



ESTRUCTURAS BÁSICAS DE MEMORIA EN C#

Manejo de cifras, cadenas (strings), arreglos unidimensionales, bidimensionales, multidimensionales y arreglo de arreglos

[Empiece a manejar múltiples datos en memoria](#)

Contenido

Otros libros del autor 2

Página web del autor y canal en Youtube 2

Sitio en GitHub 2

Licencia del software 3

Marcas registradas 3

Introducción..... 4

Trabajando con las cifras de un número entero positivo..... 5

Cadenas (strings)..... 12

 Declaración de cadenas 12

 Uso de @ en cadenas..... 13

 Constantes con cadenas 14

 Copia de cadenas 15

 Caracteres especiales..... 16

 Acceder a un determinado carácter 17

 Tamaño de la cadena y recorrerla 18

 Subcadenas..... 19

 Reemplazar caracteres 20

 Encontrar subcadenas o caracteres..... 21

 Convertir a mayúsculas y minúsculas..... 22

 StringBuilder, más veloz y se puede modificar su contenido 23

 StringBuilder, métricas de velocidad 24

 Eliminar caracteres al inicio y al final 25

 Comparar cadenas. Forma no recomendada..... 26

 Comparar cadenas. Forma recomendada..... 27

 Comparar cadenas. Ignorando las mayúsculas y minúsculas..... 28

Arreglos Unidimensionales 29

 Declaración y asignar valores 29

 Otra forma de definir y asignar al tiempo 30

 Arreglo unidimensional de cadenas..... 31

 Arreglo unidimensional. Tamaños..... 32

 Recorrer arreglos de cadenas 33

 Uso de la instrucción foreach para recorrer un arreglo 34

 Ordenar un arreglo de cadenas..... 35

 Ordenar un arreglo de cadenas. Las mayúsculas, minúsculas, tildes y diéresis..... 36

 Máximo, mínimo, suma con Linq 38

 Funciones genéricas para arreglos unidimensionales 39

 Algoritmos de ordenación 40

 Algoritmos de ordenación. Prueba de velocidad 43

Arreglo bidimensional 46

 Definición..... 46

Arreglo tridimensional..... 47

 Definición..... 47

Arreglo de arreglos..... 48

 Definición..... 48

 Prueba de velocidad, arreglo bidimensional vs arreglo de arreglos 49

Arreglos multidimensionales 50

Tipo implícito de arreglo 52

Tipo implícito en arreglo de arreglos..... 53

Convertir una cadena en un arreglo de cadenas al dividirla 54

Ordenando una cadena por dentro 56

Ordenando un arreglo de cadenas con Linq 58

 Orden por tamaño 58

 Orden por la segunda letra..... 59

 Invierte el arreglo..... 60

Otros libros del autor

Libro 14: "Iniciando en C#". En Colombia 2020. Págs. 72. Libro y código fuente descargable en: <https://github.com/ramsoftware/C-Sharp-Iniciando>

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Página web del autor y canal en Youtube

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

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

Sitio en GitHub

El código fuente se puede descargar en <https://github.com/ramsoftware/EstructuraBasicaMemoriaCSharp>

Licencia del software

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



Marcas registradas

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

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

Microsoft ® Visual Studio 2019 ® Enlace: <https://visualstudio.microsoft.com/es/vs/>

Introducción

Este libro está dirigido a un segundo curso de programación que cubre los temas de manejo de estructuras de almacenamiento en memoria. Se trabaja con las cifras de un número entero, cadenas (strings), arreglos unidimensionales, algoritmos de ordenación, arreglos bidimensionales, tridimensionales y arreglo de arreglos.

Se hace uso de Microsoft Visual Studio Community 2019.

En algunos puntos se hace mención sobre el desempeño de ciertas estructuras y se hacen medidas de tiempo como observación. A futuro escribiré un libro dedicado a las métricas de software de diversas estructuras en C#.

