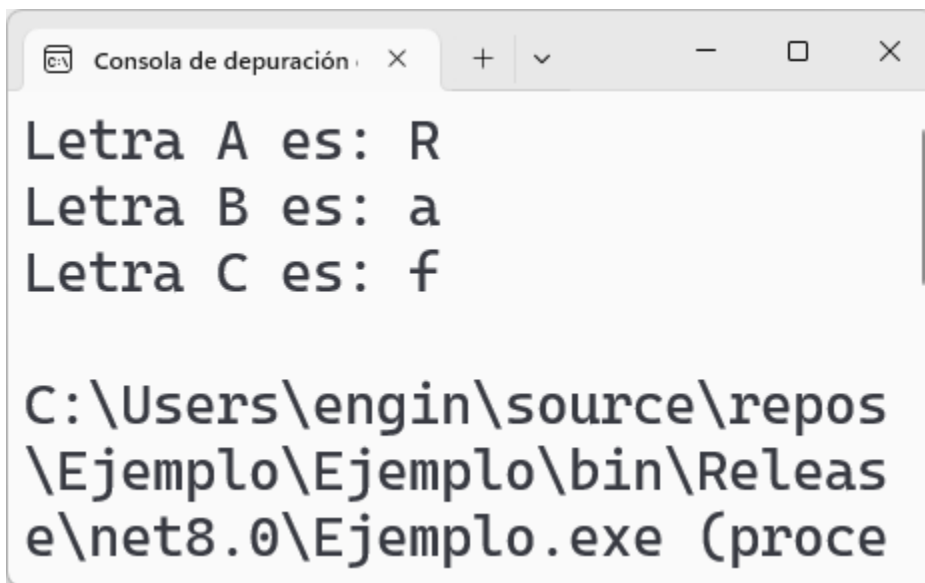
```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Declara variables de tipo double  
            double valA = 5.718987654321;  
            double valB = 3.420321098765;  
            double valC = 1.2345678901234567890123;  
  
            //Imprime esas variables  
            Console.WriteLine("Valor A es: " + valA);  
            Console.WriteLine("Valor B es: " + valB);  
            Console.WriteLine("Valor C es: " + valC);  
        }  
    }  
}
```



```
Consola de depuración de Mi  X  +  v  -  □  X  
Valor A es: 5,718987654321  
Valor B es: 3,420321098765  
Valor C es: 1,2345678901234567  
  
C:\Users\engin\source\repos\Ejem  
plo\Ejemplo\bin\Release\net8.0\E  
jemplo.exe (proceso 15812) se ce
```

Ilustración 4: Valores de tipo "double"

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Declara variables de tipo char  
            char letraA = 'R';  
            char letraB = 'a';  
            char letraC = 'f';  
  
            //Imprime esas variables  
            Console.WriteLine("Letra A es: " + letraA);  
            Console.WriteLine("Letra B es: " + letraB);  
            Console.WriteLine("Letra C es: " + letraC);  
        }  
    }  
}
```



Consola de depuración

Letra A es: R  
Letra B es: a  
Letra C es: f

C:\Users\engin\source\repos  
\Ejemplo\Ejemplo\bin\Releas  
e\net8.0\Ejemplo.exe (proce

Ilustración 5: Valores de tipo "char"

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Declara variables de tipo string  
            string cadenaA = "Esta es una prueba";  
            string cadenaB = "Programando en C#";  
            string cadenaC = "Con Visual Studio 2022";  
  
            //Imprime esas variables  
            Console.WriteLine("Cadena A es: " + cadenaA);  
            Console.WriteLine("Cadena B es: " + cadenaB);  
            Console.WriteLine("Cadena C es: " + cadenaC);  
        }  
    }  
}
```

```
Consola de depuración de Mi...  
Cadena A es: Esta es una prueba  
Cadena B es: Programando en C#  
Cadena C es: Con Visual Studio 2022  
  
C:\Users\engin\source\repos\Ejemplo\Ejemplo\bin\Release\net8.0\Ejemplo.exe (proceso 14452) se cerró con el código 0.
```

Ilustración 6: Valores de tipo "string"



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

A/008.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Declara variables de tipo float.  
            //Hay que usar el cast (float)  
            float valA = (float)5.718987654321;  
            float valB = (float)3.420321098765;  
            float valC = (float)1.234567890123456789012;  
  
            //Imprime esas variables  
            Console.WriteLine("Valor A es: " + valA);  
            Console.WriteLine("Valor B es: " + valB);  
            Console.WriteLine("Valor C es: " + valC);  
        }  
    }  
}
```

```
Consola de depuración de Mi...  
Valor A es: 5,7189875  
Valor B es: 3,420321  
Valor C es: 1,2345679  
  
C:\Users\engin\source\repos\Ejemplo\Ejemplo\bin\Release\net8.0\Ejemplo.exe (proceso 2448) se cerró con el código 0.
```

Ilustración 7: Valores almacenados por variables de tipo "float"

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:

A/009.cs

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //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);

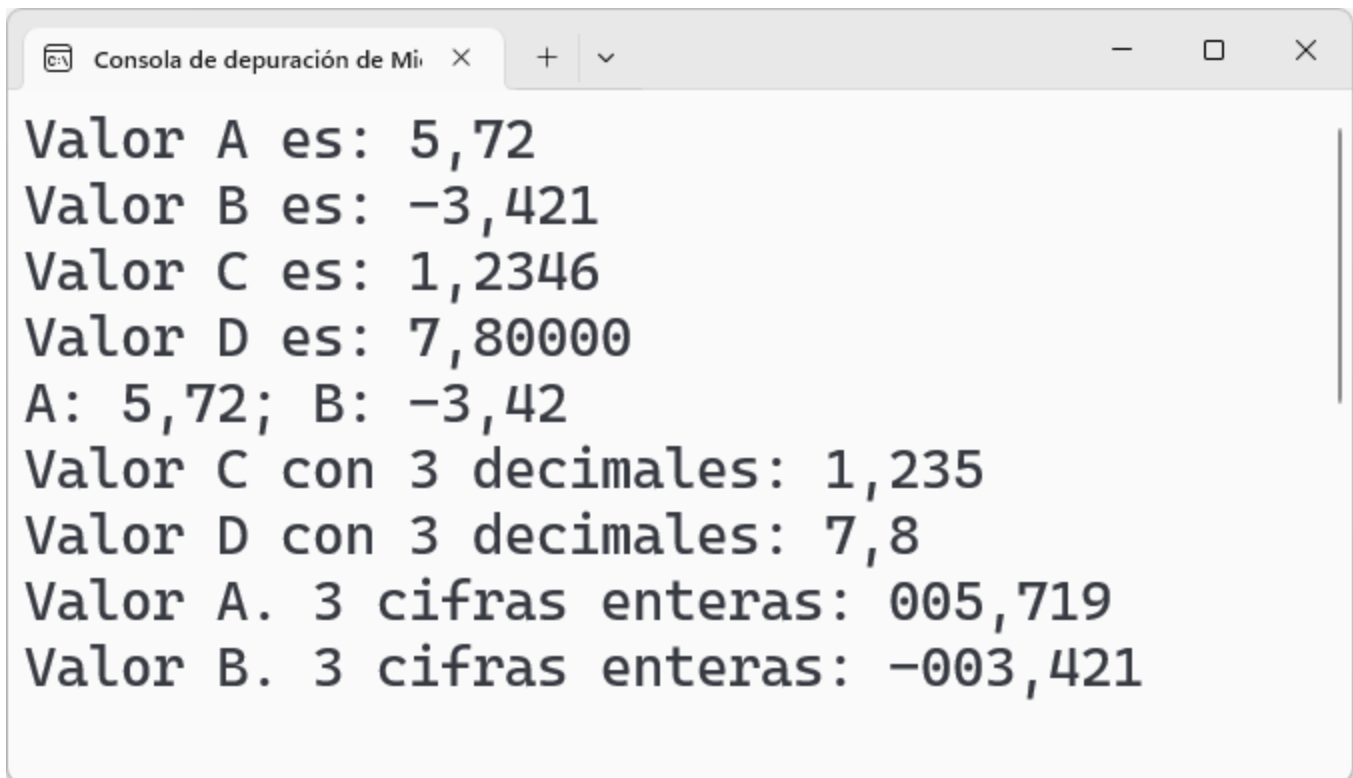
            //Imprime con un formato de al menos cuatro decimales
            Console.WriteLine("Valor C es: {0:0.0000}", valC);

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

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

            /* Máximo tres decimales */
            Console.WriteLine("Valor C con 3 decimales: {0:0.###}", valC);
            Console.WriteLine("Valor D con 3 decimales: {0:0.###}", valD);

            /* Mínimo tres cifras antes del punto decimal */
            Console.WriteLine("Valor A. 3 cifras enteras: {0:000.###}", valA);
            Console.WriteLine("Valor B. 3 cifras enteras: {0:000.###}", valB);
        }
    }
}
```



A screenshot of a debug console window titled 'Consola de depuración de Mi'. The window contains the following text:

```
Valor A es: 5,72
Valor B es: -3,421
Valor C es: 1,2346
Valor D es: 7,80000
A: 5,72; B: -3,42
Valor C con 3 decimales: 1,235
Valor D con 3 decimales: 7,8
Valor A. 3 cifras enteras: 005,719
Valor B. 3 cifras enteras: -003,421
```

*Ilustración 8: Números impresos con determinados formatos*

```

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //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("Impresión con formato de miles y 3 decimales");
            Console.WriteLine("Valor A: {0:0,0.0}", valA);
            Console.WriteLine("Valor A: {0:0,0.000}", valA);

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

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

```

```

Consola de depuración de Mi
Impresión con formato de miles y 3 decimales
Valor A: 17.902,8
Valor A: 17.902,842

Valor B: -871.901.372,4
Valor B: -871.901.372,421

Valor C: 89.341.759.342,2
Valor C: 89.341.759.342,168

```

Ilustración 9: Impresión de números de tipo double con formato

```

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Declara variables de tipo double
            double valA = 17902.8421;
            double valB = -871901372.420821098765;
            double valC = 529403.1678;

            //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);
        }
    }
}

```

Consola de depuración de Mi...

```

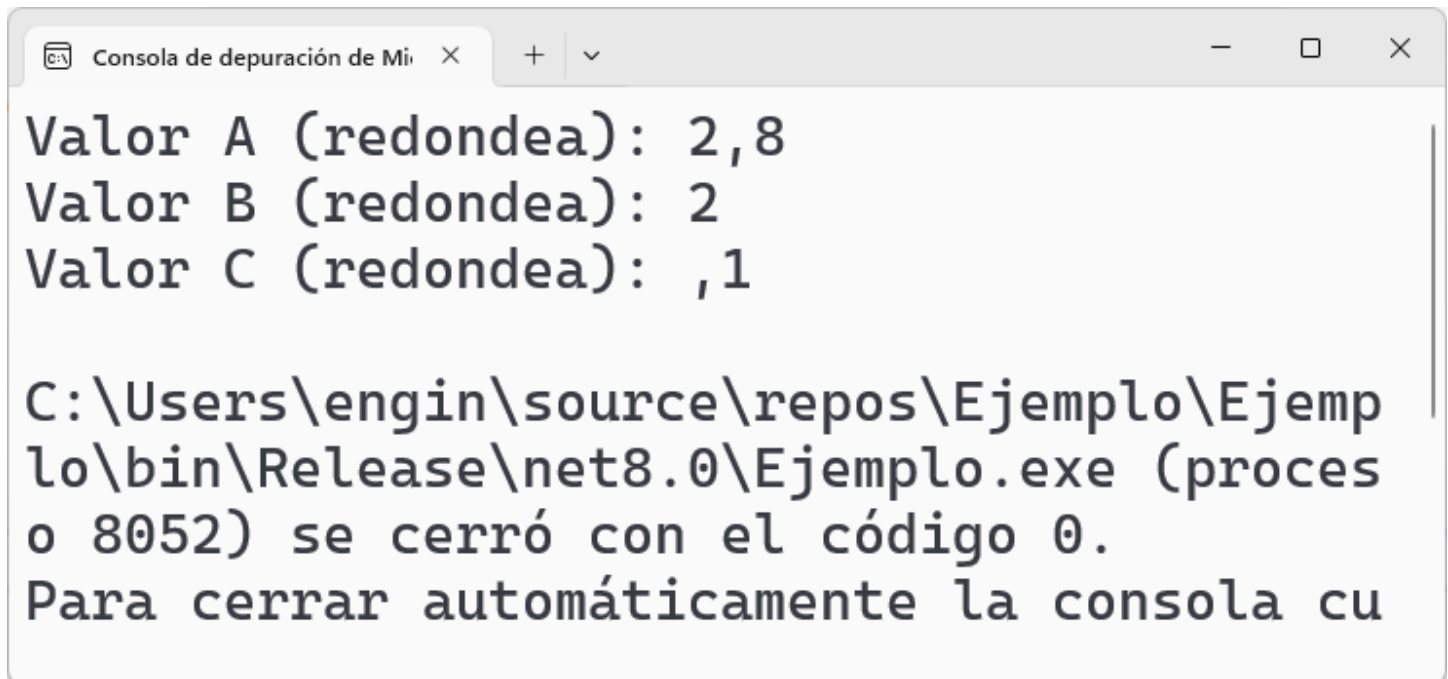
Valor A es:                17902,8
Valor B es:               -871901372,4
Valor C es:                529403,2

C:\Users\engin\source\repos\Ejemplo\Ejemplo\bin\Release\net8.0\Ejemplo.exe (proceso 1392) se cerró con el código 0.
Para cerrar automáticamente la consola cu

```

Ilustración 10: Valores tipo "double" con formato

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Declara variables de tipo double  
            double valA = 2.76;  
            double valB = 2.04;  
            double valC = 0.14;  
  
            //Imprime solo si la cifra es significativa  
            //redondea también.  
            Console.WriteLine("Valor A (redondea): {0:0.0}", valA);  
            Console.WriteLine("Valor B (redondea): {0:0.0}", valB);  
            Console.WriteLine("Valor C (redondea): {0:0.0}", valC);  
        }  
    }  
}
```



Consola de depuración de Mi

Valor A (redondea): 2,8  
Valor B (redondea): 2  
Valor C (redondea): ,1

C:\Users\engin\source\repos\Ejemplo\Ejemplo\bin\Release\net8.0\Ejemplo.exe (proceso 8052) se cerró con el código 0.  
Para cerrar automáticamente la consola cu

Ilustración 11: Valores tipo "double" con formato de redondeo

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Operador suma
            double valA = 1 + 2 + 3;

            //Operador resta
            double valB = 4 - 5 - 6;

            //Operador multiplicación
            double valC = 7 * 8 * 9;

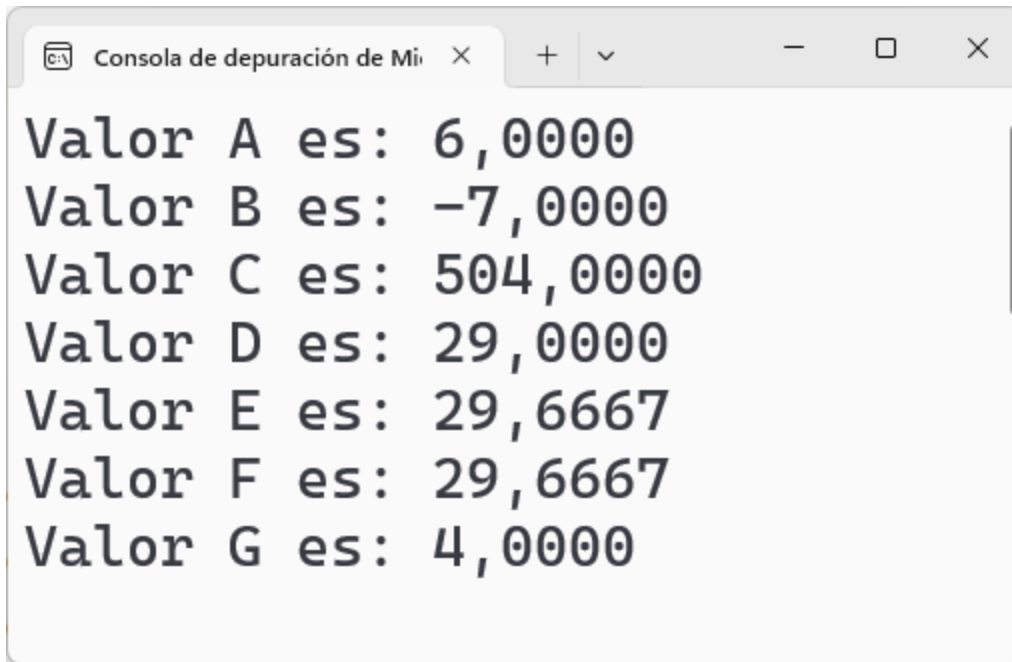
            //Operador división usando enteros
            double valD = 178 / 2 / 3;

            //Operador división usando números reales
            double valE = 178 / 2.0 / 3.0;

            //Operador división usando números enteros haciendo cast
            double valF = (double)178 / 2 / 3;

            //Operador división modular
            double valG = 70 % 6;

            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);
        }
    }
}
```



```
Valor A es: 6,0000
Valor B es: -7,0000
Valor C es: 504,0000
Valor D es: 29,0000
Valor E es: 29,6667
Valor F es: 29,6667
Valor G es: 4,0000
```

*Ilustración 12: Resultados de operaciones*

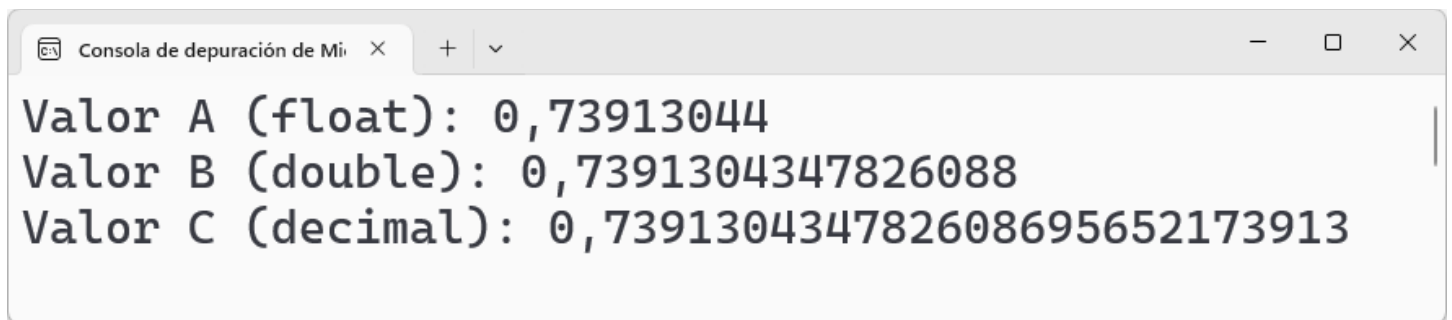
En valD el resultado es un entero porque C# lo toma como si fuese una división entera de números enteros por lo que el resultado es entero. En cambio, valE y valF retornan el resultado real, el primero por usar notación de número real (el uso del punto) y el otro por usar cast (double).



