

C# Y .NET 10

Parte 02. Cadenas

2025-12

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
Inmutabilidad de las cadenas	7
Declaración de cadenas	7
Uso de @ en cadenas.....	9
Constantes con cadenas.....	10
Copia de cadenas	11
Caracteres especiales	12
Acceder a un determinado carácter	13
Tamaño de la cadena y recorrerla	14
Subcadenas	15
Reemplazar caracteres.....	16
Encontrar subcadenas o caracteres	18
Convertir a mayúsculas y minúsculas	20
StringBuilder, más veloz y se puede modificar su contenido.....	21
StringBuilder, métricas de velocidad	22
Eliminar caracteres al inicio y al final	24
Comparar cadenas. Forma no recomendada	26
Comparar cadenas. Forma recomendada.....	27
Comparar cadenas. Ignorando las mayúsculas y minúsculas.....	28

Tabla de ilustraciones

Ilustración 1: Declaración de cadenas.....	8
Ilustración 2: Uso de @.....	9
Ilustración 3: No se puede cambiar una constante.....	10
Ilustración 4: Copia de cadenas	11
Ilustración 5: Caracteres especiales	12
Ilustración 6: Acceder a un determinado carácter	13
Ilustración 7: Tamaño de la cadena y recorrerla	14
Ilustración 8: Subcadenas	15
Ilustración 9: Reemplazar caracteres	17
Ilustración 10: Encontrar subcadenas o caracteres	19
Ilustración 11: Convertir a mayúsculas y minúsculas.....	20
Ilustración 12: StringBuilder, más veloz y se puede modificar su contenido.....	21
Ilustración 13: StringBuilder, métricas de velocidad	23
Ilustración 14: Eliminar caracteres al inicio y al final	25
Ilustración 15: Comparar cadenas. Forma no recomendada.....	26
Ilustración 16: Comparar cadenas. Forma recomendada	27
Ilustración 17: Comparar cadenas. Ignorando las mayúsculas y minúsculas.....	28

Acerca del autor

Rafael Alberto Moreno Parra

ramsoftware@gmail.com o enginelife@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 2026 ® Enlace: <https://visualstudio.microsoft.com/es/vs/>

Dedicatoria

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

Inmutabilidad de las cadenas

En C#, las cadenas son inmutables. Es decir, que una variable (de tipo string) una vez iniciada con una cadena, esta no puede cambiar ni una letra directamente. ¿Por qué? El manejo de cadenas es simplificado con la inmutabilidad porque ya no hay riesgo que estas cambien de forma inesperada. Es útil en el caso de manejo de hilos porque si hay varios accediendo a la misma cadena, esta no va a cambiar. Otro beneficio de la inmutabilidad es que pueden ser almacenadas en caché, por lo que se optimiza el rendimiento.

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);

        //Uniendo dos cadenas o concatenar
        string cadenaC = cadenaA + " UNIR " + cadenaB;
        Console.WriteLine(cadenaC);
    }
}
```

```
Consola de depuración de Mi X + ▾  
Esto es una prueba de texto  
Asignando valores alfanuméricos  
Esto es una prueba de texto UNIR Asignando valores alfanuméricos
```

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);
    }
}
```

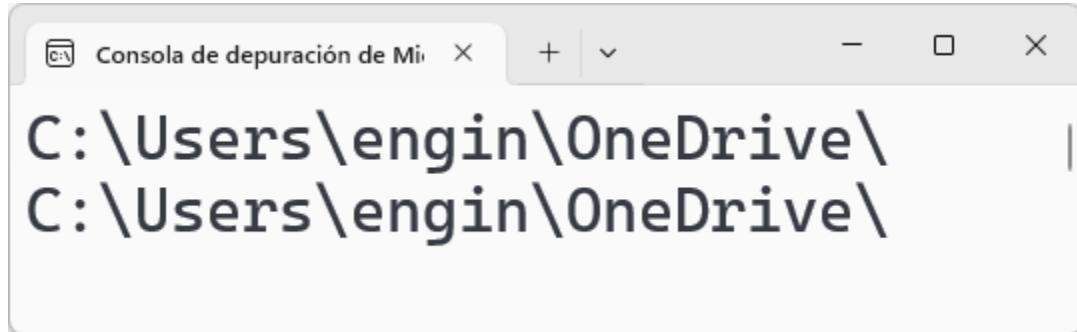


Ilustración 2: Uso de @

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 = "abcdefghijklm";

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

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

```
namespace Ejemplo;
```

0 referencias

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

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



[?] (constante local) string cadena = "abcdefghijklm"

Constante

❖ Describir con Copilot

CS0131: La parte izquierda de una asignación debe ser una variable, una propiedad o un indizador

Mostrar posibles correcciones (Alt+Entrar o Ctrl+.)

