

# C# Y .NET 8

## Parte 02. Cadenas

2024-07

Rafael Alberto Moreno Parra  
ramsoftware@gmail.com

## Contenido

Tabla de ilustraciones.....	3
Acerca del autor.....	4
Licencia de este libro .....	4
Licencia del software .....	4
Marcas registradas .....	5
Dedicatoria .....	6
Declaración de cadenas.....	7
Uso de @ en cadenas.....	8
Constantes con cadenas.....	9
Copia de cadenas .....	10
Caracteres especiales .....	11
Acceder a un determinado carácter .....	12
Tamaño de la cadena y recorrerla .....	13
Subcadenas .....	14
Reemplazar caracteres.....	15
Encontrar subcadenas o caracteres .....	16
Convertir a mayúsculas y minúsculas .....	18
StringBuilder, más veloz y se puede modificar su contenido.....	19
StringBuilder, métricas de velocidad .....	20
Eliminar caracteres al inicio y al final .....	22
Comparar cadenas. Forma no recomendada .....	24
Comparar cadenas. Forma recomendada.....	26
Comparar cadenas. Ignorando las mayúsculas y minúsculas.....	28

## Tabla de ilustraciones

Ilustración 1: Declaración de cadenas.....	7
Ilustración 2: Uso de @.....	8
Ilustración 3: No se puede cambiar una constante.....	9
Ilustración 4: Copia de cadenas.....	10
Ilustración 5: Caracteres especiales .....	11
Ilustración 6: Acceder a un determinado carácter.....	12
Ilustración 7: Tamaño de la cadena y recorrerla .....	13
Ilustración 8: Subcadenas .....	14
Ilustración 9: Reemplazar caracteres.....	15
Ilustración 10: Encontrar subcadenas o caracteres .....	17
Ilustración 11: Convertir a mayúsculas y minúsculas.....	18
Ilustración 12: StringBuilder, más veloz y se puede modificar su contenido.....	19
Ilustración 13: StringBuilder, métricas de velocidad .....	21
Ilustración 14: Eliminar caracteres al inicio y al final .....	23
Ilustración 15: Comparar cadenas. Forma no recomendada.....	25
Ilustración 16: Comparar cadenas. Forma recomendada .....	27
Ilustración 17: Comparar cadenas. Ignorando las mayúsculas y minúsculas.....	29

## Acerca del autor

Rafael Alberto Moreno Parra

[ramsoftware@gmail.com](mailto:ramsoftware@gmail.com) o [enginelifelife@hotmail.com](mailto:enginelifelife@hotmail.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.

## Declaración de cadenas

En C# hay un rico conjunto de funciones para manipular cadenas (strings), incluyendo una específica para mejorar el desempeño: `StringBuilder`

B/001.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Declarar variable de tipo cadena (string). Dos formas.  
            System.String cadenaA;  
            string cadenaB;  
  
            //Asignando valores de cadena  
            cadenaA = "Esto es una prueba de texto";  
            cadenaB = "Asignando valores alfanuméricos";  
  
            //Imprimiendo por consola  
            Console.WriteLine(cadenaA);  
            Console.WriteLine(cadenaB);  
  
            //Uniando dos cadenas o concatenar  
            string cadenaC = cadenaA + " <=0=> " + cadenaB;  
            Console.WriteLine(cadenaC);  
        }  
    }  
}
```

Ilustración 1: Declaración de cadenas

## Uso de @ en cadenas

B/002.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Rutas. Forma antigua  
            string RutaA = "C:\\Users\\engin\\OneDrive\\";  
  
            //Ruta. Forma moderna  
            string RutaB = @"C:\Users\engin\OneDrive\";  
  
            //Imprimiendo por consola  
            Console.WriteLine(RutaA);  
            Console.WriteLine(RutaB);  
        }  
    }  
}
```

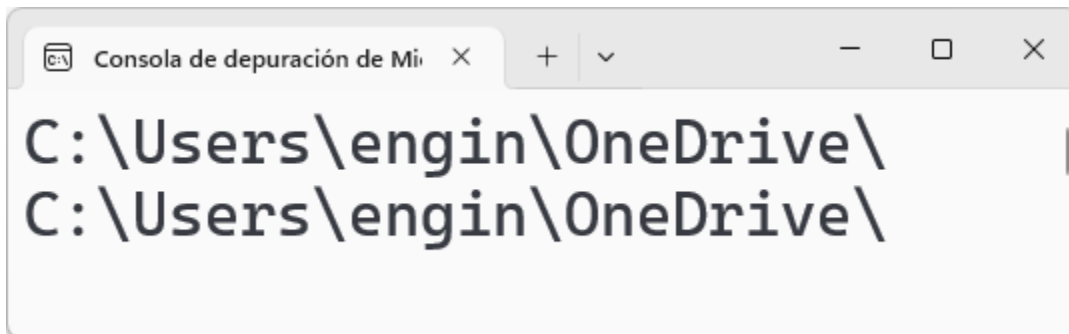


Ilustración 2: Uso de @



## Constantes con cadenas

Las constantes una vez definidas con valores, no se pueden cambiar.

B/003.cs

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Constante
            const string cadena = "abcdefghijkl";

            //Se intenta cambiar la constante y da error en compilación
            cadena = "Hola Mundo";