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

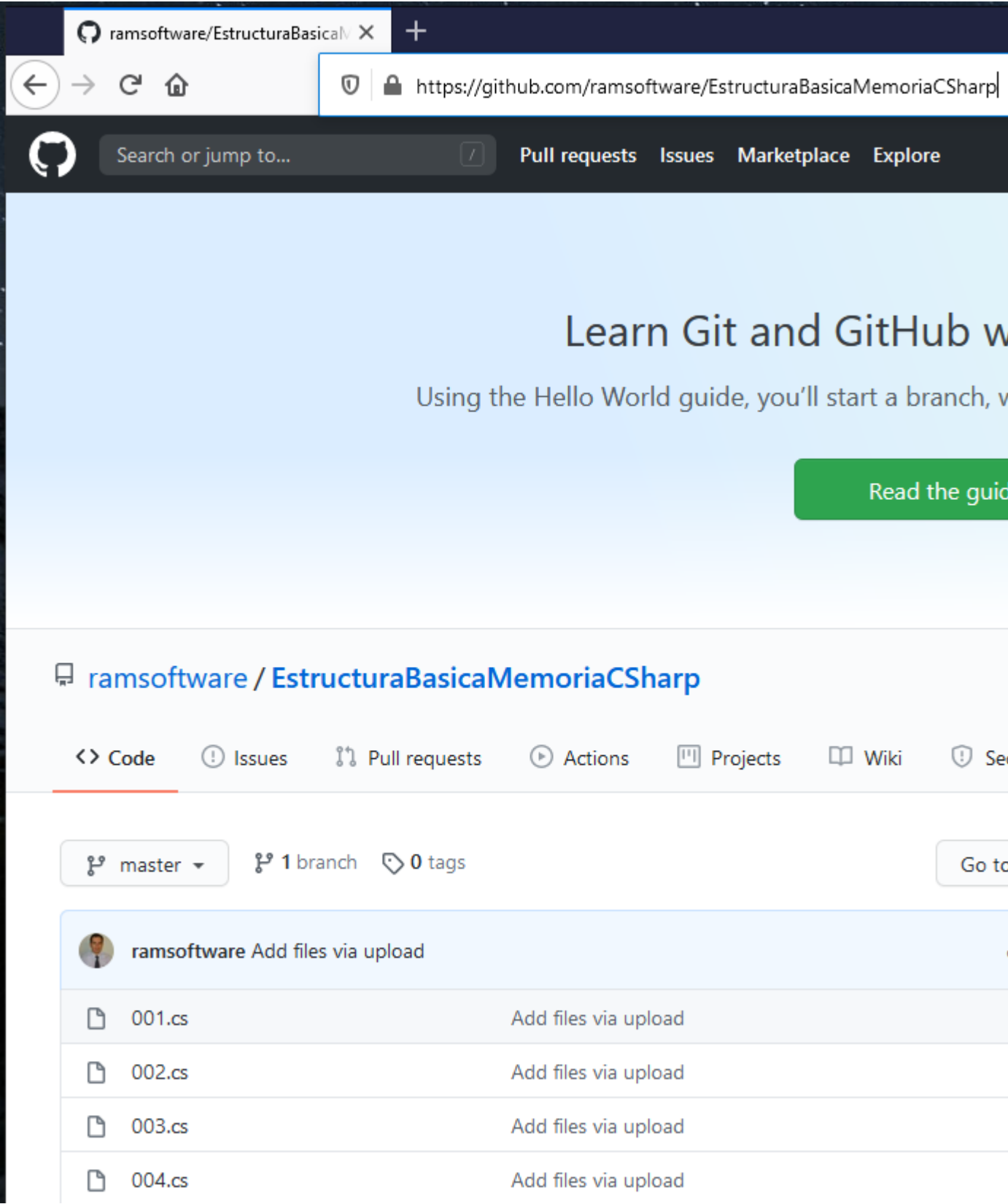


Ilustración 1: Contenido en GitHub

Trabajando con las cifras de un número entero positivo

Puede llegar a ser útil manipular las cifras de un número entero positivo, a continuación, una serie de funciones que le permitirán acceder a cualquier cifra o hacer operaciones con cifras.