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

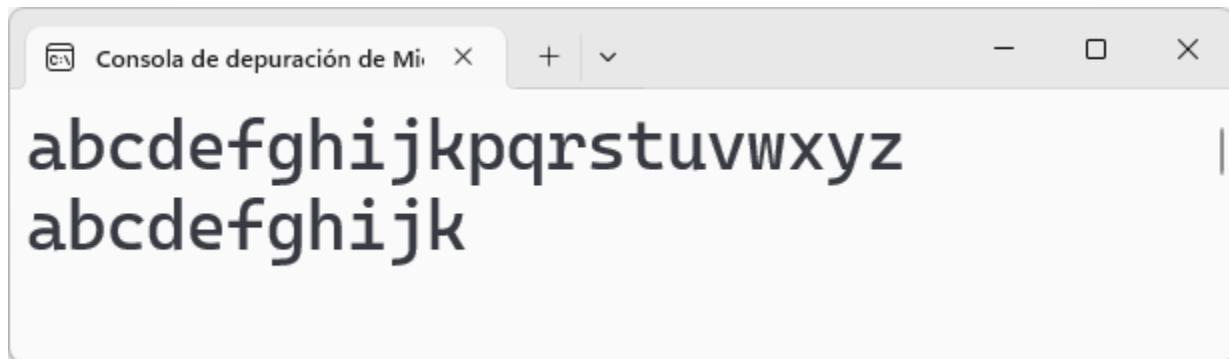
```
namespace Ejemplo;

internal class Program {
    static void Main() {
        //Inmutabilidad
        string cadenaA = "abcdefghijklk";

        //Se copian los datos de cadenaA en cadenaB
        string cadenaB = cadenaA;

        //Se agregan datos a cadenaA
        //¿Qué sucederá con cadenaB?
        cadenaA += "pqrsuvwxyz";

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



The screenshot shows a Windows Command Prompt window titled "Consola de depuración de Mi...". It displays two lines of text: "abcdefghijklk" and "abcdefghijklk". The first line is bolded, and the second line is regular, demonstrating that the assignment operator (=) creates a shallow copy of the string reference.

Ilustración 4: Copia de cadenas

Caracteres especiales

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

B/005.cs

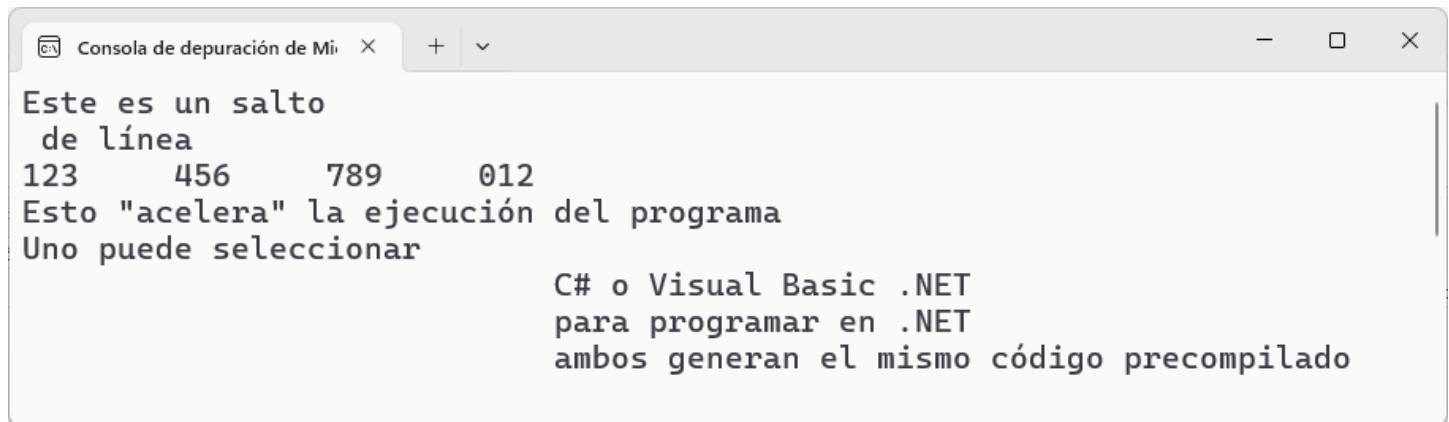
```
namespace Ejemplo;