            //Imprimiendo por consola
            Console.WriteLine(cadena);
        }
    }
}
```

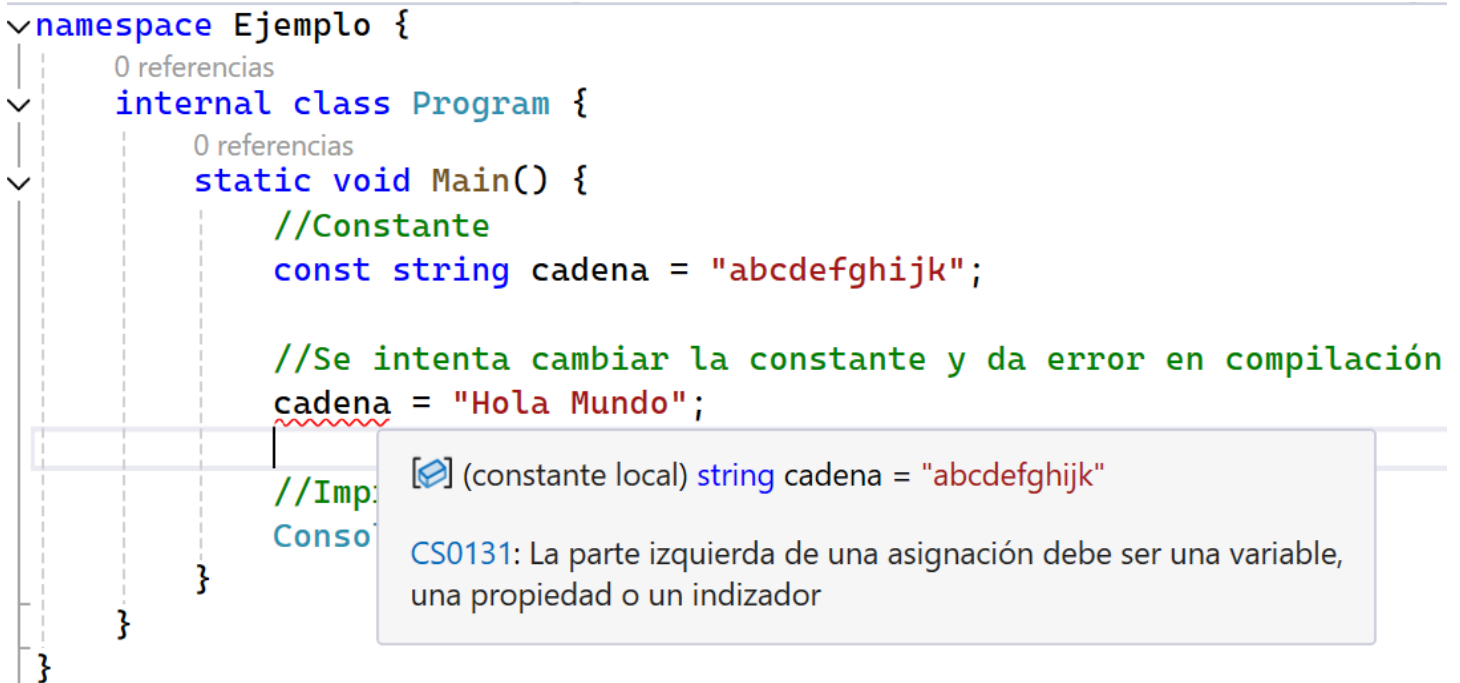


Ilustración 3: No se puede cambiar una constante

## Copia de cadenas

C# trata los strings como si fuesen un tipo de dato nativo en la práctica, por eso se copia el contenido de una variable a otra al usarse el operador de asignación (=).

B/004.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Inmutabilidad  
            string cadenaA = "abcdefghijkl";  
  
            //Se copian los datos de cadenaA en cadenaB  
            string cadenaB = cadenaA;  
  
            //Se agregan datos a cadenaA  
            //¿Qué sucederá con cadenaB?  
            cadenaA += "pqrstuvwxyz";  
  
            //Imprimiendo por consola  
            Console.WriteLine(cadenaA);  
            Console.WriteLine(cadenaB);  
        }  
    }  
}
```

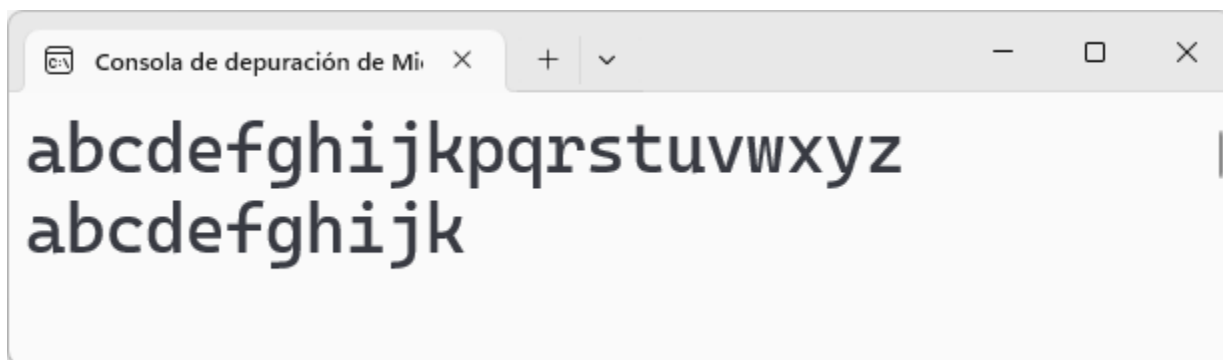


Ilustración 4: Copia de cadenas

## Caracteres especiales

Los caracteres especiales no se imprimen, sino que generan otro comportamiento al intentar "imprimirse".