001.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Funciones de números
            int numero = 27943030;
            Console.WriteLine(numero.ToString() + " es impar: " + EsImpar(numero).ToString());
            Console.WriteLine(numero.ToString() + " es par: " + EsPar(numero).ToString());
            Console.WriteLine(numero.ToString() + " total de cifras: " + TotalCifras(numero).ToString());
            Console.WriteLine(numero.ToString() + " sus dos últimas cifras: " +
Retornadosultimascifras(numero).ToString());
            Console.WriteLine(numero.ToString() + " su antepenúltima cifra: " +
AntepenultimaCifra(numero).ToString());
            Console.WriteLine(numero.ToString() + " su penúltima cifra: " +
PenultimaCifra(numero).ToString());
            Console.WriteLine(numero.ToString() + " su última cifra: " + UltimaCifra(numero).ToString());
            Console.WriteLine(numero.ToString() + " la cifra más alta: " +
LaCifraMasAlta(numero).ToString());
            Console.WriteLine(numero.ToString() + " la cifra más baja: " +
LaCifraMasBaja(numero).ToString());
            Console.WriteLine(numero.ToString() + " total de cifras iguales a 5 es: " +
Cifrashalladas(numero, 5).ToString());
            Console.WriteLine(numero.ToString() + " al invertirlo es: " +
InvierteNumero(numero).ToString());
            Console.WriteLine(numero.ToString() + " es palíndromo: " + EsPalindromo(numero).ToString());
            Console.WriteLine(numero.ToString() + " tercera cifra es: " + CifraPosicion(numero,
3).ToString());
            Console.WriteLine(numero.ToString() + " primera cifra es: " +
PrimeraCifra(numero).ToString());
            Console.WriteLine(numero.ToString() + " la suma de las cifras es: " +
SumaCifras(numero).ToString());
            Console.WriteLine(numero.ToString() + " la suma de las cifras pares es: " +
SumaCifrasPares(numero).ToString());
            Console.WriteLine(numero.ToString() + " la suma de las cifras impares es: " +
SumaCifrasImpares(numero).ToString());
            Console.WriteLine(numero.ToString() + " la multiplicación de las cifras es: " +
MultiplicaCifras(numero).ToString());
            Console.WriteLine(numero.ToString() + " la multiplicación de las cifras pares es: " +
MultiplicaCifrasPares(numero).ToString());
            Console.WriteLine(numero.ToString() + " la multiplicación de las cifras impares es: " +
MultiplicaCifrasImpares(numero).ToString());
            Console.WriteLine(numero.ToString() + " todas las cifras son pares: " +
TodasCifrasPares(numero).ToString());
            Console.WriteLine(numero.ToString() + " todas las cifras son impares: " +
TodasCifrasImpares(numero).ToString());
            Console.WriteLine(numero.ToString() + " total cifras pares: " +
TotalCifrasPares(numero).ToString());
            Console.WriteLine(numero.ToString() + " total cifras impares: " +
TotalCifrasImpares(numero).ToString());
            Console.WriteLine(numero.ToString() + " solo hay cifras menores o iguales a 5: " +
SoloCifrasMenorIgual(numero, 5).ToString());
            Console.WriteLine(numero.ToString() + " solo hay cifras mayores o iguales a 5: " +
SoloCifrasMayorIgual(numero, 5).ToString());
            Console.WriteLine(numero.ToString() + " usa distintas cifras: " +
DistintasCifras(numero).ToString());
            Console.WriteLine(numero.ToString() + " usa distintas cifras pares: " +
DistintasCifrasPares(numero).ToString());
            Console.WriteLine(numero.ToString() + " usa distintas cifras impares: " +
DistintasCifrasImpares(numero).ToString());
            Console.WriteLine(numero.ToString() + " extrayendo cifras pares: " +
NumeroSoloPares(numero).ToString());
            Console.WriteLine(numero.ToString() + " extrayendo cifras impares: " +
NumeroSoloImpares(numero).ToString());
            Console.ReadKey();
        }

        // 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;
}
}
}

```

Ejemplos de ejecución:

 C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\Ejemplo\bin\Debug\Ejemplo.exe

```

27943030 es impar: False
27943030 es par: True
27943030 total de cifras: 8
27943030 sus dos últimas cifras: 30
27943030 su antepenúltima cifra: 0
27943030 su penúltima cifra: 3
27943030 su última cifra: 0
27943030 la cifra más alta: 9
27943030 la cifra más baja: 0
27943030 total de cifras iguales a 5 es: 0
27943030 al invertirlo es: 3034972
27943030 es palíndromo: False
27943030 tercera cifra es: 9
27943030 primera cifra es: 2
27943030 la suma de las cifras es: 28
27943030 la suma de las cifras pares es: 6
27943030 la suma de las cifras impares es: 22
27943030 la multiplicación de las cifras es: 0
27943030 la multiplicación de las cifras pares es: 0
27943030 la multiplicación de las cifras impares es: 567
27943030 todas las cifras son pares: False
27943030 todas las cifras son impares: False
27943030 total cifras pares: 4
27943030 total cifras impares: 4
27943030 solo hay cifras menores o iguales a 5: False
27943030 solo hay cifras mayores o iguales a 5: False
27943030 usa distintas cifras: False
27943030 usa distintas cifras pares: False
27943030 usa distintas cifras impares: False
27943030 extrayendo cifras pares: 2400
27943030 extrayendo cifras impares: 7933

```

Ilustración 2: Trabajo con cifras