# Diferencias de precisión entre float, double y decimal

A/014.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Compara la precisión entre float y double  
            float valA = (float)(1.7 / 2.3);  
            double valB = 1.7 / 2.3;  
            decimal valC = (decimal)1.7 / (decimal)2.3;  
  
            Console.WriteLine("Valor A (float): " + valA);  
            Console.WriteLine("Valor B (double): " + valB);  
            Console.WriteLine("Valor C (decimal): " + valC);  
        }  
    }  
}
```



Consola de depuración de Mi

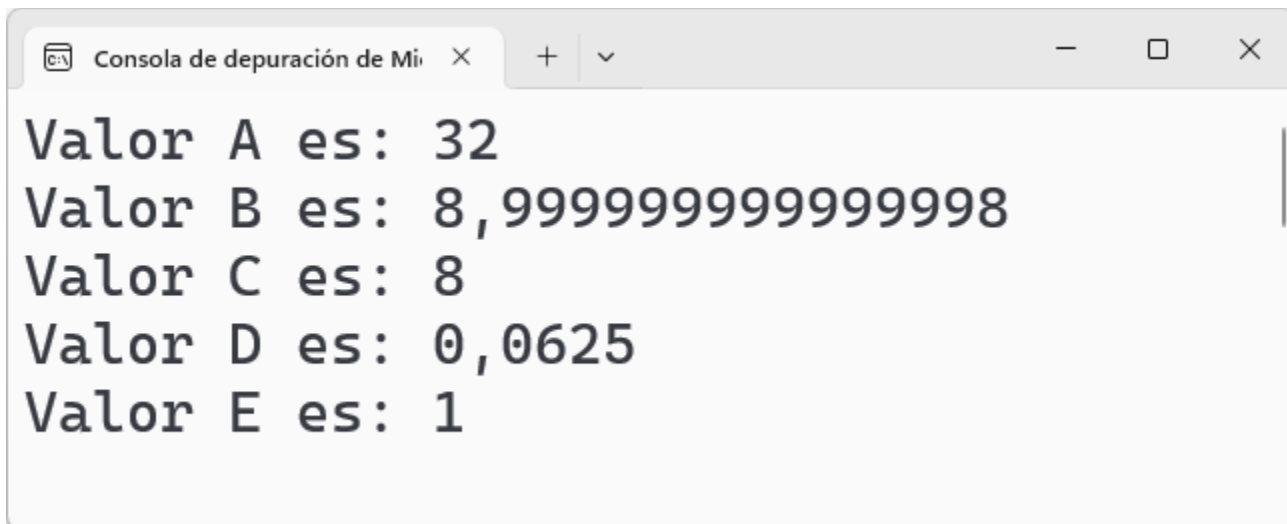
Valor A (float): 0,73913044  
Valor B (double): 0,7391304347826088  
Valor C (decimal): 0,739130434782608695652173913

Ilustración 13: Diferencias entre float, double y decimal

# Función de potencia

A/015.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Potencia (^)  
            double valA = Math.Pow(2, 5);  
  
            //Raíz cúbica de 729  
            double valB = Math.Pow(729, (double)1 / 3);  
  
            //Raíz cuadrada de 64  
            double valC = Math.Pow(64, (double)1 / 2);  
  
            //  $4^{(-2)} = 1/(4^2) = 1/16$   
            double valD = Math.Pow(4, -2);  
  
            //  $4^0$   
            double valE = Math.Pow(4, 0);  
  
            Console.WriteLine("Valor A es: " + valA);  
            Console.WriteLine("Valor B es: " + valB);  
            Console.WriteLine("Valor C es: " + valC);  
            Console.WriteLine("Valor D es: " + valD);  
            Console.WriteLine("Valor E es: " + valE);  
        }  
    }  
}
```



```
Consola de depuración de Mi  +  -  □  ×  
  
Valor A es: 32  
Valor B es: 8,9999999999999998  
Valor C es: 8  
Valor D es: 0,0625  
Valor E es: 1
```

Ilustración 14: Función de potencia



# Orden de evaluación de los operadores

A/016.cs

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            /* 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);
            double valB = (double)1 + 2 * 3 / 4 - 5;
            double valC = (double)3 + 5 - 2 * 4 / 8;
            double valD = (double)3 * 2 + Math.Pow(3, 2);

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

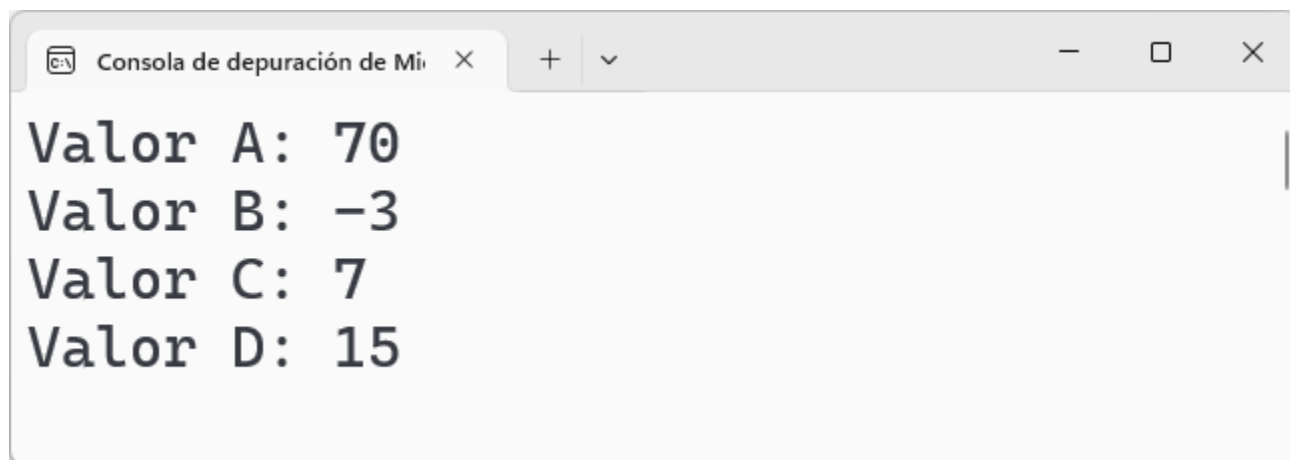
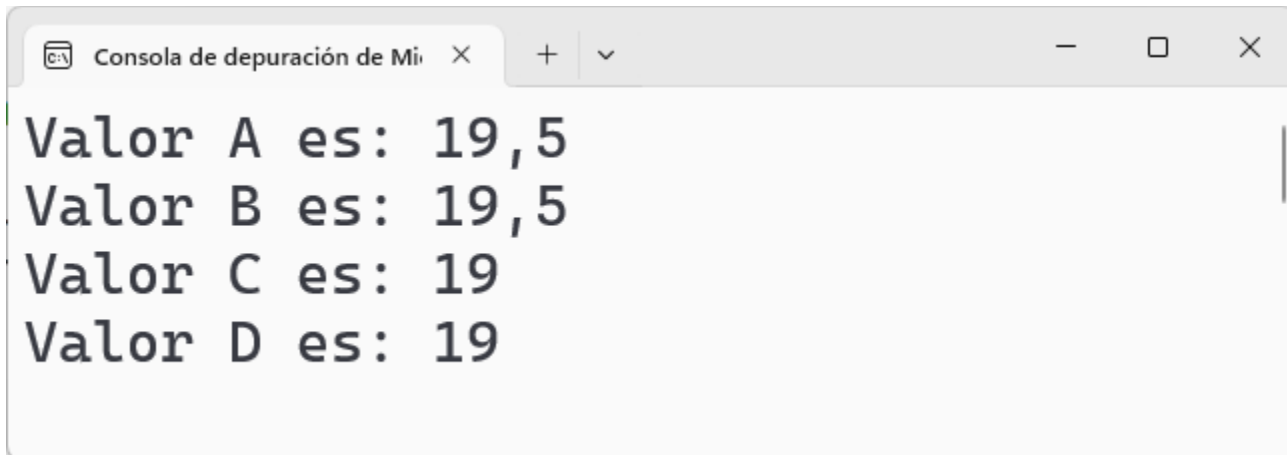


Ilustración 15: Orden de evaluación de los operadores

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //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);  
            Console.WriteLine("Valor B es: " + valB);  
            Console.WriteLine("Valor C es: " + valC);  
            Console.WriteLine("Valor D es: " + valD);  
        }  
    }  
}
```



```
Consola de depuración de Mi  X  +  v  -  □  X  
Valor A es: 19,5  
Valor B es: 19,5  
Valor C es: 19  
Valor D es: 19
```

Ilustración 16: Problemas del uso del "cast"

```

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Falla porque se interpreta como operación de enteros
            double valA = (4 / 3) + (1 / 2) - (5 / 4);

            //Falla porque el cast solo cubre la primera operación
            double valB = (double)4 / 3 + (1 / 2) - (5 / 4);

            //Operación correcta
            double valC = (4.0 / 3.0) + (1.0 / 2.0) - (5.0 / 4.0);

            //Operación correcta pero se deben usar varios cast
            double valD = (double)4 / 3 + (double)1 / 2 - (double)5 / 4;

            //Falla porque el cast solo cubre la primera operación
            double valE = (double)4 / 3 + 1 / 2 - 5 / 4;

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

```

```

Consola de depuración de Mi
Valor A: 0
Valor B: 0,333333333333333326
Valor C: 0,583333333333333333
Valor D: 0,583333333333333333
Valor E: 0,333333333333333326

```

Ilustración 17: Problemas del uso del "cast"

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

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

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

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

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

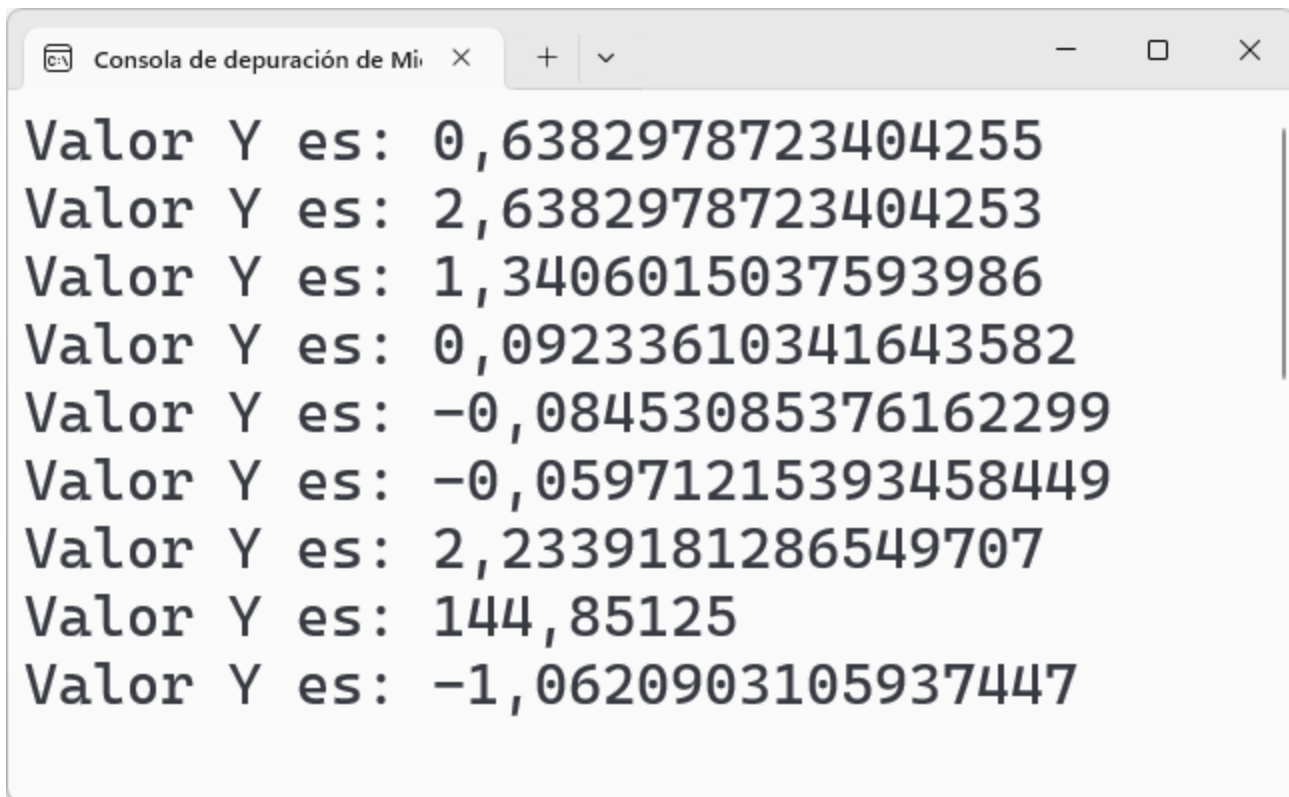
            Y = 3 / (2 - (X * X + 5));
            Console.WriteLine("Valor Y es: " + Y);

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

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

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

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



A screenshot of a C# console window titled "Consola de depuración de Mi". The window displays nine lines of output, each starting with "Valor Y es:". The values are formatted with commas as decimal separators and no trailing zeros. The values are: 0,6382978723404255; 2,6382978723404253; 1,3406015037593986; 0,09233610341643582; -0,08453085376162299; -0,05971215393458449; 2,2339181286549707; 144,85125; and -1,0620903105937447.

```
Valor Y es: 0,6382978723404255
Valor Y es: 2,6382978723404253
Valor Y es: 1,3406015037593986
Valor Y es: 0,09233610341643582
Valor Y es: -0,08453085376162299
Valor Y es: -0,05971215393458449
Valor Y es: 2,2339181286549707
Valor Y es: 144,85125
Valor Y es: -1,0620903105937447
```

*Ilustración 18: Formato algebraico a C#*



# Diferencias entre usar el cast y la conversión

A/020.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Conversión vs Cast  
            double valA = 15.7;  
  
            int valB = Convert.ToInt32(valA); //Redondea  
            int valC = (int)valA; //Trunca  
  
            //Ejemplos  
            Console.WriteLine("Valor B: " + valB);  
            Console.WriteLine("Valor C: " + valC);  
        }  
    }  
}
```

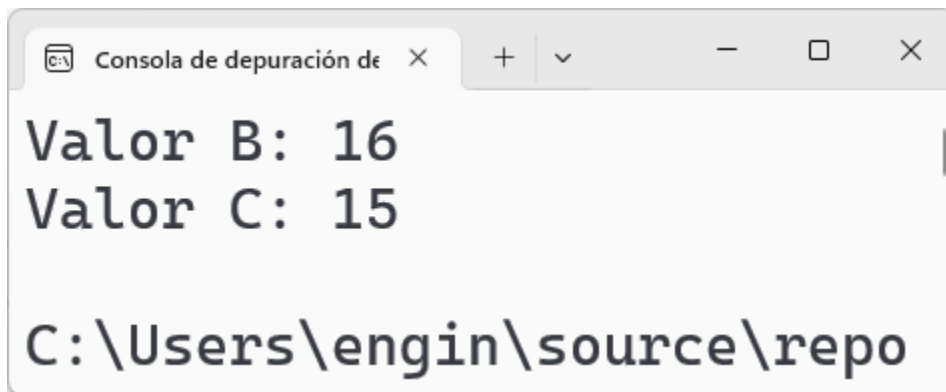


Ilustración 19: Diferencias entre usar el cast y la conversión

# Conversión de números reales con el punto decimal

A/021.cs

```
using System.Globalization;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //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: " + valB);

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

            //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: " + valF);
        }
    }
}
```