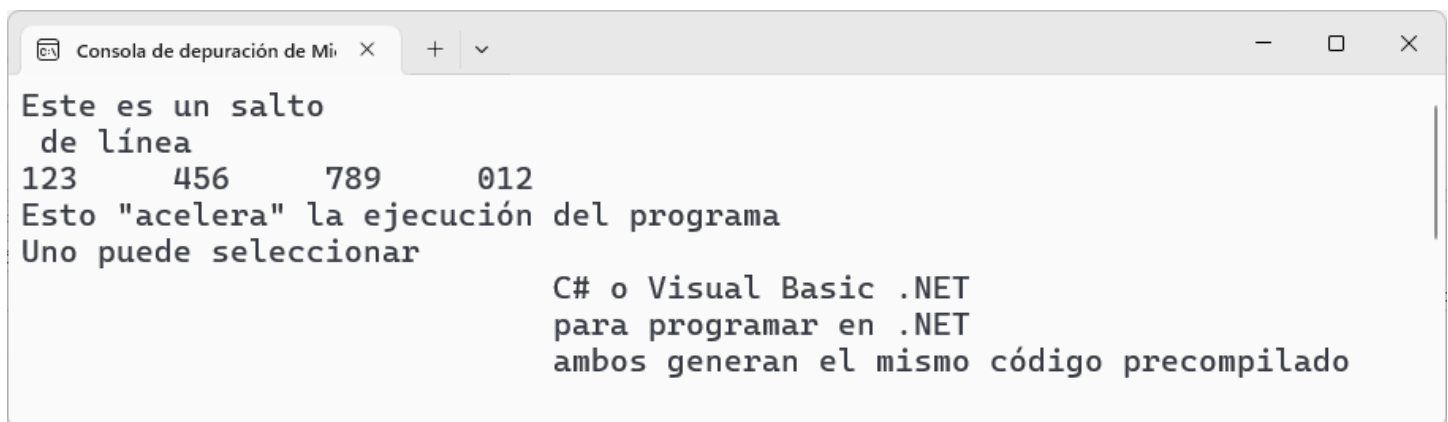
B/005.cs

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Caracteres especiales. Salto de línea
            string cadenaA = "Este es un salto \r\n de línea";
            Console.WriteLine(cadenaA);

            //Caracteres especiales. Tabuladores
            string cadenaB = "123\t456\t789\t012";
            Console.WriteLine(cadenaB);

            //Caracteres especiales. Imprimir las comillas dobles
            string cadenaC = "Esto \"acelera\" la ejecución del programa";
            Console.WriteLine(cadenaC);

            //Usando el verbatim (toma los caracteres internos)
            string cadenaD = @"Uno puede seleccionar
                             C# o Visual Basic .NET
                             para programar en .NET
                             ambos generan el mismo código precompilado";
            Console.WriteLine(cadenaD);
        }
    }
}
```



```
Este es un salto
de línea
123      456      789      012
Esto "acelera" la ejecución del programa
Uno puede seleccionar
                             C# o Visual Basic .NET
                             para programar en .NET
                             ambos generan el mismo código precompilado
```

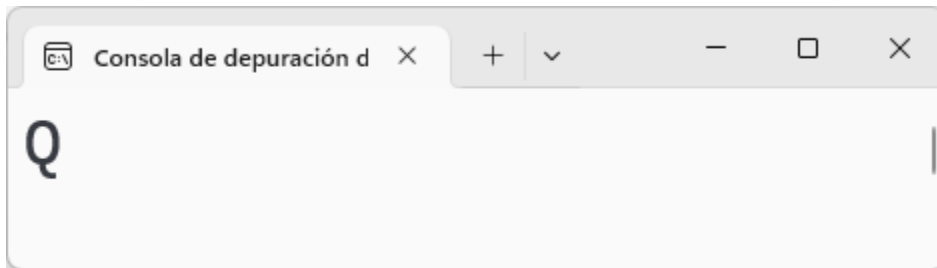
Ilustración 5: Caracteres especiales

## Acceder a un determinado carácter

En C# la primera letra en un string está en la posición cero.

B/006.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Acceder a determinado caracter  
            string cadena = "QWERTYUIOPabcdefghijklmnopqrstuvwxyz";  
            char letra = cadena[0]; //Accede a la primera letra  
            Console.WriteLine(letra);  
        }  
    }  
}
```



*Ilustración 6: Acceder a un determinado carácter*

## Tamaño de la cadena y recorrerla

Para obtener el tamaño en caracteres de una cadena se hace uso de la instrucción `Length`. En cuanto a su recorrido, se inicia en cero y la última letra sería el tamaño restándole uno.

B/007.cs

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Tamaño de cadena y recorrerla
            string cadena = "QWERTYUIOPabcdefghijklmnopqrstuvwxyz";
            int tamano = cadena.Length;
            Console.WriteLine(cadena);
            Console.WriteLine("Tamaño es: " + tamano.ToString());

            //Recorre la cadena
            for (int posicion=0; posicion < tamano; posicion++) {
                char letra = cadena[posicion]; //va de letra en letra
                Console.Write(letra.ToString() + " ; ");
            }
        }
    }
}
```

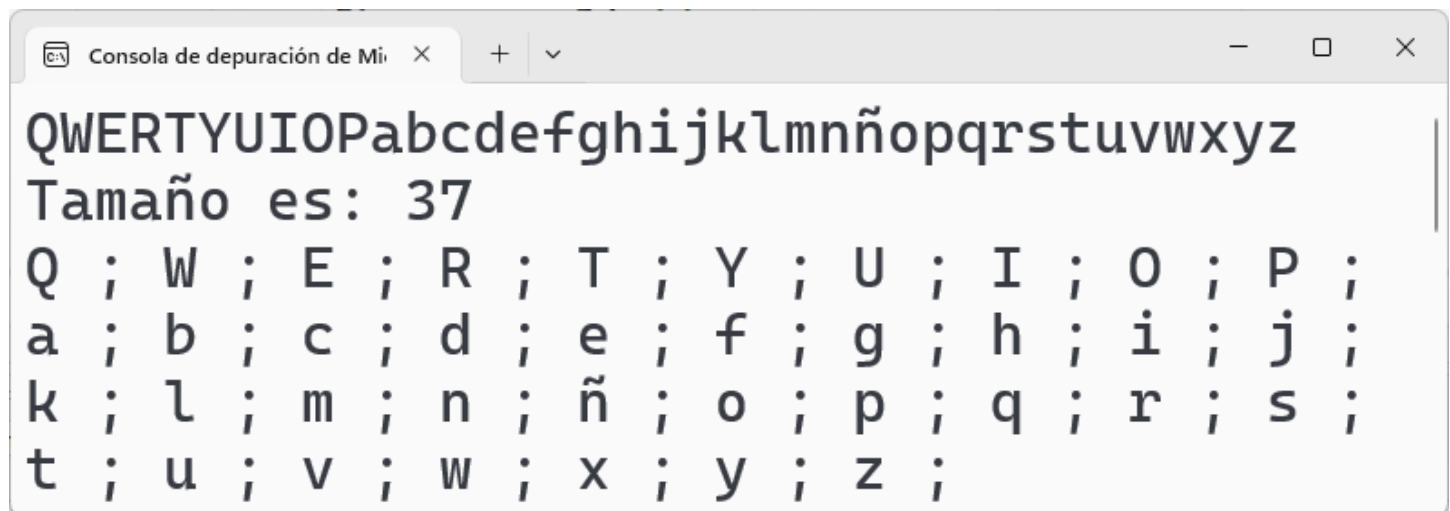


Ilustración 7: Tamaño de la cadena y recorrerla

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Subcadenas  
            string cadena = "abcdefghijklmnñopqrstuvwxyz";  
  
            //Del caracter 3 en adelante  
            string subCadA = cadena.Substring(3);  
            Console.WriteLine(subCadA);  
  
            //Del caracter 7 traiga 4 caracteres  
            string subCadB = cadena.Substring(7, 4);  
            Console.WriteLine(subCadB);  
        }  
    }  
}
```