```
19720626 es impar: False
19720626 es par: True
19720626 total de cifras: 8
19720626 sus dos últimas cifras: 26
19720626 su antepenúltima cifra: 6
19720626 su penúltima cifra: 2
19720626 su última cifra: 6
19720626 la cifra más alta: 9
19720626 la cifra más baja: 0
19720626 total de cifras iguales a 5 es: 0
19720626 al invertirlo es: 62602791
19720626 es palíndromo: False
19720626 tercera cifra es: 7
19720626 primera cifra es: 1
19720626 la suma de las cifras es: 33
19720626 la suma de las cifras pares es: 16
19720626 la suma de las cifras impares es: 17
19720626 la multiplicación de las cifras es: 0
19720626 la multiplicación de las cifras pares es: 0
19720626 la multiplicación de las cifras impares es: 63
19720626 todas las cifras son pares: False
19720626 todas las cifras son impares: False
19720626 total cifras pares: 5
19720626 total cifras impares: 3
19720626 solo hay cifras menores o iguales a 5: False
19720626 solo hay cifras mayores o iguales a 5: False
19720626 usa distintas cifras: False
19720626 usa distintas cifras pares: False
19720626 usa distintas cifras impares: True
19720626 extrayendo cifras pares: 20626
19720626 extrayendo cifras impares: 197
```

Ilustración 3: Trabajo con cifras

Cadenas (strings)

Las cadenas (strings) son una de las estructuras con gran uso en la informática y su importancia está libre de toda discusión (por ejemplo, el código HTML, que es una cadena, es interpretado por el navegador y de esa forma el usuario final ve una página web). En C# hay un rico conjunto de funciones para manipular cadenas. Cabe recordar que la primera letra de una cadena empieza en la posición cero (0). Otro asunto para recordar es que si envía una cadena a una función como parámetro y la función cambia ese parámetro, la cadena NO cambia desde donde fue llamada la función, es decir, es un paso por valor.

Declaración de cadenas

002.cs

```
using System;

namespace Ejemplo {
    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 + " <=0=> " + cadenaB;
            Console.WriteLine(cadenaC);

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

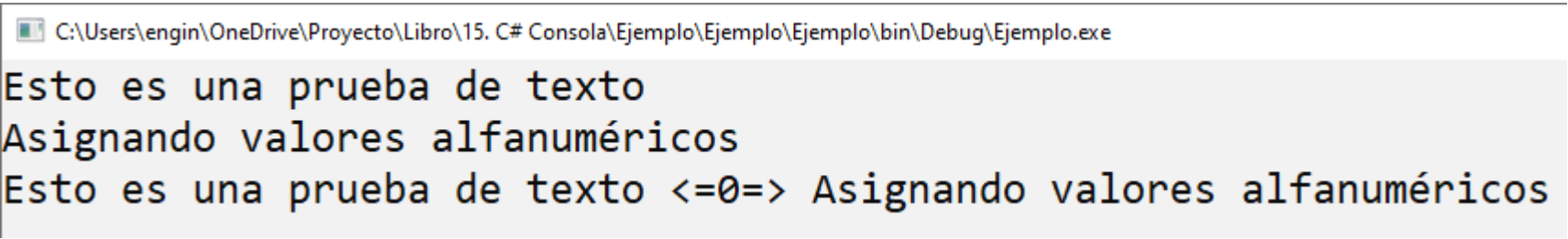


Ilustración 4: Declaración de cadenas


```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Rutas. Forma antigua
            string RutaA = "C:\\Users\\engin\\OneDrive\\Proyecto\\Libro\\16. C# Estructuras";

            //Ruta. Forma moderna
            string RutaB = @"C:\Users\engin\OneDrive\Proyecto\Libro\16. C# Estructuras";

            //Imprimiendo por consola
            Console.WriteLine(RutaA);
            Console.WriteLine(RutaB);

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

 C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\Ejemplo\bin\Debug\Ejemplo.exe

```
C:\Users\engin\OneDrive\Proyecto\Libro\16. C# Estructuras
C:\Users\engin\OneDrive\Proyecto\Libro\16. C# Estructuras
```

Ilustración 5: Uso de @ en cadenas


```
using System;

namespace Ejemplo {
    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);