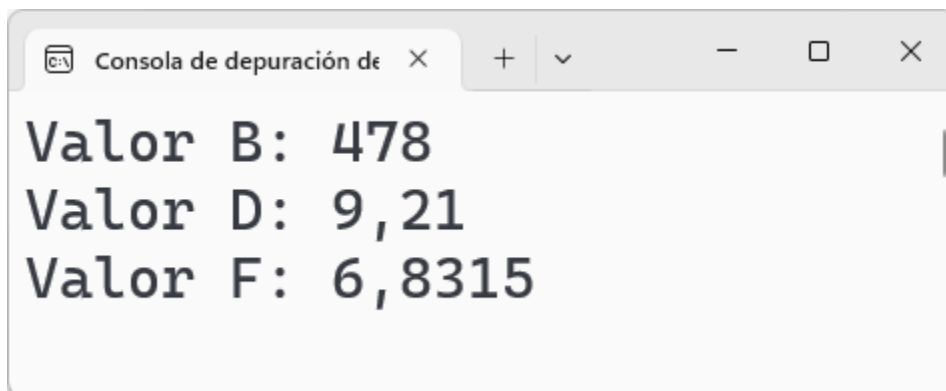


Ilustración 20: Conversión de números reales con el punto decimal

```
using System.Globalization;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //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);

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

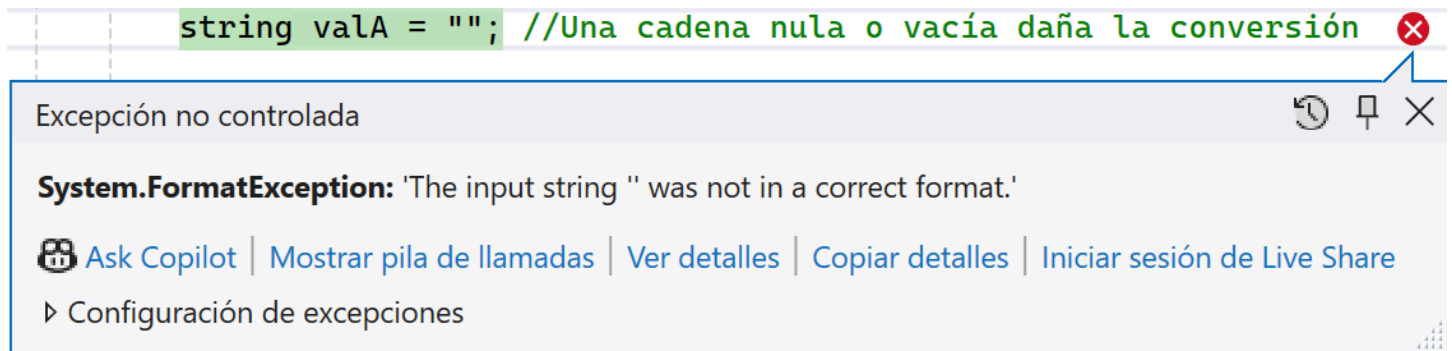


Ilustración 21: Mensaje de error

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

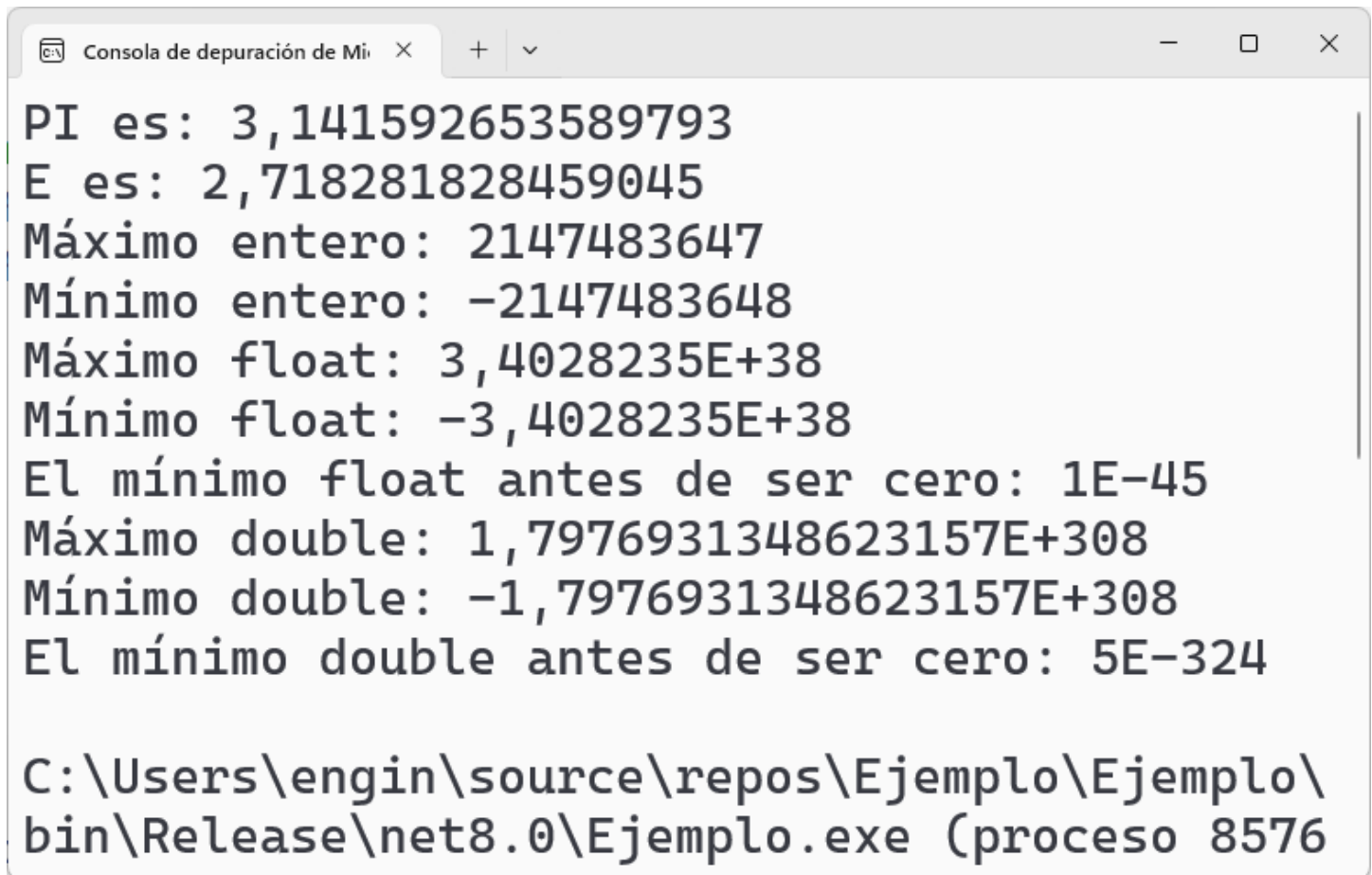
            //Máximo y mínimo valor entero
            int maximoEntero = int.MaxValue;
            int minimoEntero = int.MinValue;
            Console.WriteLine("Máximo entero: " + maximoEntero);
            Console.WriteLine("Mínimo entero: " + minimoEntero);

            //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);
            Console.WriteLine("Mínimo float: " + minimofloat);
            Console.WriteLine("El mínimo float antes de ser cero: " + minimoCero);

            //Máximo y mínimo valor double
            double maximodouble = double.MaxValue;
            double minimodouble = double.MinValue;

            //El mínimo valor antes de ser cero
            double CeroE = double.Epsilon;

            Console.WriteLine("Máximo double: " + maximodouble);
            Console.WriteLine("Mínimo double: " + minimodouble);
            Console.WriteLine("El mínimo double antes de ser cero: " + CeroE);
        }
    }
}
```



The image shows a screenshot of a Windows Debug Console window. The title bar at the top reads "Consola de depuración de Mi" followed by a close button (X) and a dropdown menu with a plus sign and a downward arrow. The window contains the following text:

```
PI es: 3,141592653589793
E es: 2,718281828459045
Máximo entero: 2147483647
Mínimo entero: -2147483648
Máximo float: 3,4028235E+38
Mínimo float: -3,4028235E+38
El mínimo float antes de ser cero: 1E-45
Máximo double: 1,7976931348623157E+308
Mínimo double: -1,7976931348623157E+308
El mínimo double antes de ser cero: 5E-324

C:\Users\engin\source\repos\Ejemplo\Ejemplo\
bin\Release\net8.0\Ejemplo.exe (proceso 8576)
```

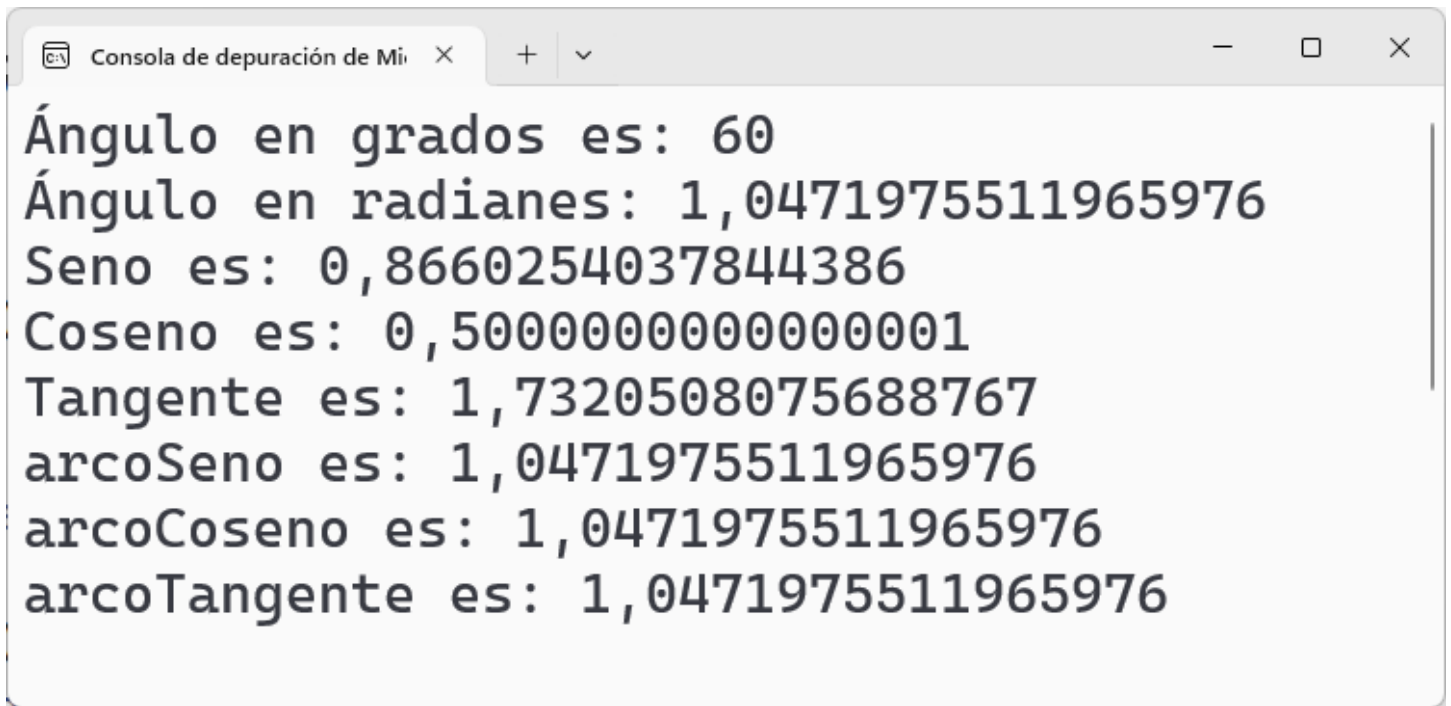
*Ilustración 22: Constantes matemáticas*

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //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);
            Console.WriteLine("Ángulo en radianes: " + anguloRadian);
            Console.WriteLine("Seno es: " + valSeno);
            Console.WriteLine("Coseno es: " + valCoseno);
            Console.WriteLine("Tangente es: " + valTangente);
            Console.WriteLine("arcoSeno es: " + arcoSeno);
            Console.WriteLine("arcoCoseno es: " + arcoCoseno);
            Console.WriteLine("arcoTangente es: " + arcoTangente);
        }
    }
}
```

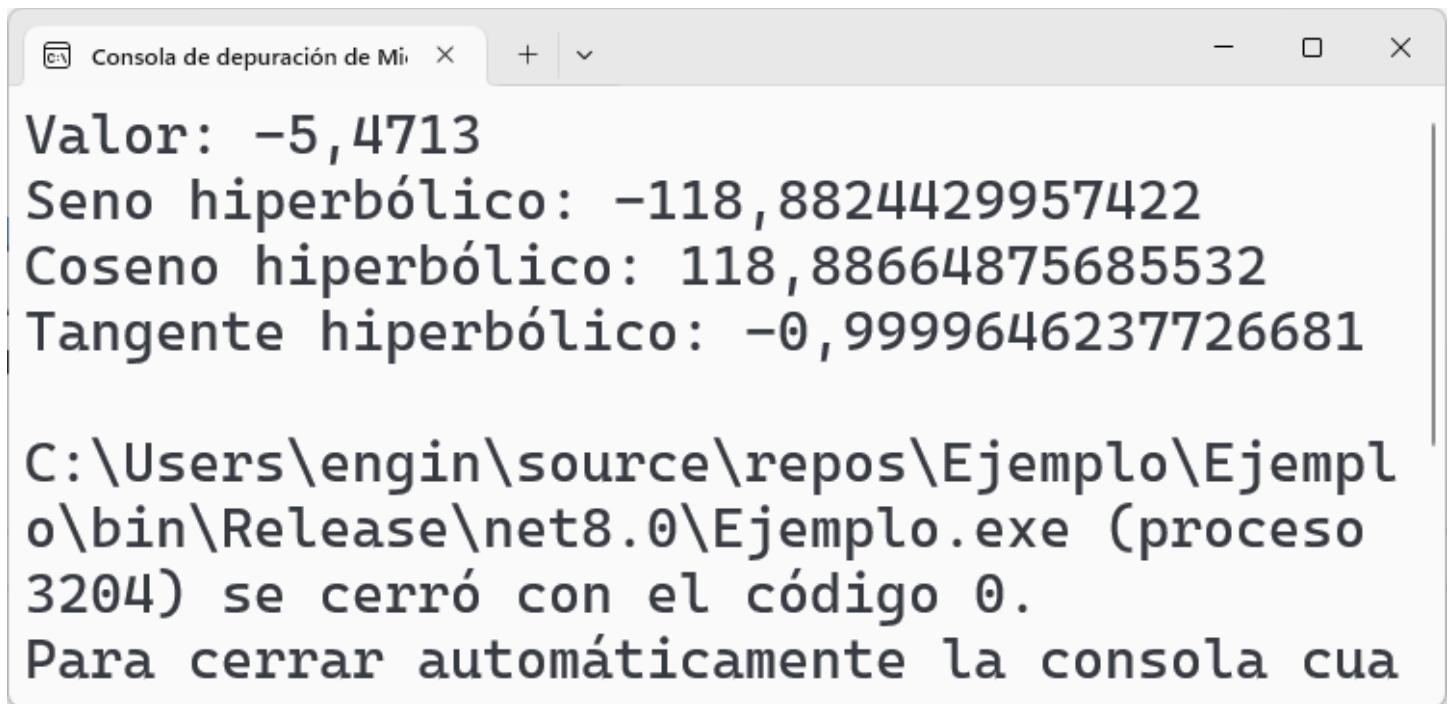


The image shows a web browser window with the developer console open. The console title is 'Consola de depuración de Mi'. It displays the following output:

```
Ángulo en grados es: 60
Ángulo en radianes: 1,0471975511965976
Seno es: 0,8660254037844386
Coseno es: 0,50000000000000000001
Tangente es: 1,7320508075688767
arcoSeno es: 1,0471975511965976
arcoCoseno es: 1,0471975511965976
arcoTangente es: 1,0471975511965976
```

*Ilustración 23: Funciones trigonométricas*

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //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);  
            Console.WriteLine("Seno hiperbólico: " + valSenoH);  
            Console.WriteLine("Coseno hiperbólico: " + valCosenoH);  
            Console.WriteLine("Tangente hiperbólico: " + valTangenteH);  
        }  
    }  
}
```



```
Consola de depuración de Mi...  
Valor: -5,4713  
Seno hiperbólico: -118,8824429957422  
Coseno hiperbólico: 118,88664875685532  
Tangente hiperbólico: -0,9999646237726681  
  
C:\Users\engin\source\repos\Ejemplo\Ejempl  
o\bin\Release\net8.0\Ejemplo.exe (proceso  
3204) se cerró con el código 0.  
Para cerrar automáticamente la consola cua
```

Ilustración 24: Funciones hiperbólicas



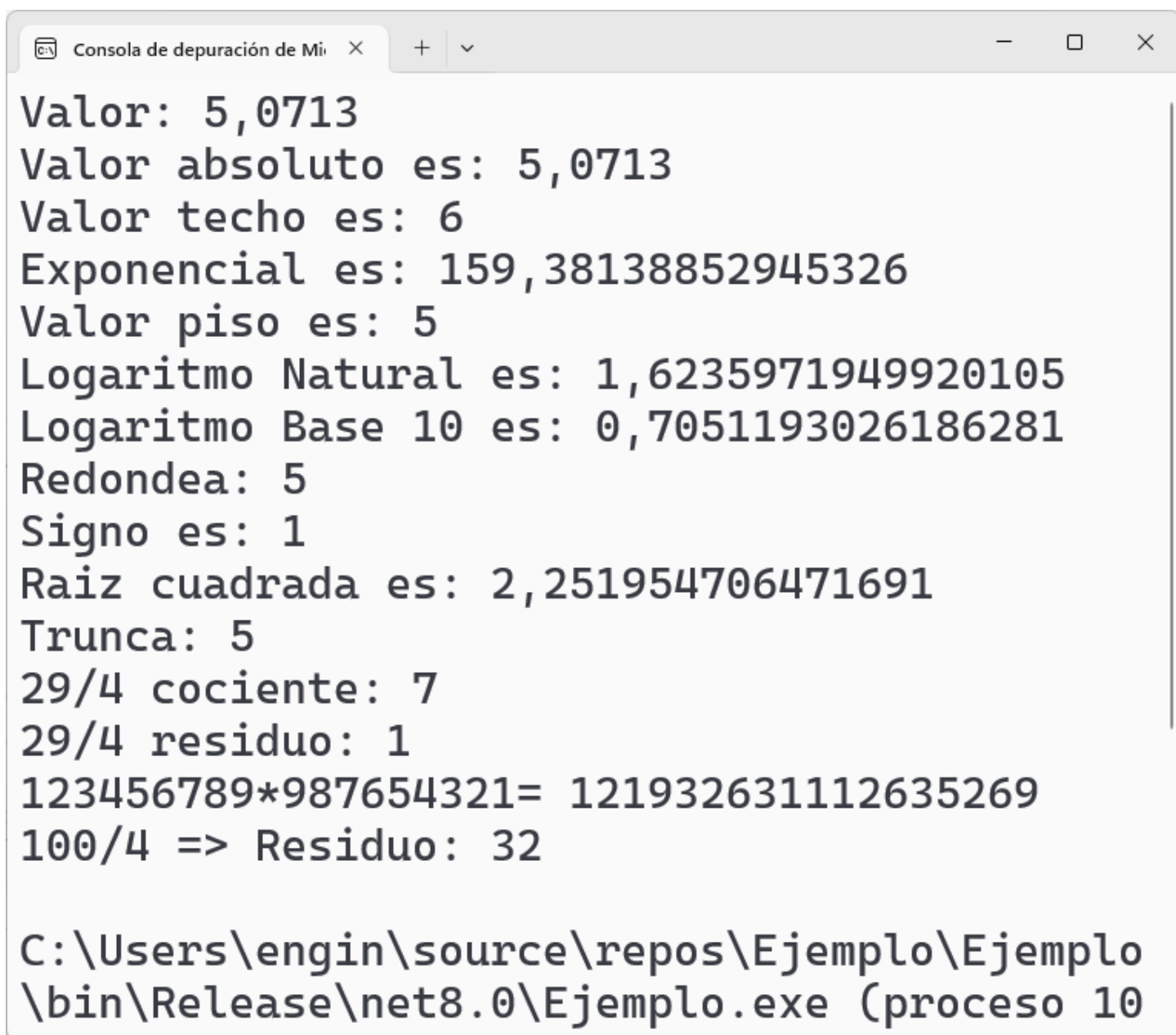
```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Otras funciones matemáticas
            double valorReal = 5.0713;
            double valAbsoluto = Math.Abs(valorReal); //valor absoluto
            double valTecho = Math.Ceiling(valorReal); //entero superior
            double valExponencial = Math.Exp(valorReal);
            double valPiso = Math.Floor(valorReal);
            double valLogaritmoNatural = Math.Log(valorReal);
            double valLogaritmoBase10 = Math.Log10(valorReal);
            double 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);
            Console.WriteLine("Valor absoluto es: " + valAbsoluto);
            Console.WriteLine("Valor techo es: " + valTecho);
            Console.WriteLine("Exponencial es: " + valExponencial);
            Console.WriteLine("Valor piso es: " + valPiso);
            Console.WriteLine("Logaritmo Natural es: " + valLogaritmoNatural);
            Console.WriteLine("Logaritmo Base 10 es: " + valLogaritmoBase10);
            Console.WriteLine("Redondea: " + valRedondea);
            Console.WriteLine("Signo es: " + valSigno);
            Console.WriteLine("Raiz cuadrada es: " + valRaizC);
            Console.WriteLine("Trunca: " + valTrunca);

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

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

            //División modular
            decimal dividendo = 100, divisor = 34;
            decimal Otroresiduo = dividendo % divisor;
            Console.WriteLine("100/4 => Residuo: " + Otroresiduo);
        }
    }
}
```