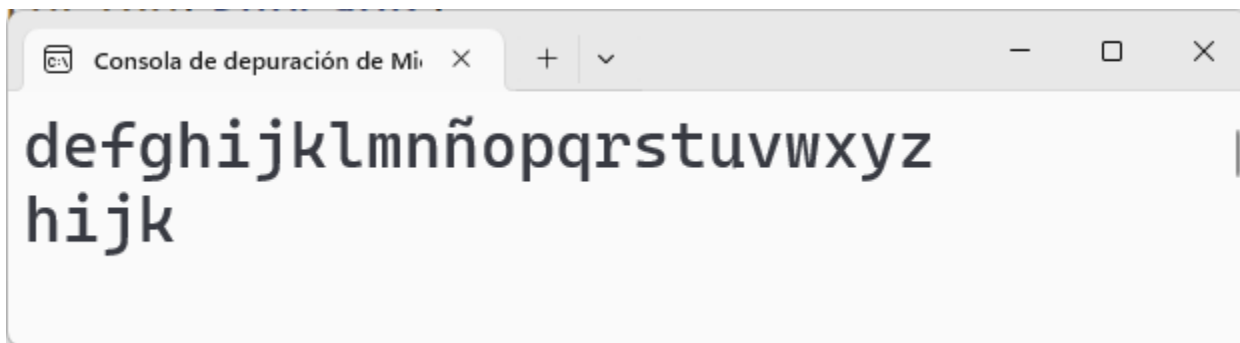


Ilustración 8: Subcadenas

# Reemplazar caracteres

B/009.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Reemplazar  
            string cadena = "manzana y naranja";  
            Console.WriteLine(cadena);  
  
            //Cambia en toda la cadena una letra por otra  
            string ReemplazaA = cadena.Replace('a', 'u');  
            Console.WriteLine(ReemplazaA);  
  
            //Cambia en toda la cadena una subcadena por otra subcadena  
            string ReemplazaB = cadena.Replace("na", "po");  
            Console.WriteLine(ReemplazaB);  
  
            //Cambia en toda la cadena una subcadena por una letra  
            string ReemplazaC = cadena.Replace("na", "x");  
            Console.WriteLine(ReemplazaC);  
  
            //Cambia en toda la cadena una letra por una subcadena  
            string ReemplazaD = cadena.Replace("n", "GH");  
            Console.WriteLine(ReemplazaD);  
  