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

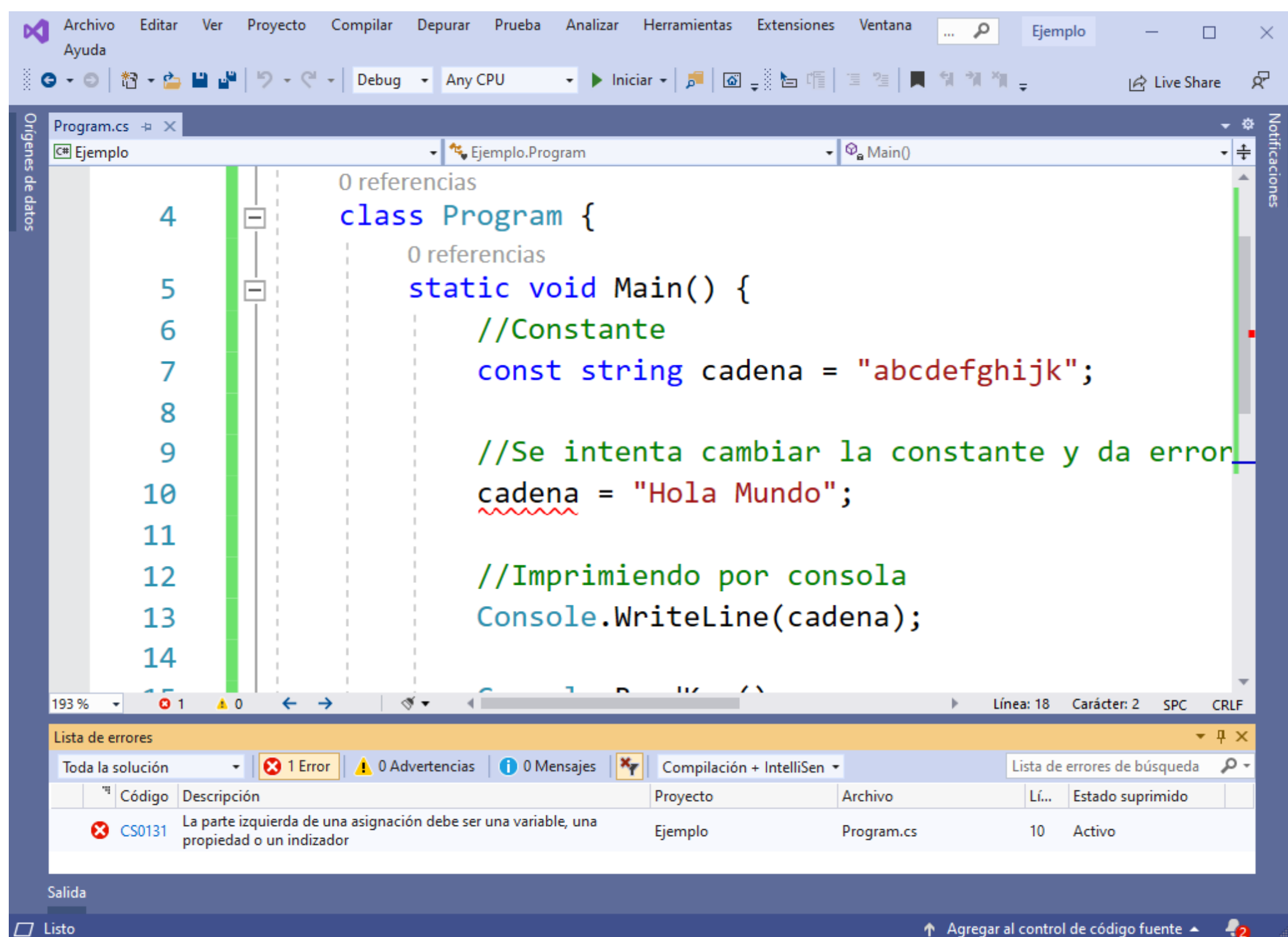


Ilustración 6: Error al cambiar una constante

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 (=).

005.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Inmutabilidad
            string cadenaA = "abcdefghijkl";
            string cadenaB = cadenaA; //Se copian los datos de cadenaA en cadenaB

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

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

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

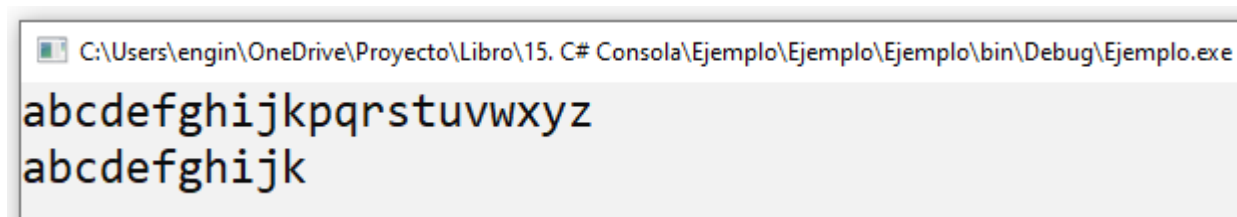


Ilustración 7: Copia de cadenas

Caracteres especiales

Los caracteres especiales no se imprimen, sino que generan otro comportamiento al intentar “imprimirse”. Esto puede ser usado a nuestro favor.