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 = "QWERTYUIOPabcdefghijklmnñopqrstuvwxyz";
        char letra = cadena[0]; //Accede a la primera letra
        Console.WriteLine(letra);
    }
}
```

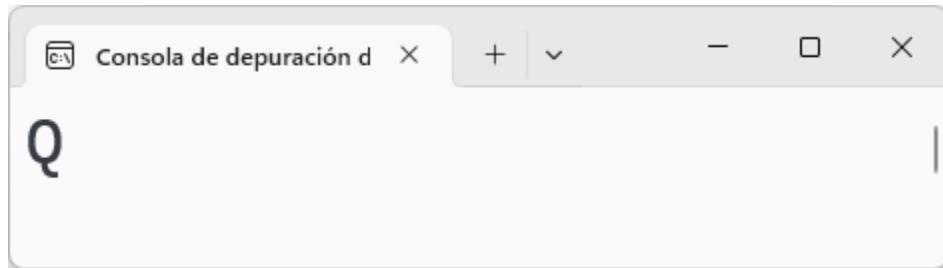


Ilustración 6: Acceder a un determinado carácter

Tamaño de la cadena y recorrerla

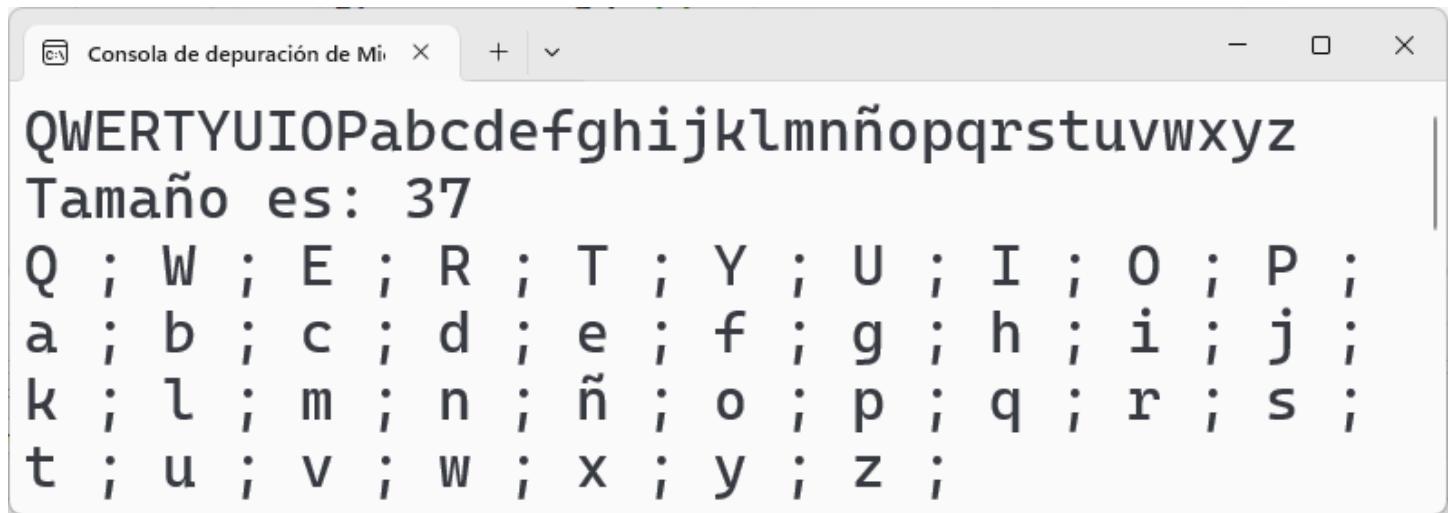
Para obtener el tamaño en caracteres de una cadena se hace uso de la instrucción Length. En cuanto a su recorrido, se inicia en cero y la última letra sería el tamaño restándole uno.

B/007.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        //Tamaño de cadena y recorrerla
        string cadena = "QWERTYUIOPabcdefghijklmnñopqrstuvwxyz";
        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() + " ; ");
        }
    }
}
```



```
QWERTYUIOPabcdefghijklmnñopqrstuvwxyz
Tamaño es: 37
Q ; W ; E ; R ; T ; Y ; U ; I ; O ; P ;
a ; b ; c ; d ; e ; f ; g ; h ; i ; j ;
k ; l ; m ; n ; ñ ; o ; p ; q ; r ; s ;
t ; u ; v ; w ; x ; y ; z ;
```