            //Cambia en toda la cadena una letra por vacío  
            string ReemplazaE = cadena.Replace("a", "");  
            Console.WriteLine(ReemplazaE);  
        }  
    }  
}
```

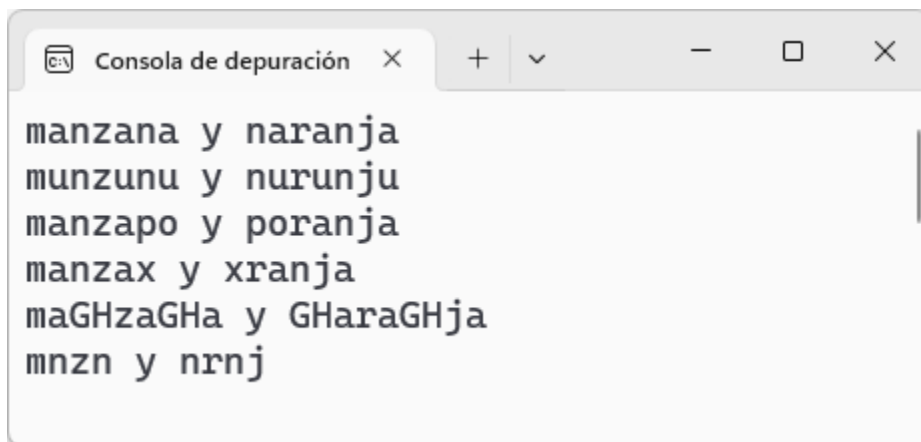


Ilustración 9: Reemplazar caracteres

## Encontrar subcadenas o caracteres

B/010.cs

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Encontrar subcadenas
            string cadena = "manzana y naranja";
            Console.WriteLine(cadena);

            //Busca la primera posición de la letra "a"
            int posA = cadena.IndexOf('a');
            Console.WriteLine("Posición de la primera 'a' es: " + posA);

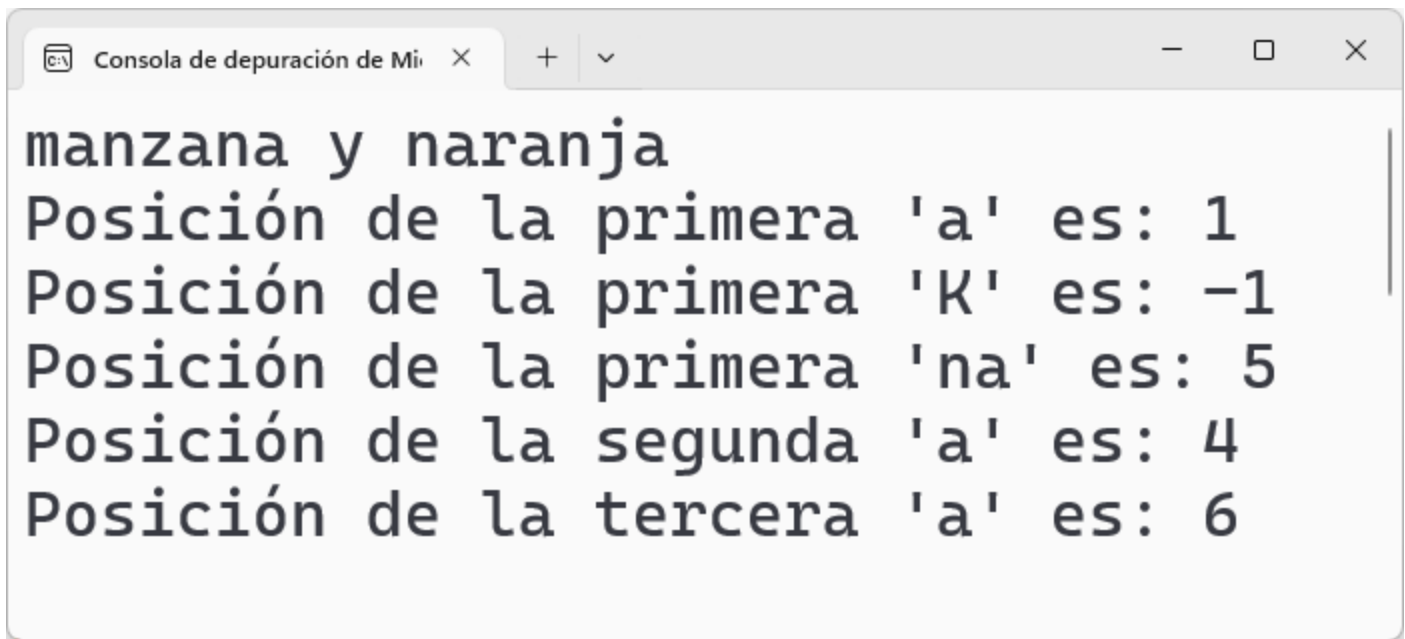
            //Busca una letra que no existe
            int posB = cadena.IndexOf('K');
            Console.WriteLine("Posición de la primera 'K' es: " + posB);

            //Busca la primera posición de la subcadena "na"
            int posC = cadena.IndexOf("na");
            Console.WriteLine("Posición de la primera 'na' es: " + posC);

            //Busca la segunda posición de la letra "a"
            int posD = cadena.IndexOf('a', posA + 1);
            Console.WriteLine("Posición de la segunda 'a' es: " + posD);

            //Busca la tercera posición de la letra "a"
            int posE = cadena.IndexOf('a', posD + 1);
            Console.WriteLine("Posición de la tercera 'a' es: " + posE);
        }
    }
}
```





The image shows a web browser window with a single tab titled 'Consola de depuración de Mi'. The developer console is open, displaying the following text:

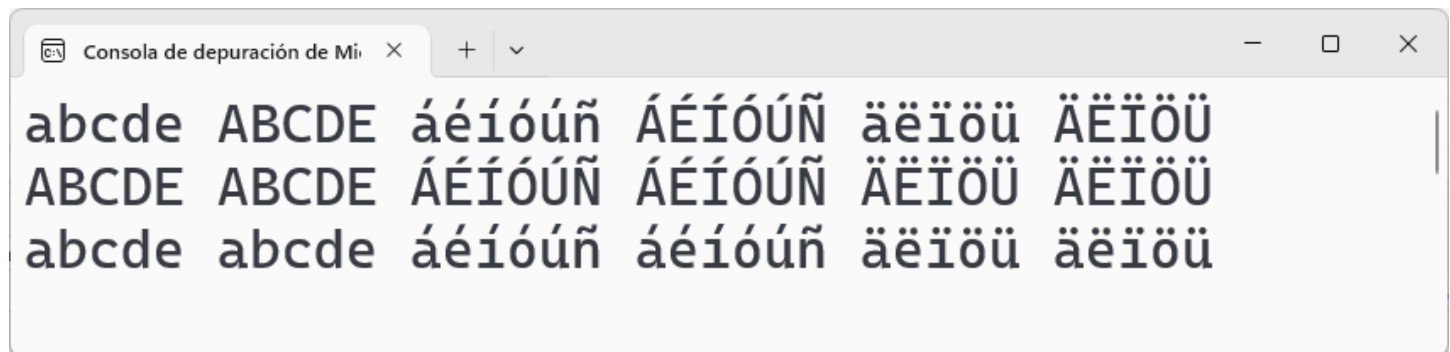
```
manzana y naranja
Posición de la primera 'a' es: 1
Posición de la primera 'K' es: -1
Posición de la primera 'na' es: 5
Posición de la segunda 'a' es: 4
Posición de la tercera 'a' es: 6
```

*Ilustración 10: Encontrar subcadenas o caracteres*

# Convertir a mayúsculas y minúsculas

B/011.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Mayúsculas y minúsculas  
            string cadena = "abcde ABCDE áéíóúñ ÁÉÍÓÚÑ äëïöü ÄËÏÖÜ";  
            Console.WriteLine(cadena);  
  
            //Convierte a mayúscula  
            string mayuscula = cadena.ToUpper();  
            Console.WriteLine(mayuscula);  
  
            //Convierte a minúscula  
            string minuscula = cadena.ToLower();  
            Console.WriteLine(minuscula);  
        }  
    }  
}
```



abcde ABCDE áéíóúñ ÁÉÍÓÚÑ äëïöü ÄËÏÖÜ  
ABCDE ABCDE ÁÉÍÓÚÑ ÁÉÍÓÚÑ ÄËÏÖÜ ÄËÏÖÜ  
abcde abcde áéíóúñ áéíóúñ äëïöü äëïöü

Ilustración 11: Convertir a mayúsculas y minúsculas

# StringBuilder, más veloz y se puede modificar su contenido

B/012.cs