006.cs

```
using System;

namespace Ejemplo {
    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);

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

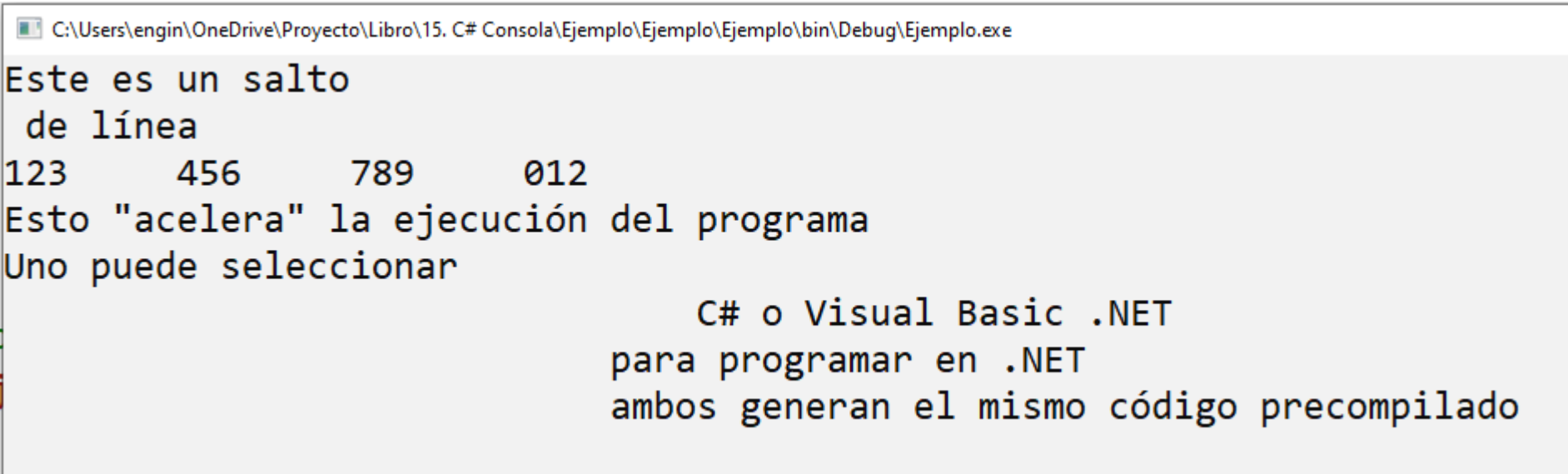


Ilustración 8: Caracteres especiales

```
using System;

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

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

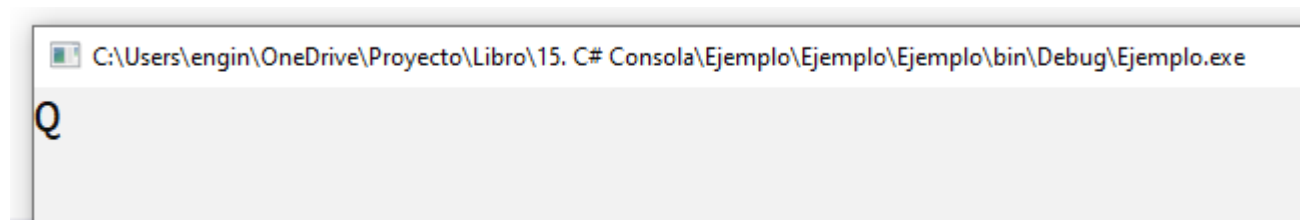


Ilustración 9: Accede a la primera letra

Tamaño de la cadena y recorrerla

Se hace uso de la instrucción Length. Tener cuidado porque esta instrucción retorna el total de caracteres de una cadena, pero si piensa recorrer esa cadena debe empezar desde la posición cero(0) hasta uno menos de la longitud (Length -1).

008.cs

```
using System;

namespace Ejemplo {
    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() + " ; ");
            }

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

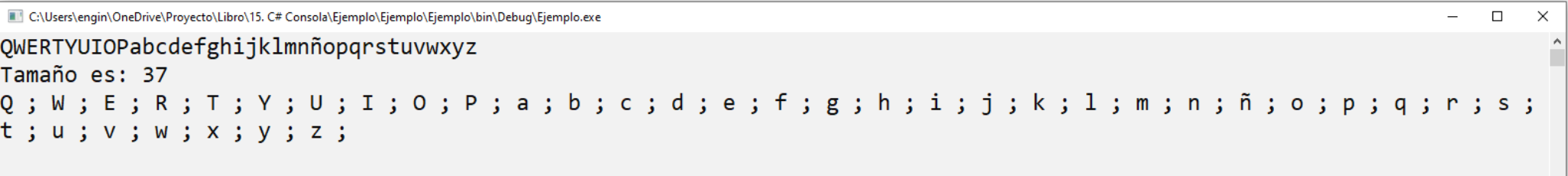


Ilustración 10: Recorrer la cadena

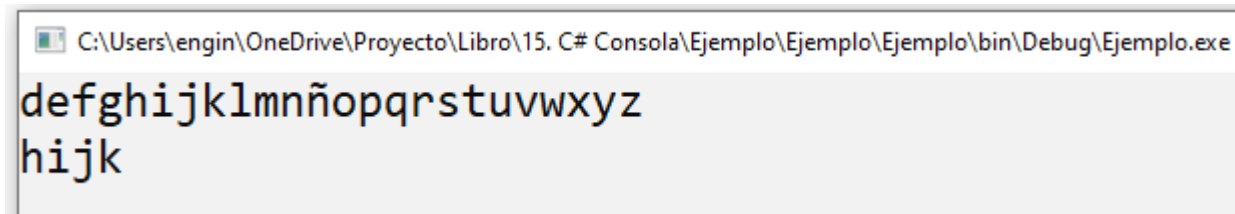
```
using System;