Ilustración 7: Tamaño de la cadena y recorrerla

Subcadenas

B/008.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        //Subcadenas
        string cadena = "abcdefghijklmnñopqrstuvwxyz";

        //Del carácter 3 en adelante
        string subCadA = cadena.Substring(3);
        Console.WriteLine(subCadA);

        //Del carácter 7 traiga 4 caracteres
        string subCadB = cadena.Substring(7, 4);
        Console.WriteLine(subCadB);
    }
}
```

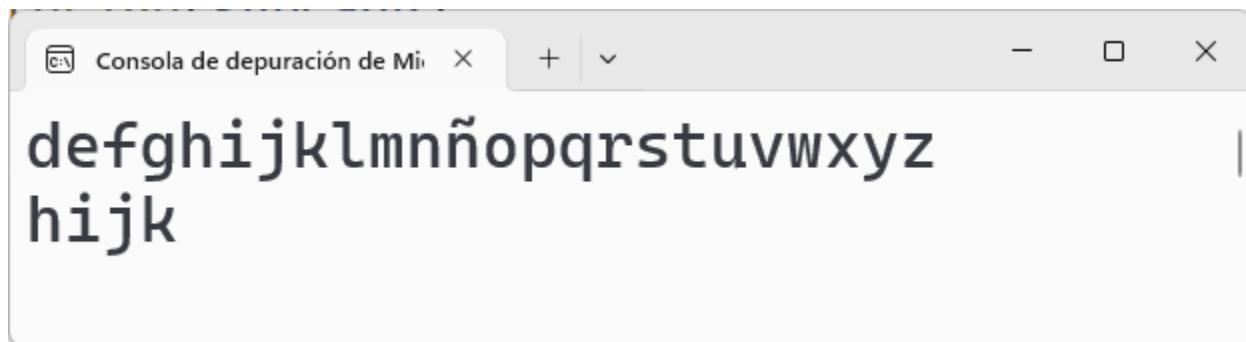


Ilustración 8: Subcadenas

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);
    }
}
```

```
manzana y naranja
munzunu y nurunju
manzapo y poranja
manzax y xranja
maGHzaGHa y GHaraGHja
mnzn y nrnj
```

Ilustración 9: Reemplazar caracteres

Considerar el siguiente código en C#:

```
string original = "Hola";
string modificada = original.Replace("H", "M");
```

```
Console.WriteLine(original); // Salida: Hola
```

```
Console.WriteLine(modificada); // Salida: Mola
```

En este ejemplo, el método Replace no modifica la cadena original, sino que crea una nueva cadena con el resultado de la operación. La cadena original permanece intacta, demostrando su inmutabilidad.