```
using System.Text;
//Librería requerida para StringBuilder

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //StringBuilder, mayor velocidad y se puede modificar
            StringBuilder cadenaRapida = new("Prueba");
            Console.WriteLine(cadenaRapida);

            //Cambia el primer caracter
            cadenaRapida[0] = 'e';
            Console.WriteLine(cadenaRapida);

            //Agrega caracteres
            for (int Numero = 0; Numero <= 9; Numero++)
                cadenaRapida.Append(Numero);

            Console.WriteLine(cadenaRapida);
        }
    }
}
```

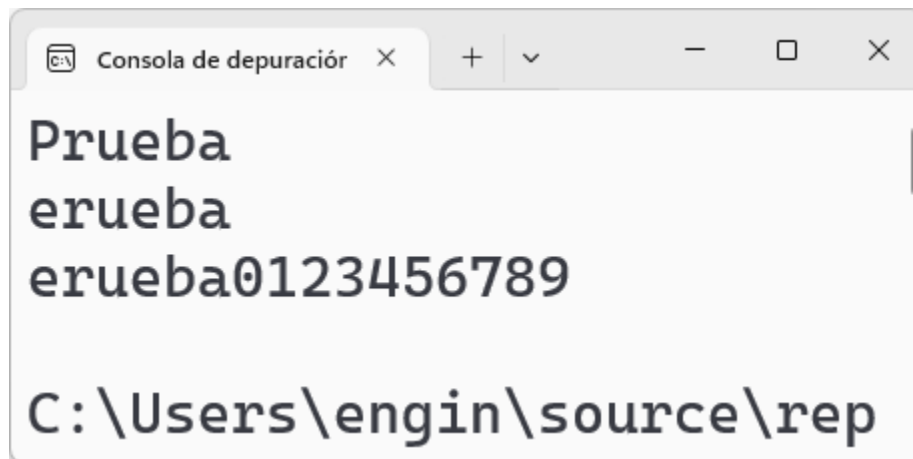


Ilustración 12: StringBuilder, más veloz y se puede modificar su contenido

# StringBuilder, métricas de velocidad

B/013.cs

```
using System.Diagnostics;
using System.Text;

namespace Ejemplo {
    internal class Program {
        static void Main() {
            //StringBuilder, comparando la velocidad
            StringBuilder cadenaRapida = new();
            string cadenaClasica = "";
            int NumLetras = 70000;

            //Medidor de tiempos
            Stopwatch cronometro = new();

            //Agrega caracteres a un StringBuilder
            cronometro.Reset();
            cronometro.Start();

            for (int numero = 1; numero <= NumLetras; numero++)
                cadenaRapida.Append(numero);

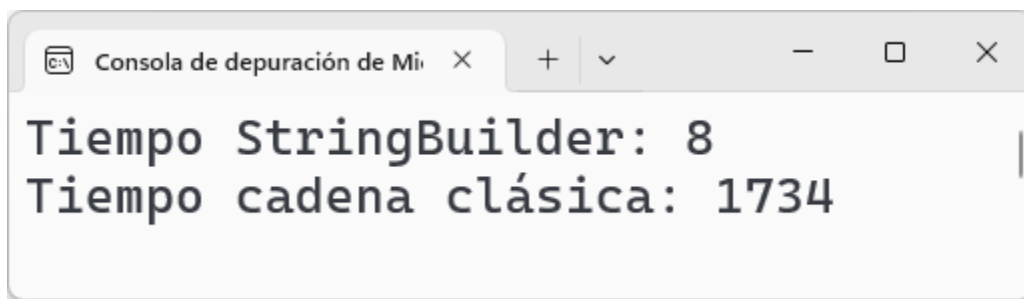
            long TBuilder = cronometro.ElapsedMilliseconds;

            //Agrega caracteres a un string
            cronometro.Reset();
            cronometro.Start();

            for (int numero = 1; numero <= NumLetras; numero++)
                cadenaClasica += numero;

            long TClasica = cronometro.ElapsedMilliseconds;

            if (cadenaClasica == cadenaRapida.ToString()) {
                Console.WriteLine("Tiempo StringBuilder: " + TBuilder);
                Console.WriteLine("Tiempo cadena clásica: " + TClasica);
            }
            else
                Console.WriteLine("Prueba errónea");
        }
    }
}
```



*Ilustración 13: StringBuilder, métricas de velocidad*

# Eliminar caracteres al inicio y al final

B/014.cs

```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Cortando espacios
            string cadenaA = "    Con espacios    ";
            Console.WriteLine "[" + cadenaA + "]");