A screenshot of a Windows Debug Console window. The title bar shows 'Consola de depuración de Mi' and standard window controls. The console displays a series of mathematical operations and their results in a monospaced font. The operations include finding the absolute value, ceiling, exponential, floor, natural logarithm, base-10 logarithm, rounding, sign, square root, truncation, division with quotient and remainder, and a large multiplication. The window path is visible at the bottom.

```
Valor: 5,0713
Valor absoluto es: 5,0713
Valor techo es: 6
Exponencial es: 159,38138852945326
Valor piso es: 5
Logaritmo Natural es: 1,6235971949920105
Logaritmo Base 10 es: 0,7051193026186281
Redondea: 5
Signo es: 1
Raiz cuadrada es: 2,251954706471691
Trunca: 5
29/4 cociente: 7
29/4 residuo: 1
123456789*987654321= 121932631112635269
100/4 => Residuo: 32

C:\Users\engin\source\repos\Ejemplo\Ejemplo
\bin\Release\net8.0\Ejemplo.exe (proceso 10
```

Ilustración 25: 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.

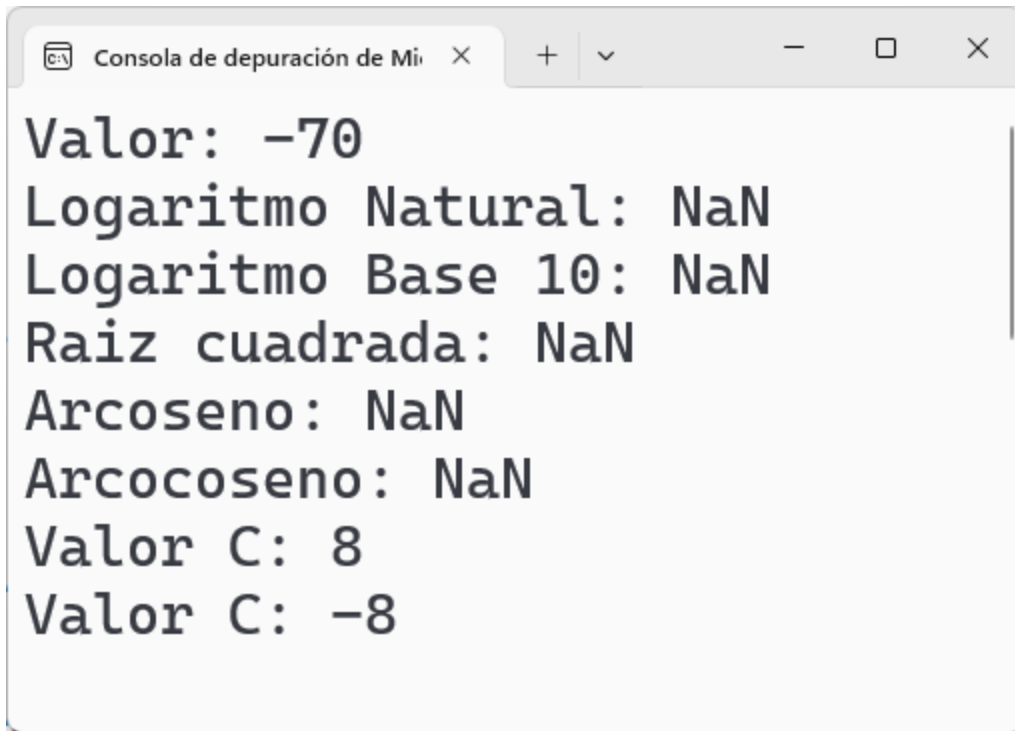
A/027.cs

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            // 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);
            Console.WriteLine("Logaritmo Natural: " + valLogaritmoNatural);
            Console.WriteLine("Logaritmo Base 10: " + valLogaritmoBase10);
            Console.WriteLine("Raiz cuadrada: " + valRaizC);
            Console.WriteLine("Arcoseno: " + arcoSeno);
            Console.WriteLine("Arcocoseno: " + arcoCoseno);

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

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



The image shows a browser's developer console with the following output:

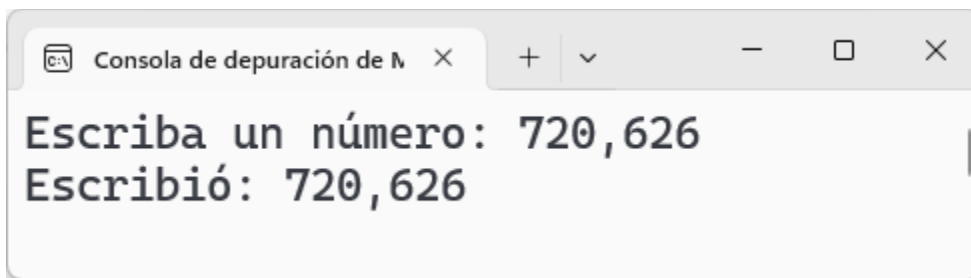
```
Valor: -70
Logaritmo Natural: NaN
Logaritmo Base 10: NaN
Raiz cuadrada: NaN
Arcoseno: NaN
Arcocoseno: NaN
Valor C: 8
Valor C: -8
```

*Ilustración 26: Manejo del NaN (Not a Number) y el infinito*

# Leer un número por consola

A/028.cs

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



*Ilustración 27: Leer un número por consola*

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //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 + " es mayor que " + valorB);
            }

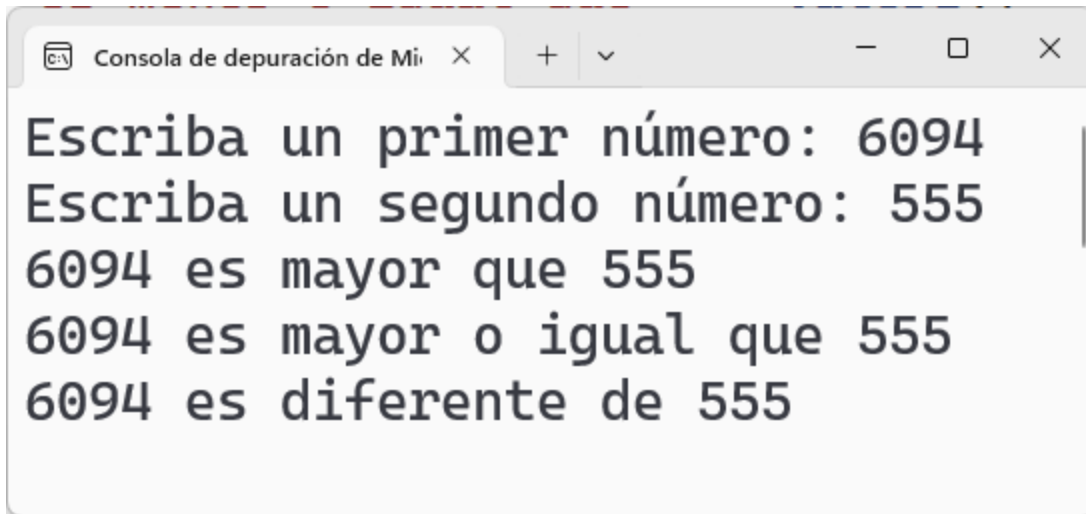
            if (valorA >= valorB) {
                Console.WriteLine(valorA + " es mayor o igual que " + valorB);
            }

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

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

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

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

A screenshot of a web browser's developer console. The console has a tab labeled 'Consola de depuración de Mi' with a close button. The output text is as follows:

```
Escriba un primer número: 6094
Escriba un segundo número: 555
6094 es mayor que 555
6094 es mayor o igual que 555
6094 es diferente de 555
```

*Ilustración 28: Si condicional*

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //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 + " es mayor que " + valorB);  
            }  
            else { //de lo contrario  
                Console.WriteLine(valorA + " es menor o igual que " + valorB);  
            }  
        }  
    }  
}
```

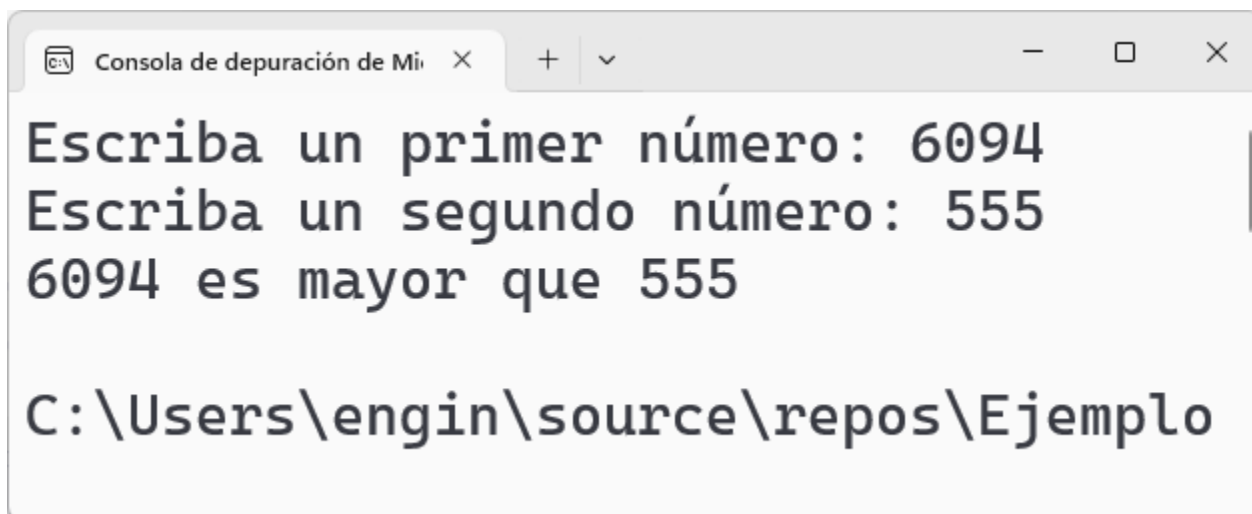


Ilustración 29: Uso de if...else



```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //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 + " es mayor que " + valorB);  
            }  
            else if (valorA < valorB) {  
                Console.WriteLine(valorA + " es menor que " + valorB);  
            }  
            else {  
                Console.WriteLine(valorA + " es igual a " + valorB);  
            }  
        }  
    }  
}
```

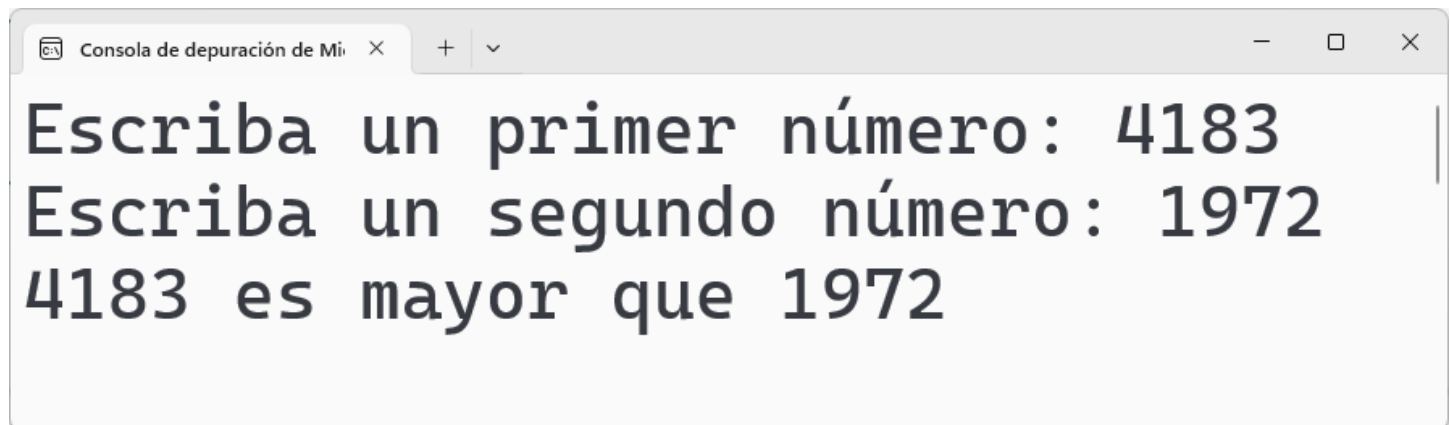


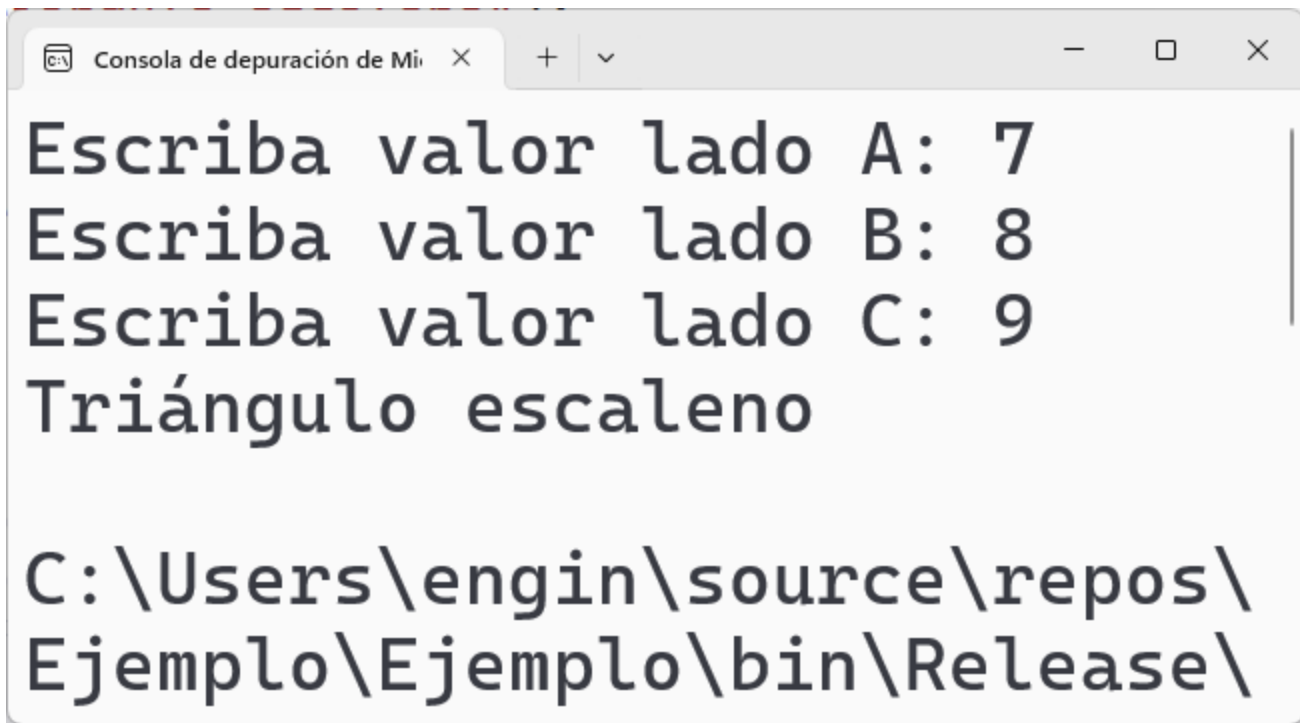
Ilustración 30: Uso de if... else if ... else

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //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");
            }
        }
    }
}
```



```
Consola de depuración de Mi X + v - □ X  
Escriba valor lado A: 7  
Escriba valor lado B: 8  
Escriba valor lado C: 9  
Triángulo escaleno  
  
C:\Users\engin\source\repos\  
Ejemplo\Ejemplo\bin\Release\
```