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

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

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

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



```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\bin\Debug\Ejemplo.exe
defghijklmnñopqrstuvwxyz
hijk
```

Ilustración 11: Subcadenas

```
using System;

namespace Ejemplo {
    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);

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

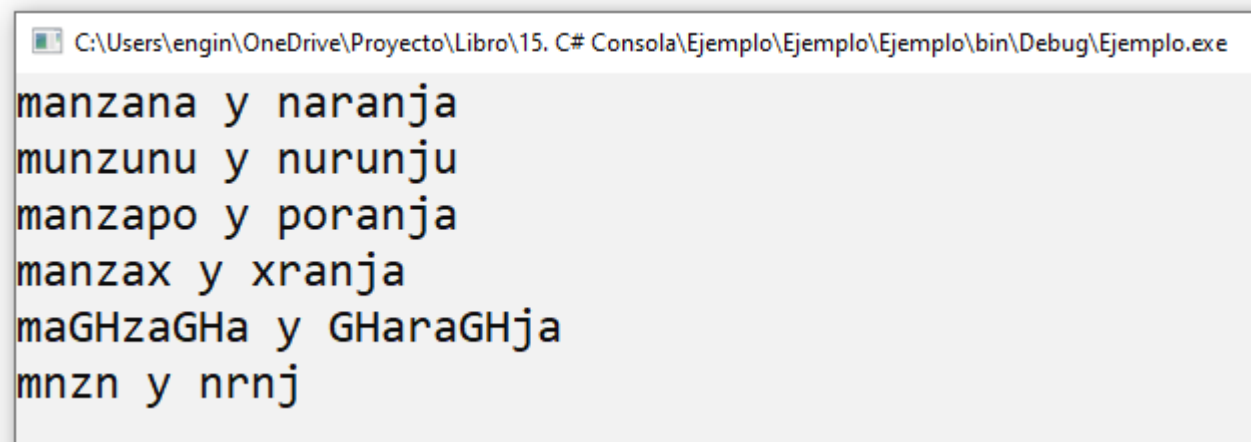


Ilustración 12: Reemplazo de cadenas y caracteres

```
using System;

namespace Ejemplo {
    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.ToString());

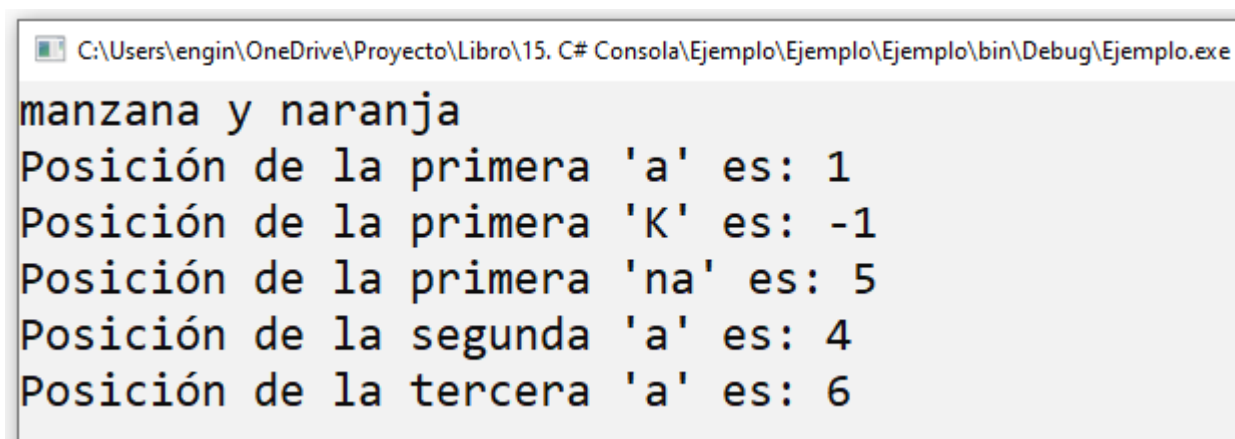
            //Busca una letra que no existe
            int posB = cadena.IndexOf('K');
            Console.WriteLine("Posición de la primera 'K' es: " + posB.ToString());