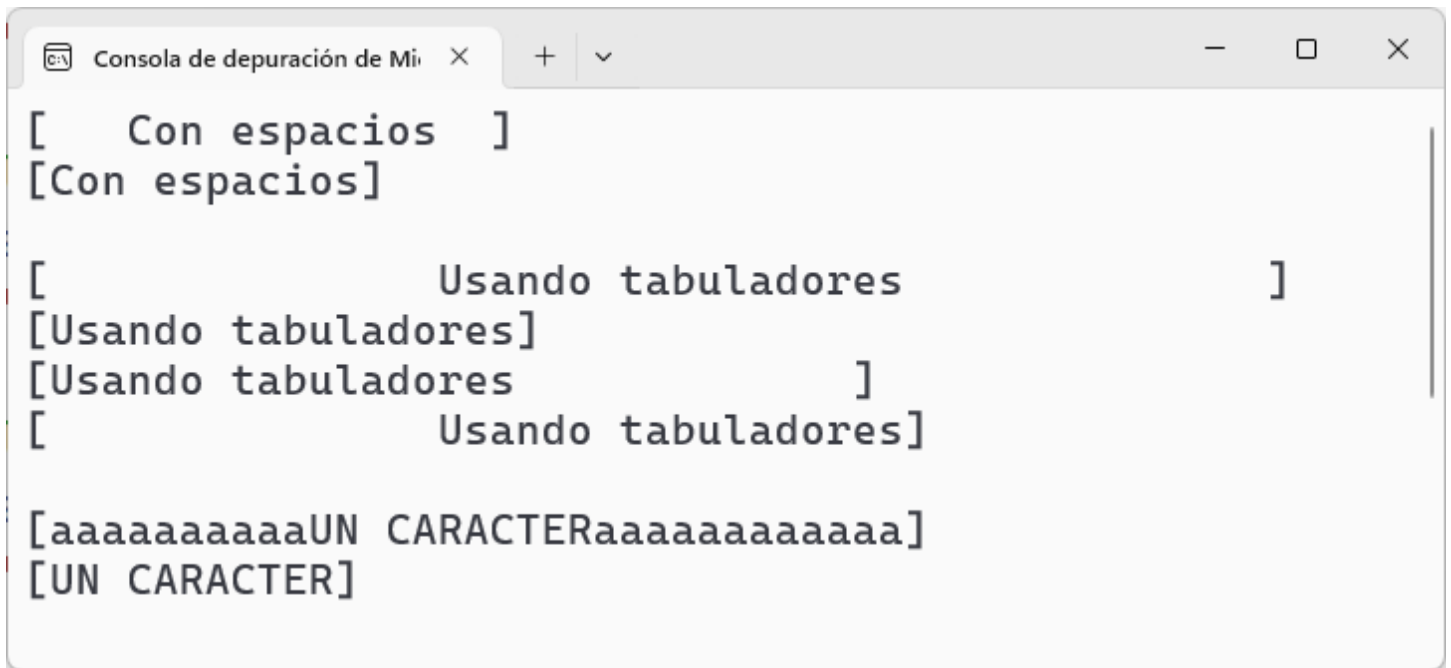
            //Quita los espacios de inicio y fin
            string cadenaB = cadenaA.Trim(' ');
            Console.WriteLine "[" + cadenaB + "]\r\n");

            //¿Y si son tabuladores? Los retira: \t
            string cadenaC = "\t\tUsando tabuladores\t\t";
            Console.WriteLine "[" + cadenaC + "]");
            string cadenaD = cadenaC.Trim('\t');
            Console.WriteLine "[" + cadenaD + "]");

            //Retirar los tabuladores de la izquierda
            string cadenaE = cadenaC.TrimStart('\t');
            Console.WriteLine "[" + cadenaE + "]");

            //Retirar los tabuladores de la derecha
            string cadenaF = cadenaC.TrimEnd('\t');
            Console.WriteLine "[" + cadenaF + "]\r\n");

            //Retirar otro caracter
            cadenaA = "aaaaaaaaaUN CARACTERaaaaaaaaa";
            Console.WriteLine "[" + cadenaA + "]");
            string cadenaG = cadenaA.Trim('a');
            Console.WriteLine "[" + cadenaG + "]");
        }
    }
}
```



The image shows a debug console window titled 'Consola de depuración de Mi'. It contains several lines of text demonstrating string formatting with spaces and tabs. The text is as follows:

```
[  Con espacios  ]  
[Con espacios]  
  
[          Usando tabuladores          ]  
[Usando tabuladores]  
[Usando tabuladores          ]  
[          Usando tabuladores]  
  
[aaaaaaaaaaaaUN CARACTERaaaaaaaaaaaaa]  
[UN CARACTER]
```

*Ilustración 14: Eliminar caracteres al inicio y al final*

## Comparar cadenas. Forma no recomendada

No se recomienda hacer uso del operador == al usar cadenas.

B/015.cs

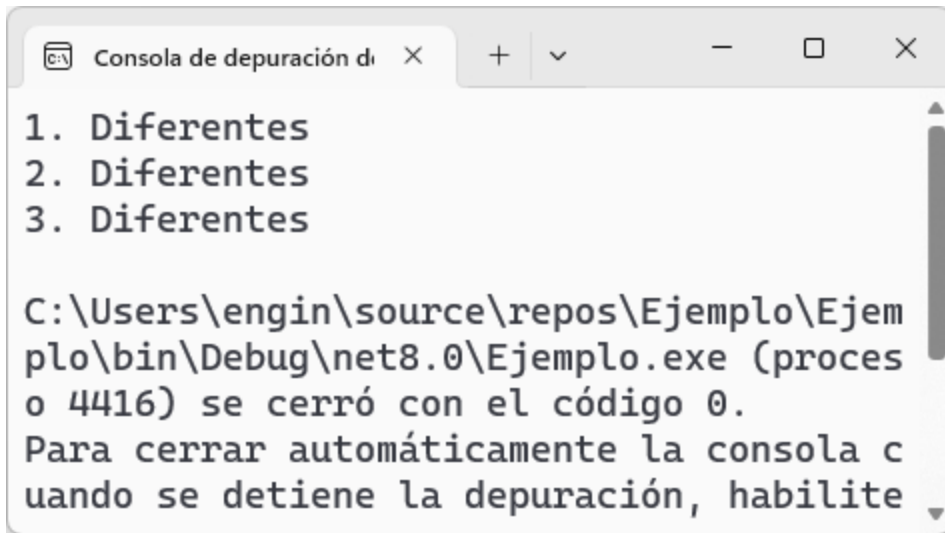
```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Comparar cadenas
            string cadenaA = "abcdefghij";
            string cadenaB = "Abcdefghij";
            string cadenaC = "abcdefghij ";
            string cadenaD = "abcdefg hij";