*Ilustración 31: Uso del AND &&*

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //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");  
            }  
        }  
    }  
}
```

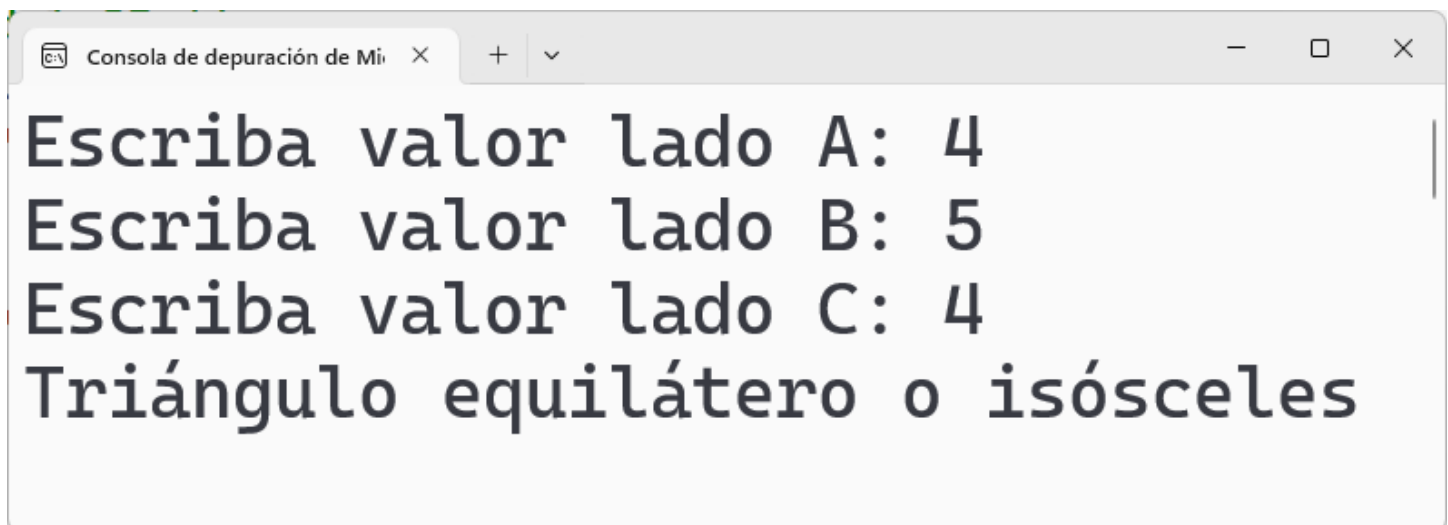


Ilustración 32: Uso del OR ||

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Lee un valor entero  
            Console.WriteLine("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("Fuera del rango de 1 a 5");  
                    break; //Inclusive el default requiere break  
            }  
        }  
    }  
}
```

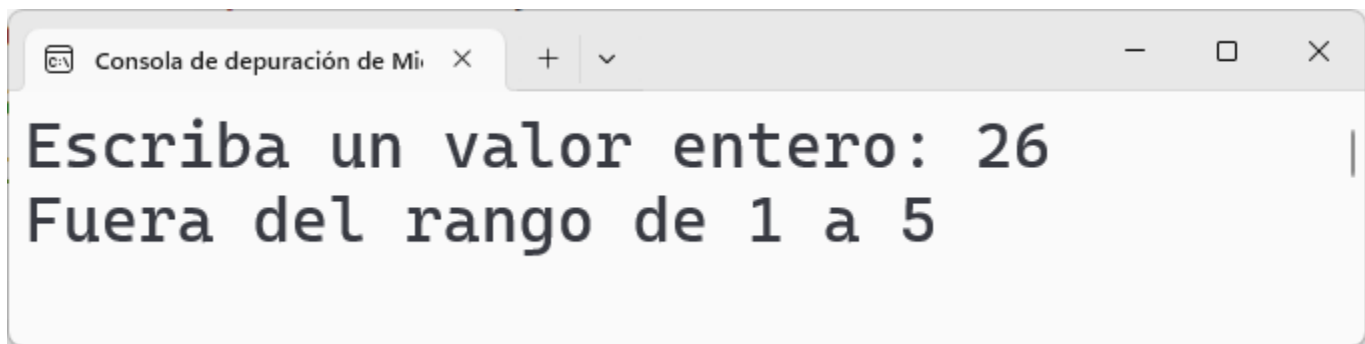


Ilustración 33: Uso del switch

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //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);  
        }  
    }  
}
```

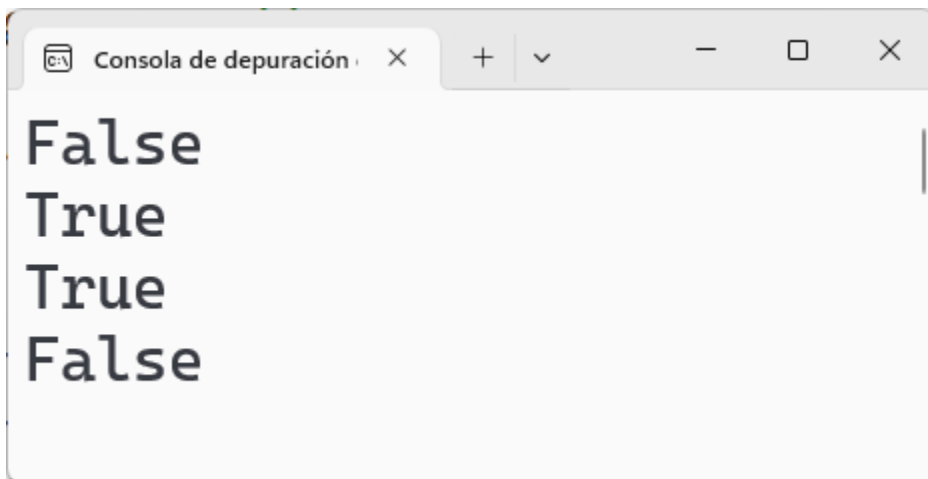
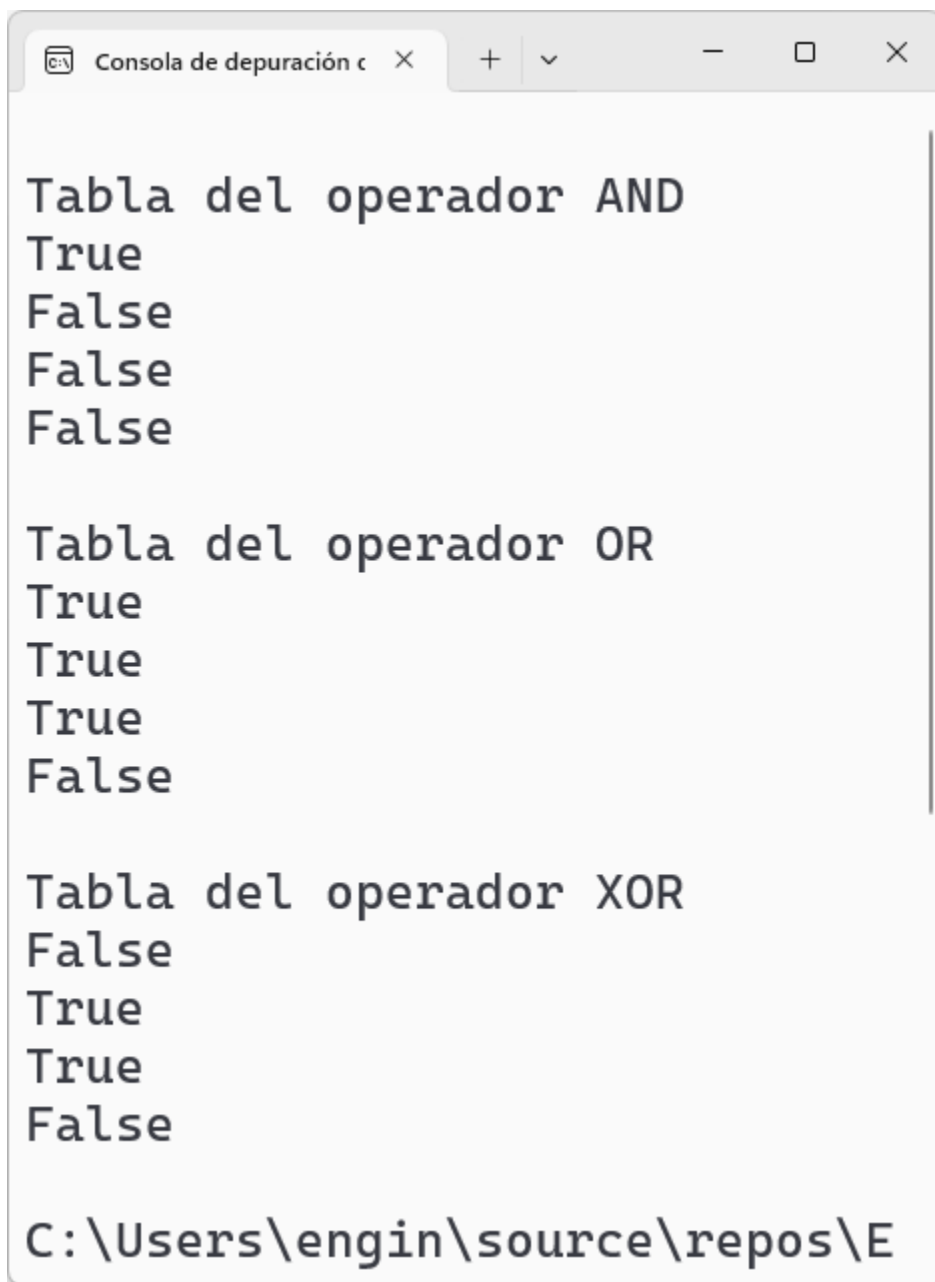


Ilustración 34: Operadores booleanos

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //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);
        }
    }
}
```



```
Consola de depuración c  X + v - □ X

Tabla del operador AND
True
False
False
False

Tabla del operador OR
True
True
True
False

Tabla del operador XOR
False
True
True
False

C:\Users\engin\source\repos\E
```

*Ilustración 35: Tablas de verdad*

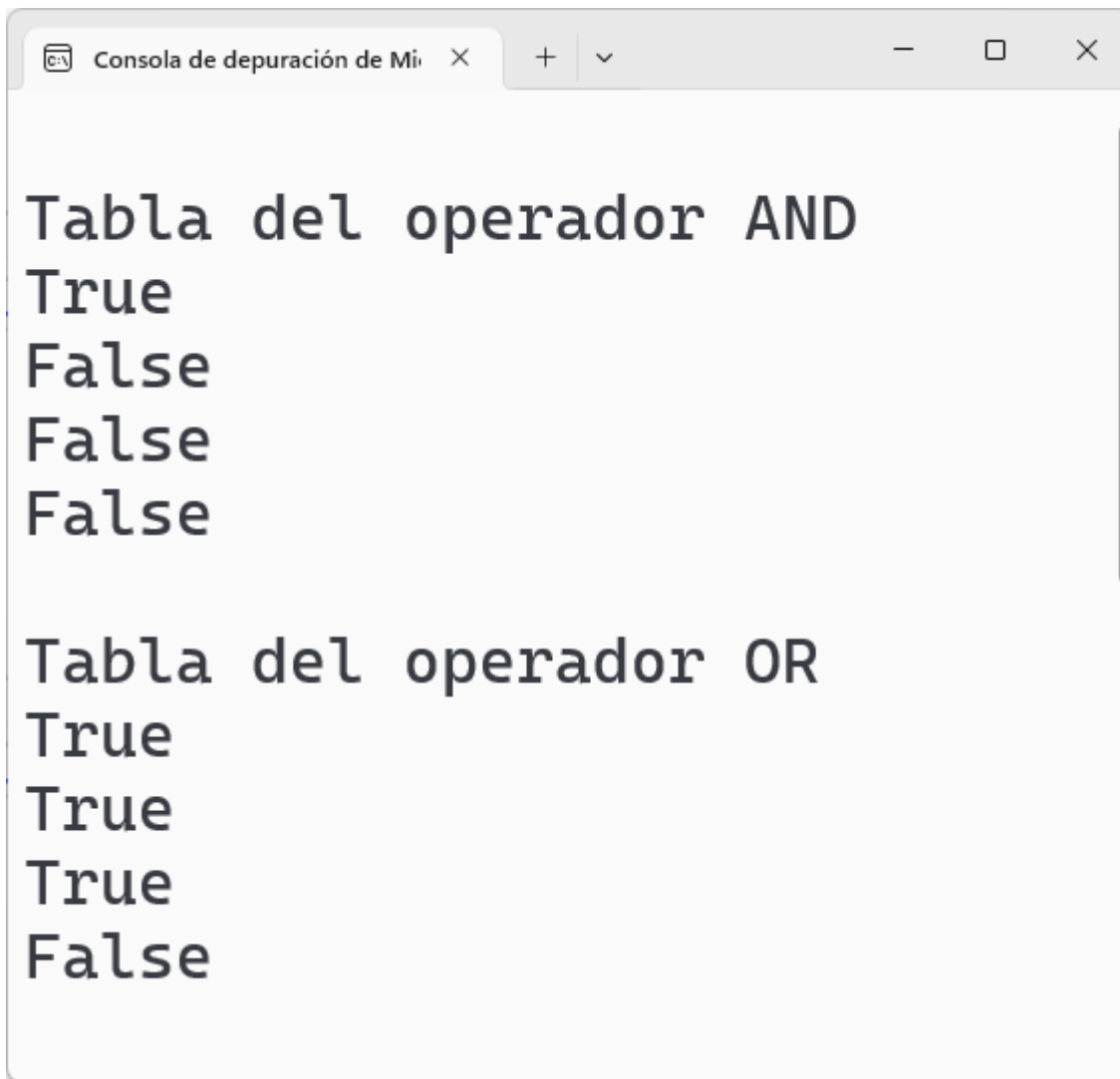


```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Tabla del AND
            Console.WriteLine("\r\nTabla del operador AND");
            Console.WriteLine(true && true);
            Console.WriteLine(true && false);
            Console.WriteLine(false && 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. */

        }
    }
}
```



The image shows a browser window with the developer console open. The console has a tab labeled 'Consola de depuración de Mi' and standard window controls. It displays two sections of text:

**Tabla del operador AND**  
True  
False  
False  
False

**Tabla del operador OR**  
True  
True  
True  
False

Ilustración 36: Tablas de verdad usando && y ||

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            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. */  
        }  
    }  
}
```

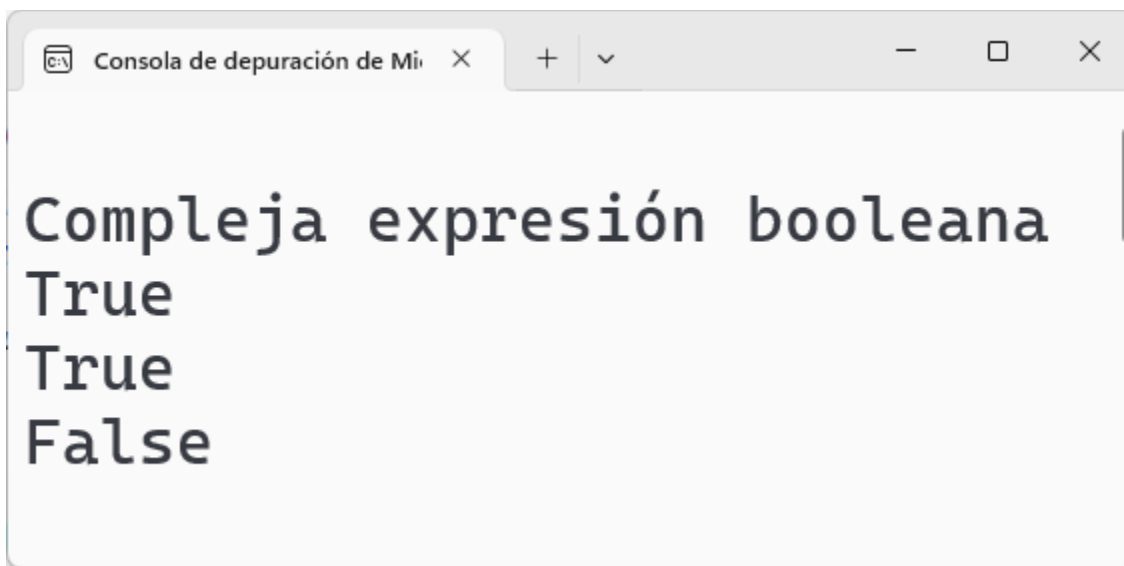


Ilustración 37: Expresión booleana

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            Console.WriteLine("Compleja 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 ||  
            */  
        }  
    }  
}
```

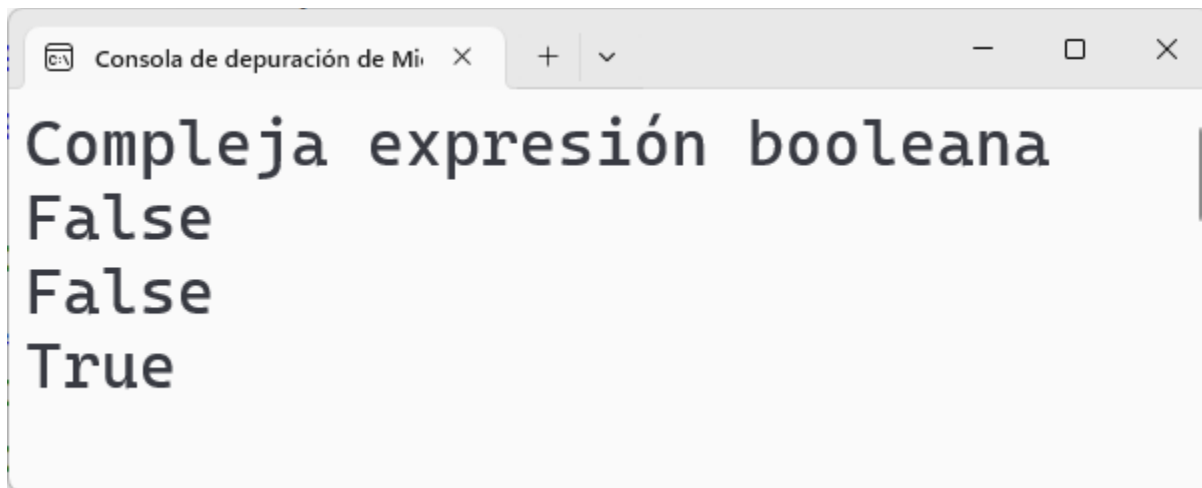


Ilustración 38: Precedencia de los operadores

## El operador "?", un si condicional

Usado en una asignación, funciona de esta manera:

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

A/040.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Operador ?  
            int valA = 10;  
            int valB = 13;  
            int Mayor = valA > valB ? valA: valB;  
            Console.WriteLine(Mayor);  
        }  
    }  
}
```

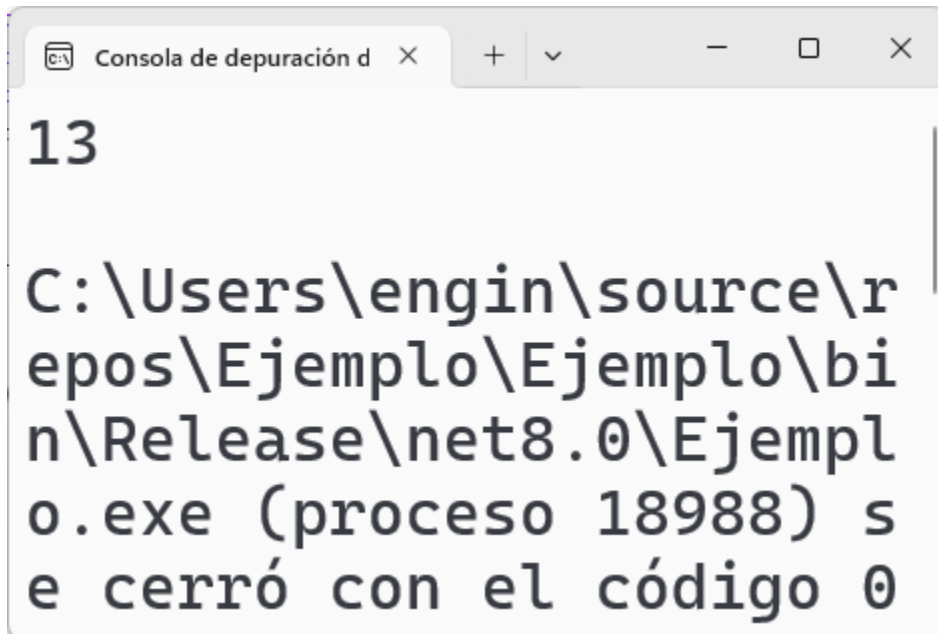


Ilustración 39: El operador "?", un si condicional

## Captura de error con try...catch

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

A/041.cs

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

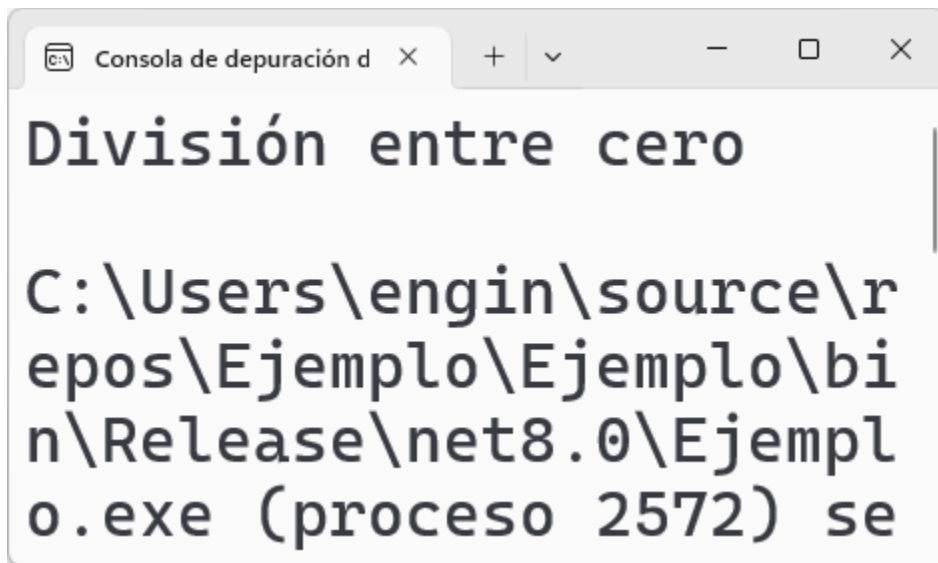


Ilustración 40: Captura de error con try...catch

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

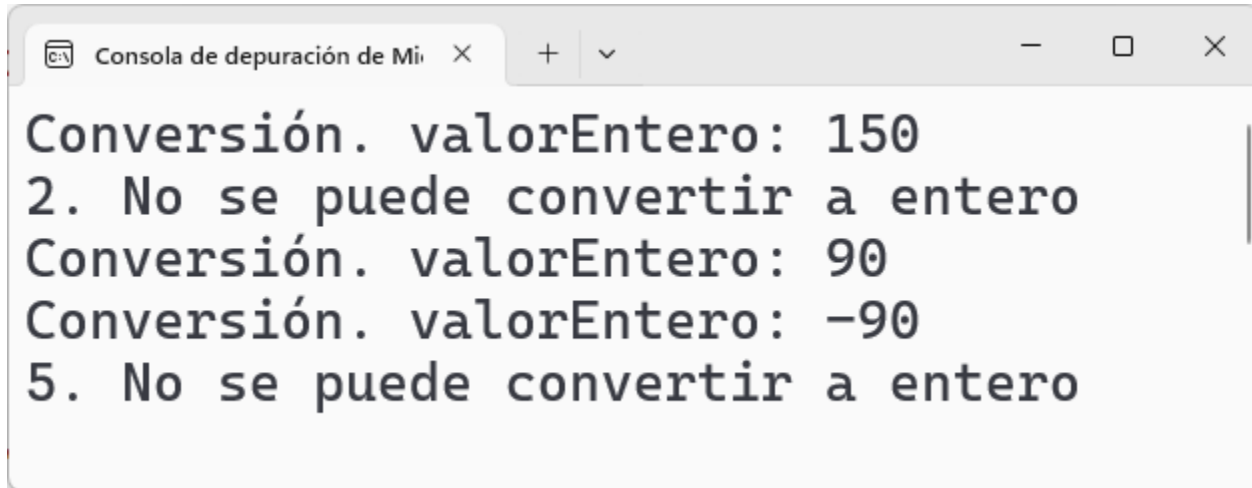
            //Segundo ejemplo
            Numero = "4.178"; //Un punto en la cadena
            if (Int32.TryParse(Numero, out valorEntero)) {
                Console.WriteLine("Conversión. valorEntero: " + valorEntero);
            }
            else {
                Console.WriteLine("2. No se puede convertir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

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

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

            //Quinto ejemplo
            Numero = "- 90"; //Espacio intermedio
            if (Int32.TryParse(Numero, out valorEntero)) {
                Console.WriteLine("Conversión. valorEntero: " + valorEntero);
            }
        }
    }
}
```

```
        else {  
            Console.WriteLine("5. No se puede convertir a entero");  
        }  
    }  
}
```



*Ilustración 41: Uso de TryParse para conversión*



# Ciclo for

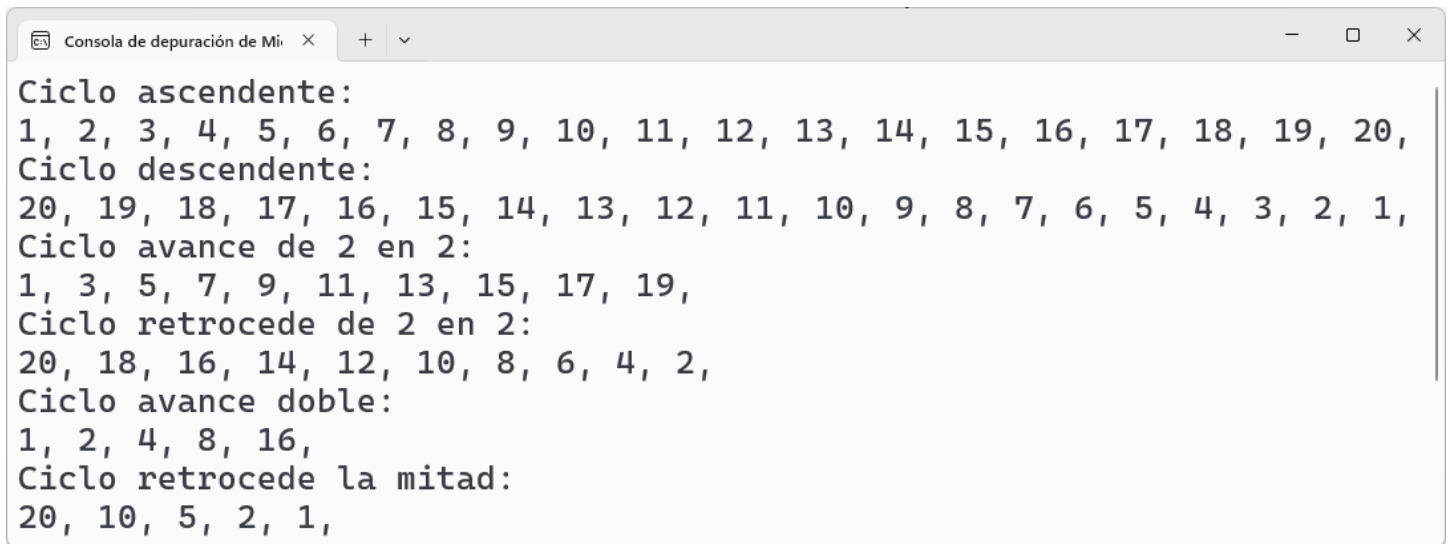
Su sintaxis es:

```
for (variable = valor inicio; condición para que se mantenga el ciclo;  
incremento/decremento de esa variable) { }
```

A/043.cs

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

}



```
Consola de depuración de Mi x + v  
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 avance de 2 en 2:  
1, 3, 5, 7, 9, 11, 13, 15, 17, 19,  
Ciclo retrocede de 2 en 2:  
20, 18, 16, 14, 12, 10, 8, 6, 4, 2,  
Ciclo avance doble:  
1, 2, 4, 8, 16,  
Ciclo retrocede la mitad:  
20, 10, 5, 2, 1,
```

Ilustración 42: Ciclo for

## Ciclo while

Su sintaxis es:

```
while (condición para que se mantenga en el ciclo) {  
}
```

A/044.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            int Contador;  
  
            //Ciclo while ascendente  
            Console.WriteLine("Ciclo ascendente:");  
            Contador = 1;  
            while (Contador <= 20) {  
                Console.Write(Contador + ", ");  
                Contador++;  
            }  
  
            //Ciclo while descendente  
            Console.WriteLine("\r\nCiclo descendente:");  
            Contador = 20;  
            while (Contador >= 1) {  
                Console.Write(Contador + ", ");  
                Contador--;  
            }  
  
            //Ciclo while ascendente, avance de 2 en 2  
            Console.WriteLine("\r\nCiclo avance de 2 en 2:");  
            Contador = 1;  
            while (Contador <= 20) {  
                Console.Write(Contador + ", ");  
                Contador += 2;  
            }  
  
            //Ciclo while descendente, retrocede de 2 en 2  
            Console.WriteLine("\r\nCiclo retrocede de 2 en 2:");  
            Contador = 20;  
            while (Contador >= 1) {  
                Console.Write(Contador + ", ");  
                Contador -= 2;  
            }  
  
            //Ciclo while ascendente, avance doble  
            Console.WriteLine("\r\nCiclo avance doble:");
```

```

Contador = 1;
while (Contador <= 20) {
    Console.Write(Contador + ", ");
    Contador *= 2;
}

//Ciclo while descendente, retrocede la mitad
Console.WriteLine("\r\nCiclo retrocede la mitad:");
Contador = 20;
while (Contador >= 1) {
    Console.Write(Contador + ", ");
    Contador /= 2;
}
}
}
}

```

```

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

```

Ilustración 43: Ciclo while

## Ciclo do...while

Su sintaxis es:

```
do {
```

```
} while (condición para que se mantenga en el ciclo)
```

A/045.cs

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            int Contador;

            //Ciclo do-while ascendente
            Console.WriteLine("Ciclo ascendente:");
            Contador = 1;
            do {
                Console.Write(Contador + ", ");
                Contador++;
            } while (Contador <= 20);

            //Ciclo do-while descendente
            Console.WriteLine("\r\nCiclo descendente:");
            Contador = 20;
            do {
                Console.Write(Contador + ", ");
                Contador--;
            } while (Contador >= 1);

            //Ciclo do-while ascendente, avance de 2 en 2
            Console.WriteLine("\r\nCiclo avance de 2 en 2:");
            Contador = 1;
            do {
                Console.Write(Contador + ", ");
                Contador += 2;
            } while (Contador <= 20);

            //Ciclo do-while descendente, retrocede de 2 en 2
            Console.WriteLine("\r\nCiclo retrocede de 2 en 2:");
            Contador = 20;
            do {
                Console.Write(Contador + ", ");
                Contador -= 2;
            } while (Contador >= 1);

            //Ciclo do-while ascendente, avance doble
```

```

        Console.WriteLine("\r\nCiclo avance doble:");
        Contador = 1;
        do {
            Console.Write(Contador + ", ");
            Contador *= 2;
        } while (Contador <= 20);

        //Ciclo do-while descendente, retrocede la mitad
        Console.WriteLine("\r\nCiclo retrocede la mitad:");
        Contador = 20;
        do {
            Console.Write(Contador + ", ");
            Contador /= 2;
        } while (Contador >= 1);
    }
}

```

```

Consola de depuración de Mi
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 avance de 2 en 2:
1, 3, 5, 7, 9, 11, 13, 15, 17, 19,
Ciclo retrocede de 2 en 2:
20, 18, 16, 14, 12, 10, 8, 6, 4, 2,
Ciclo avance doble:
1, 2, 4, 8, 16,
Ciclo retrocede la mitad:
20, 10, 5, 2, 1,

```

Ilustración 44: Ciclo do...while

## Romper ciclo con break

A/046.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            int Contador;  
  
            //Rompe el ciclo con break  
            Console.WriteLine("Ciclo ascendente:");  
            Contador = 1;  
            do {  
                //Si Contador es múltiplo de 13  
                //se rompe el ciclo con break  
                if (Contador % 13 == 0) break;  
                Console.Write(Contador + ", ");  
                Contador++;  
            } while (Contador <= 20);  
        }  
    }  
}
```

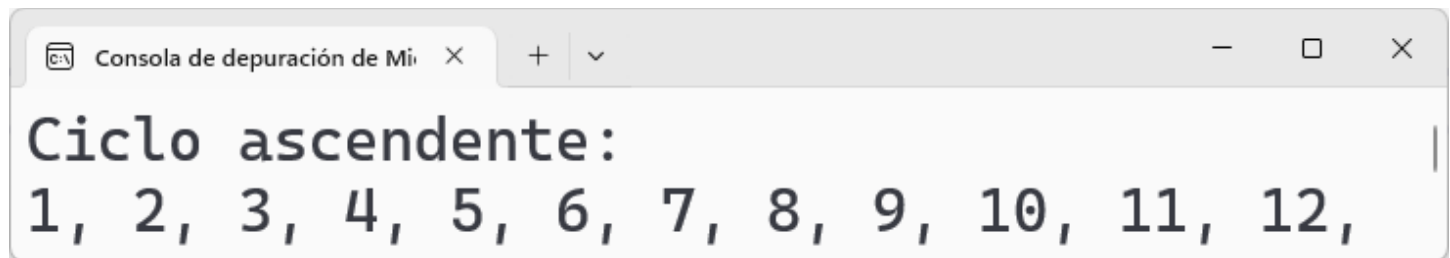


Ilustración 45: Romper ciclo con break

## Uso del continue en ciclos

A/047.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            int Contador;  
  
            Console.WriteLine("Ciclo ascendente:");  
            Contador = 0;  
            do {  
                Contador++;  
  
                //Si Contador es par entonces va  
                //a la siguiente iteración, no  
                //ejecuta lo que está después.  
                if (Contador % 2 == 0) continue;  
  
                Console.Write(Contador + ", ");  
            } while (Contador <= 20);  
        }  
    }  
}
```



Ilustración 46: Uso del continue en ciclos



```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
  
            //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 + ":" + segundo);  
                }  
            }  
        }  
    }  
}
```

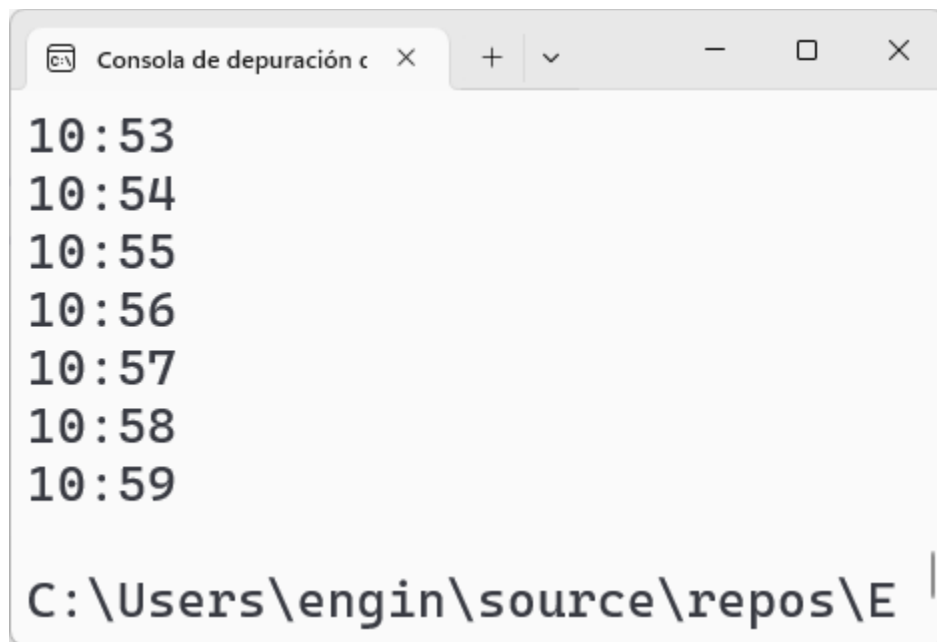
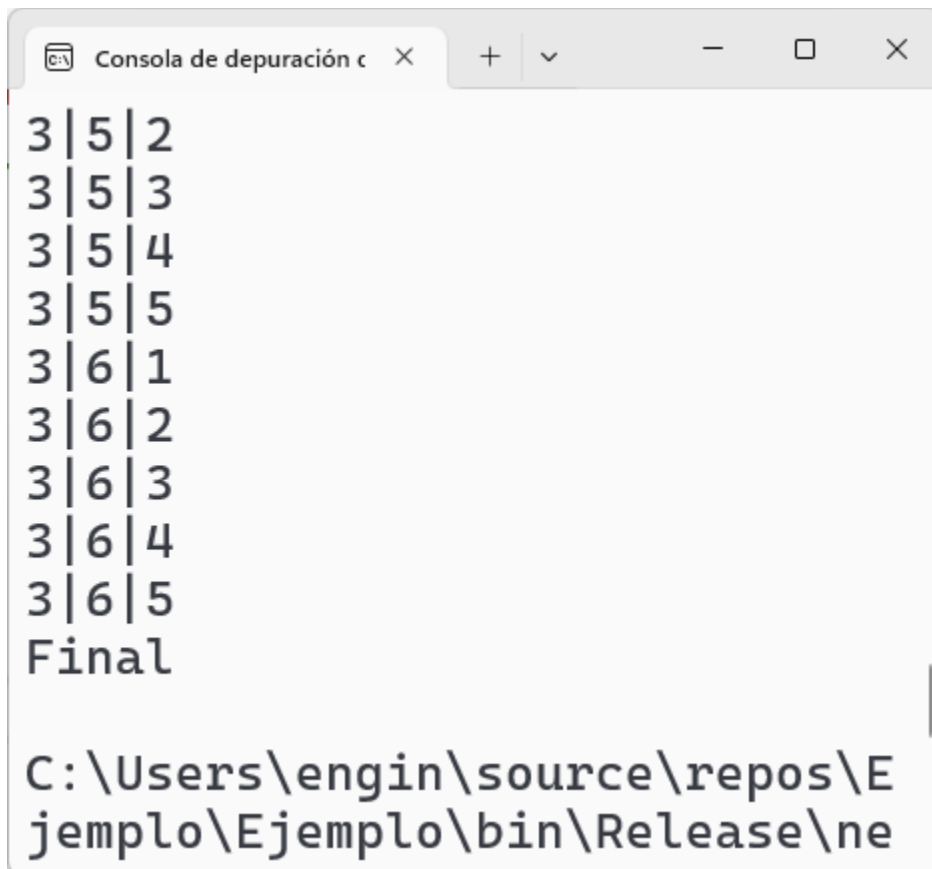