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 screenshot shows a terminal window with the title 'Consola de depuración de Mi...'. The window contains 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() {
        //Encontrar subcadenas
        string cadena = "manzana y naranja";
        Console.WriteLine(cadena);

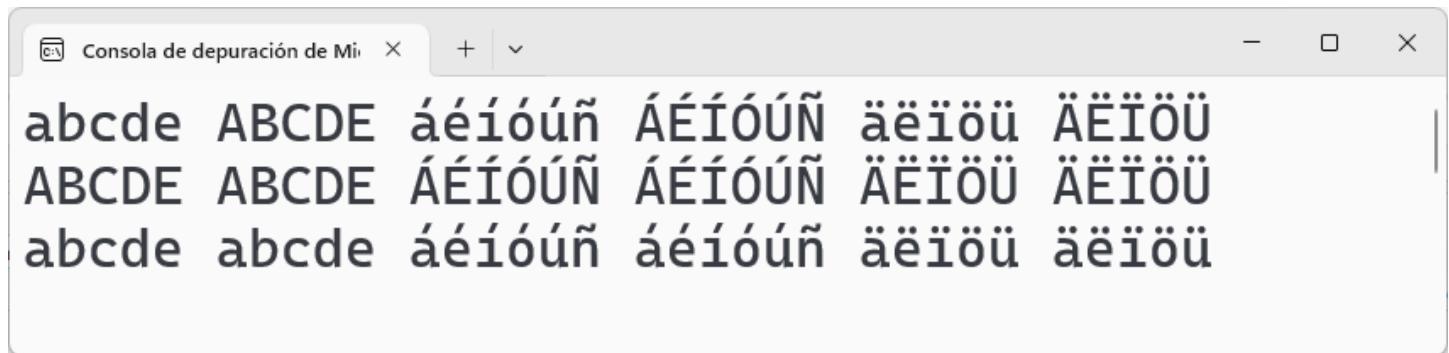
        //Busca la primera posición de la letra "a"
        int posA = cadena.IndexOf('a');
        Console.WriteLine("Posición de la primera 'a' es: " + posA);

        //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);
    }
}
```



```
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 carácter
        cadenaRapida[0] = 'e';
        Console.WriteLine(cadenaRapida);

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

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

The screenshot shows a Windows Command Prompt window titled "Consola de depuración". The window contains the following text:
Prueba
erueba
erueba0123456789
C:\Users\engin\source\rep

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 = 90000;

            //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");
        }
    }
}
```

A screenshot of a Java IDE's terminal window titled "Consola de depuración de Mi". The window contains two lines of text: "Tiempo StringBuilder: 8" and "Tiempo cadena clásica: 4527".

```
Consola de depuración de Mi + ▾
Tiempo StringBuilder: 8
Tiempo cadena clásica: 4527
```

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 = "aaaaaaaaaaUN CARACTERaaaaaaaaaaa";
        Console.WriteLine("[" + cadenaA + "]");
        string cadenaG = cadenaA.Trim('a');
        Console.WriteLine("[" + cadenaG + "]");
    }
}
```

The screenshot shows a terminal window titled "Consola de depuración de Mi". The window contains several lines of text demonstrating string manipulation:

- [Con espacios]
- [Con espacios]
- [Usando tabuladores]
- [Usando tabuladores]
- [Usando tabuladores]
- [Usando tabuladores]
- [aaaaaaaaaaUN CARACTERaaaaaaaaaaa]
- [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 = "abcdefghijkl";
        string cadenaB = "Abcdefghij";
        string cadenaC = "abcdefghijkl ";
        string cadenaD = "abcdefghijkl 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");
    }
}
```



```
1. Diferentes
2. Diferentes
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 = "abcdefghijkl";
        string cadenaB = "Abcdefghij";
        string cadenaC = "abcdefghijkl ";
        string cadenaD = "abcdefghijkl 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");
    }
}
```



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 = "abcdefghijkl";
        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
```

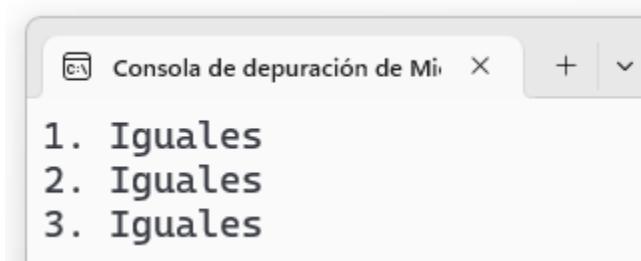


Ilustración 17: Comparar cadenas. Ignorando las mayúsculas y minúsculas