            //Forma 1 de comparar. No recomendada.
            if (cadenaA == cadenaB)
                Console.WriteLine("1. Iguales");
            else
                Console.WriteLine("1. Diferentes");

            if (cadenaA == cadenaC)
                Console.WriteLine("2. Iguales");
            else
                Console.WriteLine("2. Diferentes");

            if (cadenaA == cadenaD)
                Console.WriteLine("3. Iguales");
            else
                Console.WriteLine("3. Diferentes");
        }
    }
}
```





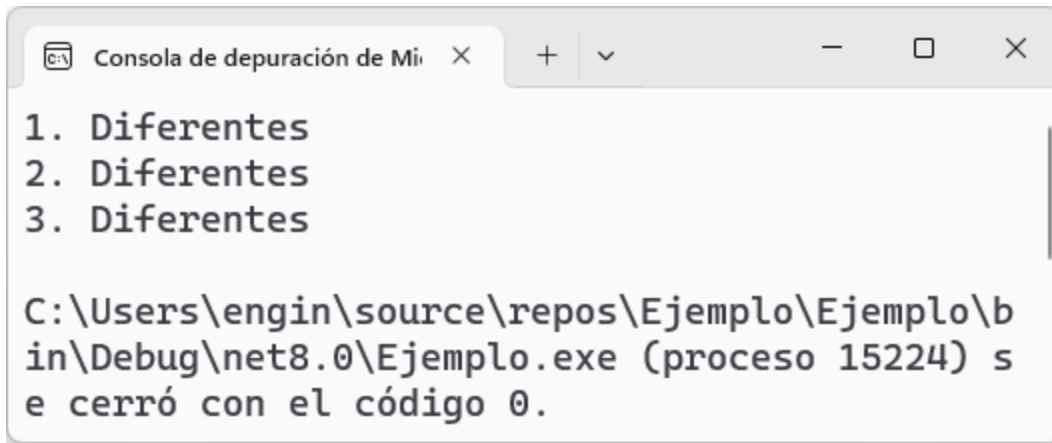
*Ilustración 15: Comparar cadenas. Forma no recomendada*

## Comparar cadenas. Forma recomendada

Para comparar es mejor usar el operador Equals

B/016.cs

```
namespace Ejemplo {  
    internal class Program {  
        static void Main() {  
            //Comparar cadenas  
            string cadenaA = "abcdefghij";  
            string cadenaB = "Abcdefghij";  
            string cadenaC = "abcdefghij ";  
            string cadenaD = "abcdefg hij";  
  
            //Forma 2 de comparar. Recomendada.  
            if (cadenaA.Equals(cadenaB))  
                Console.WriteLine("1. Iguales");  
            else  
                Console.WriteLine("1. Diferentes");  
  
            if (cadenaA.Equals(cadenaC))  
                Console.WriteLine("2. Iguales");  
            else  
                Console.WriteLine("2. Diferentes");  
  
            if (cadenaA.Equals(cadenaD))  
                Console.WriteLine("3. Iguales");  
            else  
                Console.WriteLine("3. Diferentes");  
        }  
    }  
}
```



```
1. Diferentes
2. Diferentes
3. Diferentes

C:\Users\engin\source\repos\Ejemplo\Ejemplo\bin\Debug\net8.0\Ejemplo.exe (proceso 15224) se cerró con el código 0.
```

*Ilustración 16: Comparar cadenas. Forma recomendada*

# Comparar cadenas. Ignorando las mayúsculas y minúsculas

B/017.cs

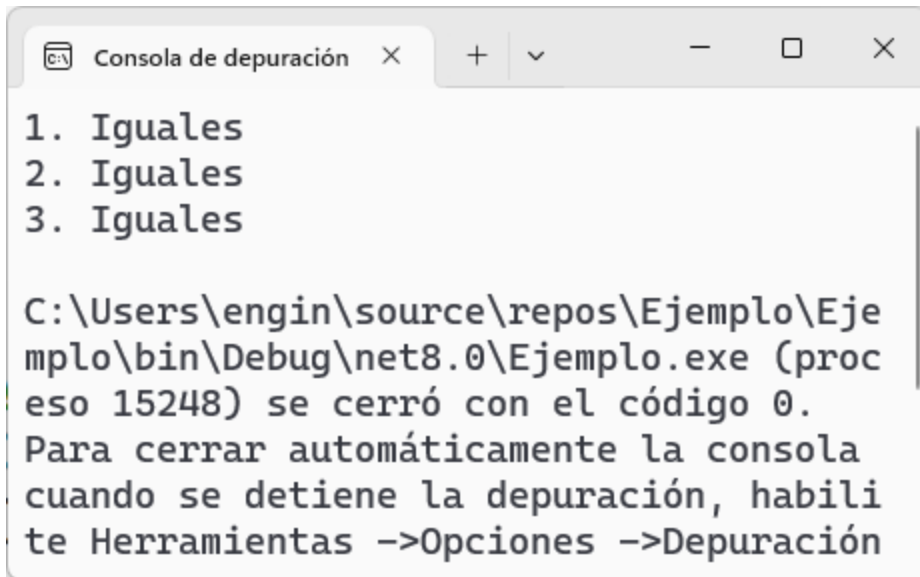
```
namespace Ejemplo {
    internal class Program {
        static void Main() {
            //Comparar cadenas
            string cadenaA = "abcdefghij";
            string cadenaB = "Abcdefghij";
            string cadenaC = "aBCDEfghiJ";
            string cadenaD = "ABCDEFGHIj";

            //Forma 2 de comparar ignorando mayúsculas y minúsculas
            if (cadenaA.Equals(cadenaB, StringComparison.OrdinalIgnoreCase))
                Console.WriteLine("1. Iguales");
            else
                Console.WriteLine("1. Diferentes");

            if (cadenaA.Equals(cadenaC, StringComparison.OrdinalIgnoreCase))
                Console.WriteLine("2. Iguales");
            else
                Console.WriteLine("2. Diferentes");

            if (cadenaA.Equals(cadenaD, StringComparison.OrdinalIgnoreCase))
                Console.WriteLine("3. Iguales");
            else
                Console.WriteLine("3. Diferentes");
        }
    }
}

//Más información:
//https://docs.microsoft.com/en-us/dotnet/csharp/how-to/compare-strings
```



*Ilustración 17: Comparar cadenas. Ignorando las mayúsculas y minúsculas*