Ilustración 47: Ciclos anidados

## Rompiendo ciclos anidados

A/049.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
  
            for (int numA = 1; numA <= 3; numA++) {  
                for (int numB = 1; numB <= 6; numB++) {  
                    for (int numC = 1; numC <= 9; numC++) {  
                        Console.Write(numA + "|");  
                        Console.WriteLine(numB + "|" + numC);  
                        //Sólo rompe el ciclo más interno  
                        if (numC == 5) break;  
                    }  
                }  
            }  
            Console.WriteLine("Final");  
        }  
    }  
}
```



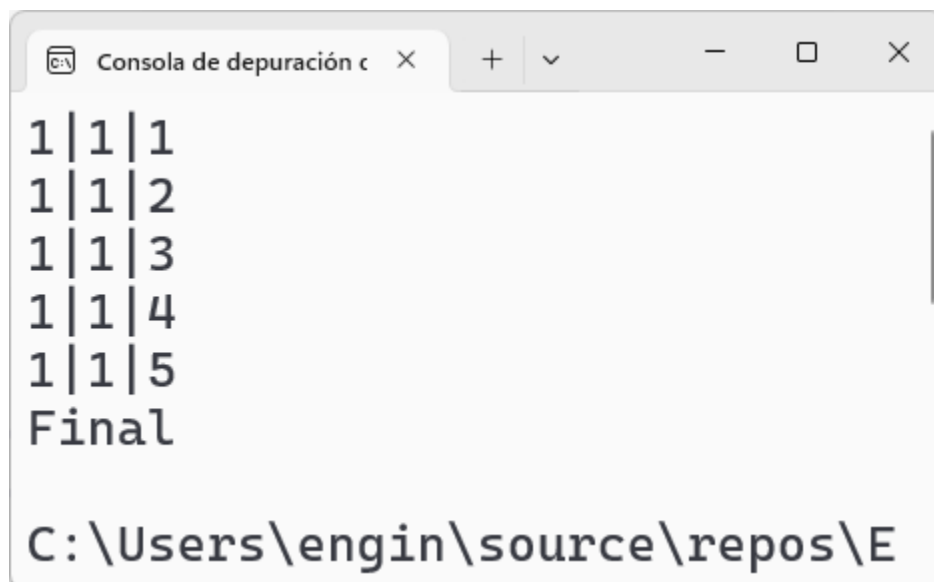
```
Consola de depuración c  X  +  v  -  □  X  
3|5|2  
3|5|3  
3|5|4  
3|5|5  
3|6|1  
3|6|2  
3|6|3  
3|6|4  
3|6|5  
Final  
  
C:\Users\engin\source\repos\Ejemplo\Ejemplo\bin\Release\ne
```

Ilustración 48: Rompiendo ciclos anidados

## Uso del goto

A/050.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
  
            for (int numA = 1; numA <= 3; numA++) {  
                for (int numB = 1; numB <= 6; numB++) {  
                    for (int numC = 1; numC <= 9; numC++) {  
                        Console.Write(numA + "|");  
                        Console.WriteLine(numB + "|" + numC);  
                        //Sale de todos los ciclos  
                        if (numC == 5) goto afuera;  
                    }  
                }  
            }  
  
            afuera: Console.WriteLine("Final");  
        }  
    }  
}
```



```
1|1|1  
1|1|2  
1|1|3  
1|1|4  
1|1|5  
Final  
  
C:\Users\engin\source\repos\E
```

Ilustración 49: Uso del goto

## Diferencias de precisión entre float y double

A/051.cs

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //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 + " y " + acumB);
        }
    }
}
```

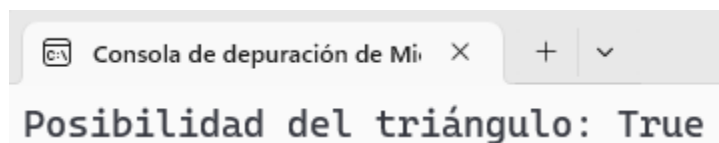


Ilustración 50: Diferencias de precisión entre float y double

Muy recomendado: usar double.

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Uso de funciones
            double ladoA = 3;
            double ladoB = 4;
            double ladoC = 5;
            bool Posible = TrianguloPosible(ladoA, ladoB, ladoC);
            Console.WriteLine("Posibilidad del triángulo: " + Posible);
        }

        //Retorna true si el triángulo es posible
        static bool TrianguloPosible(double valA, double valB, double valC) {
            return valA + valB >= valC &&
                valA + valC >= valB &&
                valB + valC >= valA;
        }
    }
}
```



*Ilustración 51: Función que retorna dato de tipo bool*

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

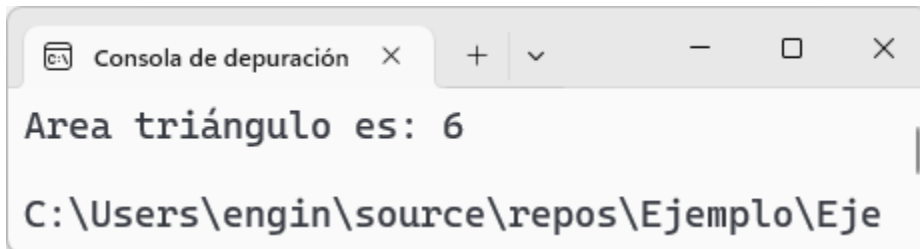
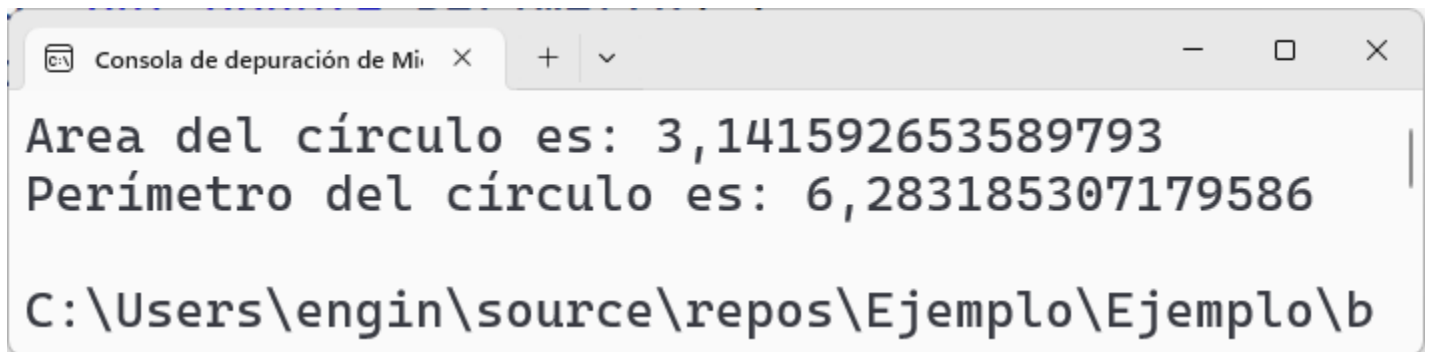


Ilustración 52: Funciones

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //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);  
            Console.WriteLine("Perímetro del círculo es: " + perimetro);  
        }  
  
        //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;  
        }  
    }  
}
```

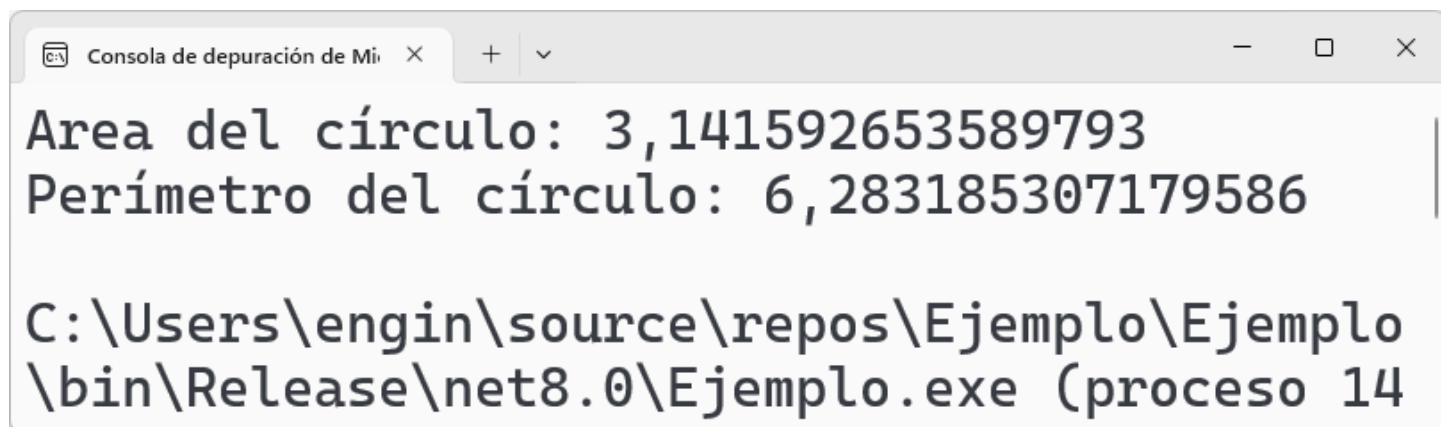


Consola de depuración de Mi

Area del círculo es: 3,141592653589793  
Perímetro del círculo es: 6,283185307179586  
C:\Users\engin\source\repos\Ejemplo\Ejemplo\b

Ilustración 53: Funciones con doble salida

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Función con doble salida. Nuevo en C# 7.0  
            double radio = 1;  
            double perimetro, area;  
            Circulo(radio, out perimetro, out area);  
            Console.WriteLine("Area del círculo: " + area);  
            Console.WriteLine("Perímetro del círculo: " + perimetro);  
        }  
  
        //Retorna dos valores: el área y el perímetro del círculo  
        static void Circulo(double R, out double Perimetro, out double Area) {  
            Area = Math.PI * R * R;  
            Perimetro = 2 * Math.PI * R;  
        }  
    }  
}
```



Consola de depuración de Mi

Area del círculo: 3,141592653589793  
Perímetro del círculo: 6,283185307179586

C:\Users\engin\source\repos\Ejemplo\Ejemplo  
\bin\Release\net8.0\Ejemplo.exe (proceso 14)

Ilustración 54: Funciones con doble salida



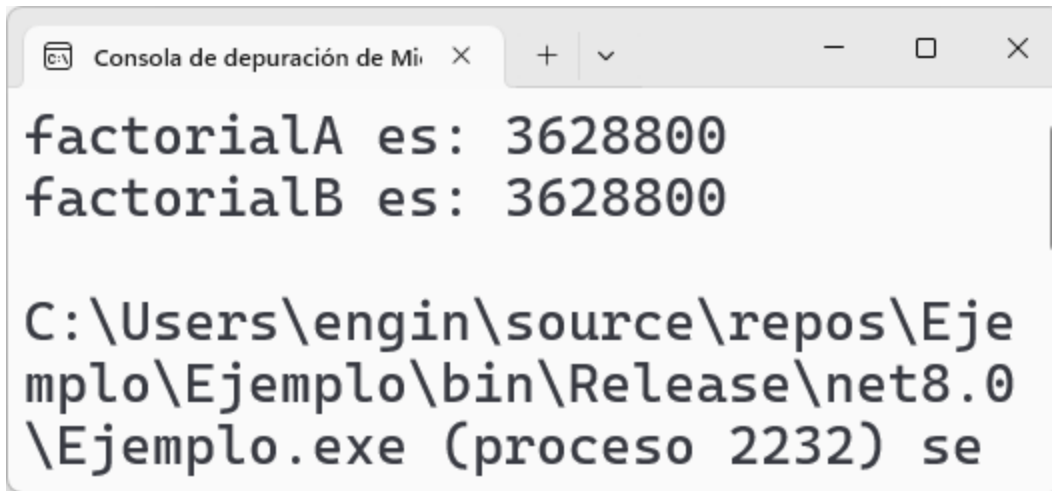
```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Función iterativa y recursiva
            long valor = 10;
            long factorialA, factorialB;

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

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

        //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);
        }
    }
}
```



A screenshot of a Windows Debug Console window. The title bar reads 'Consola de depuración de Mi' followed by a close button. The window contains two lines of output: 'factorialA es: 3628800' and 'factorialB es: 3628800'. Below these, a file path is displayed: 'C:\Users\engin\source\repos\Ejemplo\Ejemplo\bin\Release\net8.0\Ejemplo.exe (proceso 2232) se'. The text is in a monospaced font.

```
factorialA es: 3628800
factorialB es: 3628800

C:\Users\engin\source\repos\Ejemplo\Ejemplo\bin\Release\net8.0
\Ejemplo.exe (proceso 2232) se
```

*Ilustración 55: Funciones iterativas y recursivas*

```
namespace Ejemplo {  
    internal class Program {  
        //Una variable que es conocida por todas las funciones  
        static int valor;  
  
        static void Main() {  
            //Ámbito de las variables  
            valor = 17;  
            Console.WriteLine("Valor al iniciar: " + valor);  
            SubrutinaA();  
            Console.WriteLine("Después de SubrutinaA(): " + valor);  
            SubrutinaB();  
            Console.WriteLine("Después de SubrutinaB(): " + valor);  
        }  
  
        //Un procedimiento  
        static void SubrutinaA() {  
            valor = 590;  
        }  
  
        //Otro procedimiento  
        static void SubrutinaB() {  
            valor = 3246;  
        }  
    }  
}
```

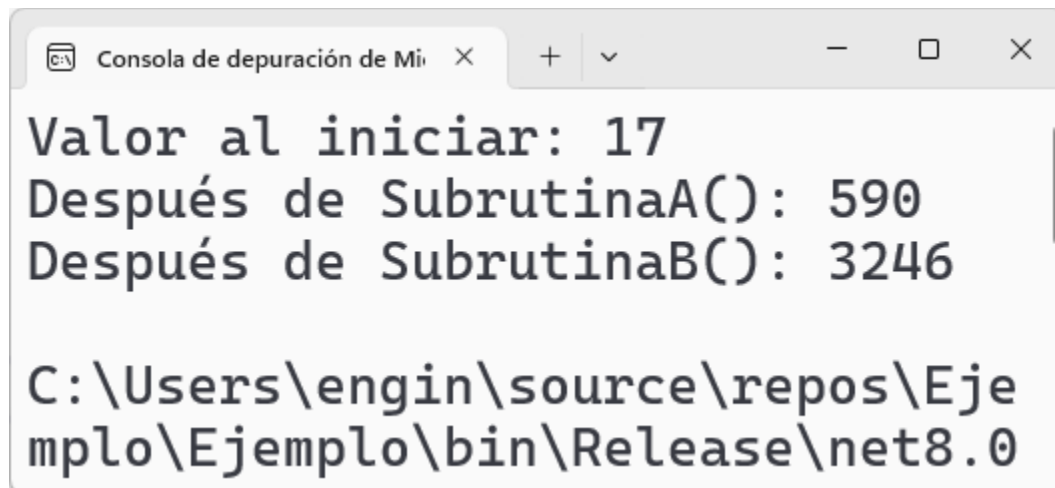


Ilustración 56: Ámbito de las variables

```

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Ámbito de las variables. Variable local
            int valor = 17;
            Console.WriteLine("Valor inicial: " + valor);
            SubrutinaA();
            Console.WriteLine("Después de SubrutinaA(): " + valor);
            SubrutinaB();
            Console.WriteLine("Después de SubrutinaB(): " + valor);
        }

        //Un procedimiento.
        //La variable "valor" es otra distinta a la del "Main"
        static void SubrutinaA() {
            int valor = 590;
        }

        //Otro procedimiento. La variable
        //"valor" es otra distinta a la del "Main"
        static void SubrutinaB() {
            int valor = 3246;
        }
    }
}

```

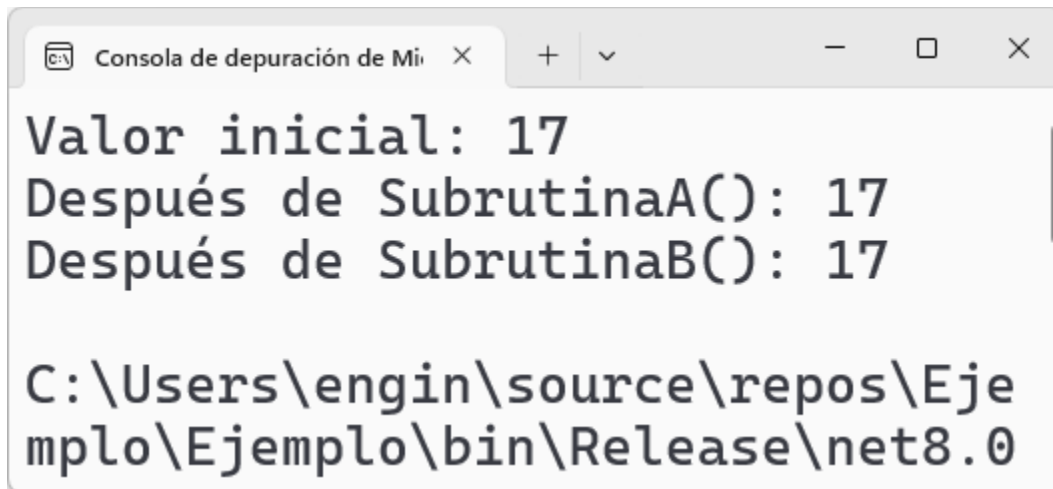


Ilustración 57: Ámbito de las variables

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Funciones de números
            int Numero = 16832929;
            Console.Write("Número: ");
            Console.WriteLine(Numero);

            Console.Write("Impar: ");
            Console.WriteLine(EsImpar(Numero));

            Console.Write("Par: ");
            Console.WriteLine(EsPar(Numero));

            Console.Write("Total de cifras: ");
            Console.WriteLine(TotalCifras(Numero));

            Console.Write("Dos últimas cifras: ");
            Console.WriteLine(RetornadosultimasCifras(Numero));

            Console.Write("Antepenúltima cifra: ");
            Console.WriteLine(AntepenultimaCifra(Numero));

            Console.Write("Penúltima cifra: ");
            Console.WriteLine(PenultimaCifra(Numero));

            Console.Write("Última cifra: ");
            Console.WriteLine(UltimaCifra(Numero));

            Console.Write("Cifra más alta: ");
            Console.WriteLine(LaCifraMasAlta(Numero));

            Console.Write("Cifra más baja: ");
            Console.WriteLine(LaCifraMasBaja(Numero));

            Console.Write("Total de cifras iguales a 5 es: ");
            Console.WriteLine(Cifrashalladas(Numero, 5));

            Console.Write("Al invertirlo es: ");
            Console.WriteLine(InvierteNumero(Numero));

            Console.Write("Es palíndromo: ");
            Console.WriteLine(EsPalindromo(Numero));

            Console.Write("Tercera cifra es: ");
```

```
Console.WriteLine(CifraPosicion(Numero, 3));

Console.Write("Primera cifra es: ");
Console.WriteLine(PrimeraCifra(Numero));

Console.Write("Suma de las cifras es: ");
Console.WriteLine(SumaCifras(Numero));

Console.Write("Suma de cifras pares es: ");
Console.WriteLine(SumaCifrasPares(Numero));

Console.Write("Suma de cifras impares es: ");
Console.WriteLine(SumaCifrasImpares(Numero));

Console.Write("Multiplicación de cifras es: ");
Console.WriteLine(MultiplicaCifras(Numero));

Console.Write("Multiplicación de cifras pares es: ");
Console.WriteLine(MultiplicaCifrasPares(Numero));

Console.Write("Multiplicación de cifras impares es: ");
Console.WriteLine(MultiplicaCifrasImpares(Numero));

Console.Write("Todas las cifras son pares: ");
Console.WriteLine(TodasCifrasPares(Numero));

Console.Write("Todas las cifras son impares: ");
Console.WriteLine(TodasCifrasImpares(Numero));

Console.Write("Total cifras pares: ");
Console.WriteLine(TotalCifrasPares(Numero));

Console.Write("Total cifras impares: ");
Console.WriteLine(TotalCifrasImpares(Numero));

Console.Write("Solo hay cifras menores o iguales a 5: ");
Console.WriteLine(SoloCifrasMenorIgual(Numero, 5));

Console.Write("Solo hay cifras mayores o iguales a 5: ");
Console.WriteLine(SoloCifrasMayorIgual(Numero, 5));

Console.Write("Usa distintas cifras: ");
Console.WriteLine(DistintasCifras(Numero));

Console.Write("Usa distintas cifras pares: ");
Console.WriteLine(DistintasCifrasPares(Numero));

Console.Write("Usa distintas cifras impares: ");
```

```

        Console.WriteLine(DistintasCifrasImpares(Numero));

        Console.Write("Extrayendo cifras pares: ");
        Console.WriteLine(NumeroSoloPares(Numero));

        Console.Write("Extrayendo cifras impares: ");
        Console.WriteLine(NumeroSoloImpares(Numero));
    }

    // Retorna true si un número es impar
    static bool EsImpar(int Numero) {
        return Numero % 2 == 1;
    }

    // Retorna true si un número es par
    static bool EsPar(int Numero) {
        return Numero % 2 == 0;
    }

    // Retorna el número de Cifras de un número
    static int TotalCifras(int Numero) {
        int Copia = Numero;
        int Cuenta = 0;
        while (Copia != 0) {
            Cuenta++;
            Copia /= 10;
        }
        return Cuenta;
    }

    // Retorna las dos últimas Cifras del número
    static int RetornadosultimasCifras(int Numero) {
        return Numero % 100;
    }

    // Retorna la antepenúltima Cifra de un número entero
    static int AntepenultimaCifra(int Numero) {
        return (Numero / 100) % 10;
    }

    // Retorna la penúltima Cifra de un número entero
    static int PenultimaCifra(int Numero) {
        return (Numero / 10) % 10;
    }

    // Retorna la última Cifra de un número entero
    static int UltimaCifra(int Numero) {
        return Numero % 10;
    }

```

```

}

// Retorna la Cifra más alta
static int LaCifraMasAlta(int Numero) {
    int Copia = Numero;
    int Cifra = 0;
    while (Copia != 0) {
        if (Copia % 10 > Cifra) Cifra = Copia % 10;
        Copia /= 10;
    }
    return Cifra;
}

// Retorna la Cifra más baja
static int LaCifraMasBaja(int Numero) {
    if (Numero == 0) return 0;
    int Copia = Numero;
    int Cifra = 9;
    while (Copia != 0) {
        if (Copia % 10 < Cifra) Cifra = Copia % 10;
        Copia /= 10;
    }
    return Cifra;
}

// Dice cuántas Cifras de determinado número hay en el número enviado
static int Cifrashalladas(int Numero, int Cifra) {
    int Copia = Numero;
    int Acumula = 0;
    while (Copia != 0) {
        if (Copia % 10 == Cifra) Acumula++;
        Copia /= 10;
    }
    return Acumula;
}

// Invierte un número
static int InvierteNumero(int Numero) {
    int Copia = Numero;
    int Multiplica = (int)Math.Pow(10, TotalCifras(Copia) - 1);
    int Acumula = 0;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        Acumula += Cifra * Multiplica;
        Multiplica /= 10;
        Copia /= 10;
    }
    return Acumula;
}

```



```

}

// Retorna true si el número es palíndromo
static bool EsPalindromo(int Numero) {
    if (Numero == InvierteNumero(Numero)) return true;
    return false;
}

//Retorna la Cifra de una determinada posición
static int CifraPosicion(int Numero, int Posicion) {
    int Copia = InvierteNumero(Numero);
    int Pos = 1;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        if (Pos == Posicion) return Cifra;
        Copia /= 10;
        Pos++;
    }
    return 0;
}

// Retorna la primera Cifra de un número
static int PrimeraCifra(int Numero) {
    return CifraPosicion(Numero, 1);
}

// Retorna la sumatoria de las Cifras de un número
static int SumaCifras(int Numero) {
    int Copia = Numero;
    int Acumula = 0;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        Acumula += Cifra;
        Copia /= 10;
    }
    return Acumula;
}

// Retorna la sumatoria de las Cifras pares de un número
static int SumaCifrasPares(int Numero) {
    int Copia = Numero;
    int Acumula = 0;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        if (Cifra % 2 == 0) Acumula += Cifra;
        Copia /= 10;
    }
    return Acumula;
}

```

```

}

// Retorna la sumatoria de las Cifras impares de un número
static int SumaCifrasImpares(int Numero) {
    int Copia = Numero;
    int Acumula = 0;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        if (Cifra % 2 != 0) Acumula += Cifra;
        Copia /= 10;
    }
    return Acumula;
}

// Retorna el producto de las Cifras de un número
static int MultiplicaCifras(int Numero) {
    if (Numero == 0) return 0;
    int Copia = Numero;
    int Acumula = 1;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        Acumula *= Cifra;
        Copia /= 10;
    }
    return Acumula;
}

// Retorna el producto de las Cifras pares de un número
static int MultiplicaCifrasPares(int Numero) {
    int Copia = Numero;
    int Acumula = 1;
    bool HayPar = false;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        if (Cifra % 2 == 0) {
            Acumula *= Cifra;
            HayPar = true;
        }
        Copia /= 10;
    }
    if (HayPar) return Acumula;
    return 0;
}

// Retorna el producto de las Cifras impares de un número
static int MultiplicaCifrasImpares(int Numero) {
    int Copia = Numero;
    int Acumula = 1;

```

```

    bool HayImpar = false;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        if (Cifra % 2 != 0) {
            Acumula *= Cifra;
            HayImpar = true;
        }
        Copia /= 10;
    }
    if (HayImpar) return Acumula;
    return 0;
}

// Retorna true si todas las Cifras son pares
static bool TodasCifrasPares(int Numero) {
    int Copia = Numero;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        if (Cifra % 2 != 0) return false;
        Copia /= 10;
    }
    return true;
}

// Retorna true si todas las Cifras son impares
static bool TodasCifrasImpares(int Numero) {
    int Copia = Numero;
    if (Copia == 0) return false;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        if (Cifra % 2 == 0) return false;
        Copia /= 10;
    }
    return true;
}

// Retorna el número de Cifras pares
static int TotalCifrasPares(int Numero) {
    int Copia = Numero;
    int Acumula = 0;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        if (Cifra % 2 == 0) Acumula++;
        Copia /= 10;
    }
    return Acumula;
}

```

```

// Retorna el número de Cifras impares
static int TotalCifrasImpares(int Numero) {
    int Copia = Numero;
    int Acumula = 0;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        if (Cifra % 2 != 0) Acumula++;
        Copia /= 10;
    }
    return Acumula;
}

// Retorna true si el número tiene sólo Cifras menores o iguales a Cifra
static bool SoloCifrasMenorIgual(int Numero, int Cifra) {
    int Copia = Numero;
    while (Copia != 0) {
        if (Copia % 10 > Cifra) return false;
        Copia /= 10;
    }
    return true;
}

// Retorna true si el número tiene sólo Cifras mayores o iguales a Cifra
static bool SoloCifrasMayorIgual(int Numero, int Cifra) {
    int Copia = Numero;
    while (Copia != 0) {
        if (Copia % 10 < Cifra) return false;
        Copia /= 10;
    }
    return true;
}

// Retorna true si todas las Cifras son distintas
static bool DistintasCifras(int Numero) {
    for (int Cifra = 0; Cifra <= 9; Cifra++) {
        int Copia = Numero;
        int Cuenta = 0;
        while (Copia != 0) {
            if (Copia % 10 == Cifra) Cuenta++;
            if (Cuenta > 1) return false;
            Copia /= 10;
        }
    }
    return true;
}

// Retorna si todas las Cifras pares son distintas
static bool DistintasCifrasPares(int Numero) {

```

```

    for (int Cifra = 0; Cifra <= 8; Cifra += 2) {
        int Copia = Numero;
        int Cuenta = 0;
        while (Copia != 0) {
            if (Copia % 10 == Cifra) Cuenta++;
            if (Cuenta > 1) return false;
            Copia /= 10;
        }
    }
    return true;
}

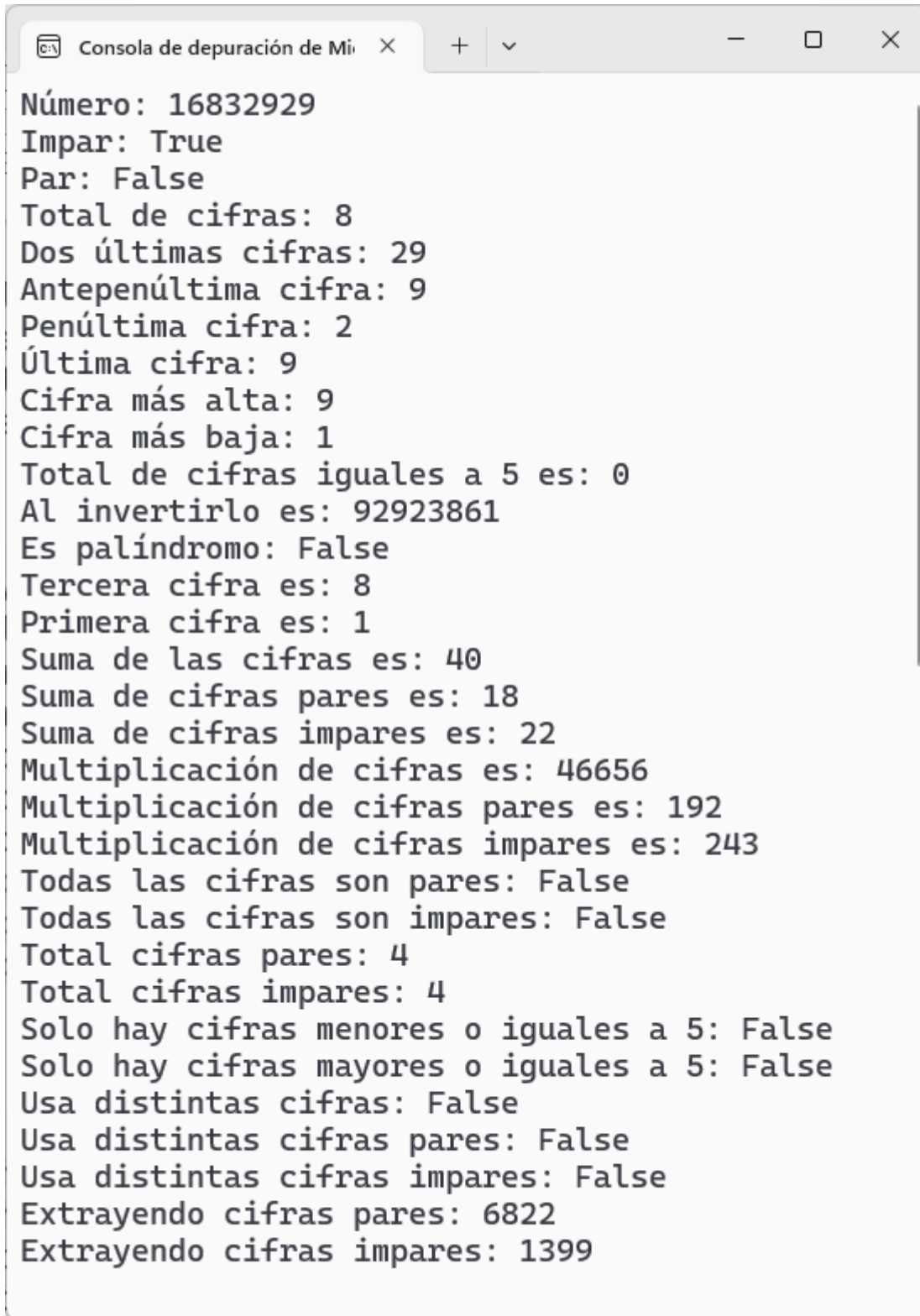
// Retorna si todas las Cifras impares son distintas
static bool DistintasCifrasImPares(int Numero) {
    for (int Cifra = 1; Cifra <= 9; Cifra += 2) {
        int Copia = Numero;
        int Cuenta = 0;
        while (Copia != 0) {
            if (Copia % 10 == Cifra) Cuenta++;
            if (Cuenta > 1) return false;
            Copia /= 10;
        }
    }
    return true;
}

//Retorna un número con solo las Cifras pares
static int NumeroSoloPares(int Numero) {
    int Copia = Numero;
    int Acumula = 0;
    int Posicion = 1;
    while (Copia != 0) {
        int Cifra = Copia % 10;
        if (Cifra % 2 == 0) {
            Acumula += Posicion * Cifra;
            Posicion *= 10;
        }
        Copia /= 10;
    }
    return Acumula;
}

//Retorna un número con solo las Cifras impares
static int NumeroSoloImpares(int Numero) {
    int Copia = Numero;
    int Acumula = 0;
    int Posicion = 1;
    while (Copia != 0) {

```

```
        int Cifra = Copia % 10;
        if (Cifra % 2 != 0) {
            Acumula += Posicion * Cifra;
            Posicion *= 10;
        }
        Copia /= 10;
    }
    return Acumula;
}
}
```



```
Número: 16832929
Impar: True
Par: False
Total de cifras: 8
Dos últimas cifras: 29
Antepenúltima cifra: 9
Penúltima cifra: 2
Última cifra: 9
Cifra más alta: 9
Cifra más baja: 1
Total de cifras iguales a 5 es: 0
Al invertirlo es: 92923861
Es palíndromo: False
Tercera cifra es: 8
Primera cifra es: 1
Suma de las cifras es: 40
Suma de cifras pares es: 18
Suma de cifras impares es: 22
Multiplicación de cifras es: 46656
Multiplicación de cifras pares es: 192
Multiplicación de cifras impares es: 243
Todas las cifras son pares: False
Todas las cifras son impares: False
Total cifras pares: 4
Total cifras impares: 4
Solo hay cifras menores o iguales a 5: False
Solo hay cifras mayores o iguales a 5: False
Usa distintas cifras: False
Usa distintas cifras pares: False
Usa distintas cifras impares: False
Extrayendo cifras pares: 6822
Extrayendo cifras impares: 1399
```

Ilustración 58: Manejo de cifras