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

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

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

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



```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\Ejemplo\bin\Debug\Ejemplo.exe
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 13: Ubicación de caracteres


```
using System;

namespace Ejemplo {
    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);

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

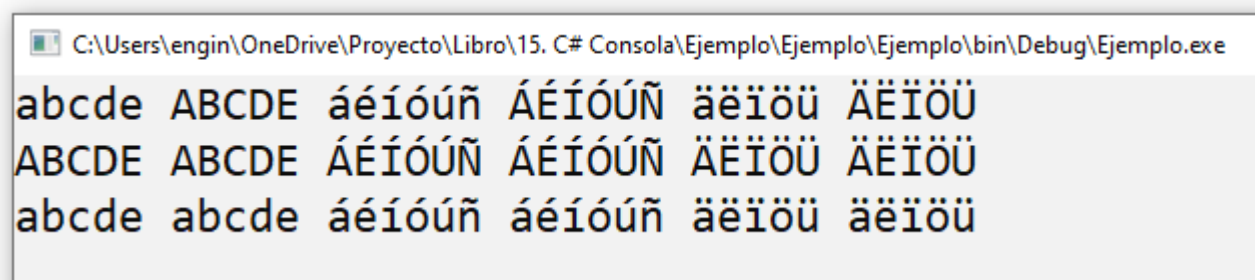


Ilustración 14: Convertir a mayúsculas y minúsculas

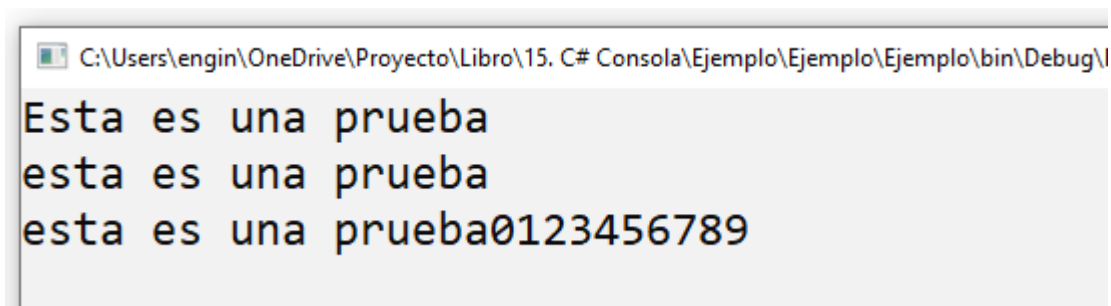
```
using System;
using System.Text; //Requiere para StringBuilder

namespace Ejemplo {
    class Program {
        static void Main() {
            //StringBuilder, mayor velocidad y se puede modificar
            StringBuilder cadenaRapida = new StringBuilder("Esta es una prueba");
            Console.WriteLine(cadenaRapida.ToString());

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

            //Agrega caracteres
            for (int numero=0; numero <= 9; numero++) {
                cadenaRapida.Append(numero.ToString());
            }
            Console.WriteLine(cadenaRapida.ToString());

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

*Ilustración 15: StringBuilder*

```
using System;
using System.Diagnostics; //Requiere para medir tiempos
using System.Text; //Requiere para StringBuilder

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

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

            //Agrega caracteres a un StringBuilder
            cronometro.Reset();
            cronometro.Start();
            for (int numero=0; numero <= 20000; numero++) {
                cadenaRapida.Append(numero.ToString());
            }
            long TBuilder = cronometro.ElapsedMilliseconds;

            //Agrega caracteres a un string
            cronometro.Reset();
            cronometro.Start();
            for (int numero = 0; numero <= 20000; numero++) {
                cadenaClasica += numero.ToString();
            }
            long TClasica = cronometro.ElapsedMilliseconds;

            Console.WriteLine("Tiempo en milisegundos StringBuilder: " + TBuilder.ToString());
            Console.WriteLine("Tiempo en milisegundos cadena clásica: " + TClasica.ToString());
            Console.WriteLine("Tamaño StringBuilder: " + cadenaRapida.Length);
            Console.WriteLine("Tamaño cadena clásica: " + cadenaClasica.Length);

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

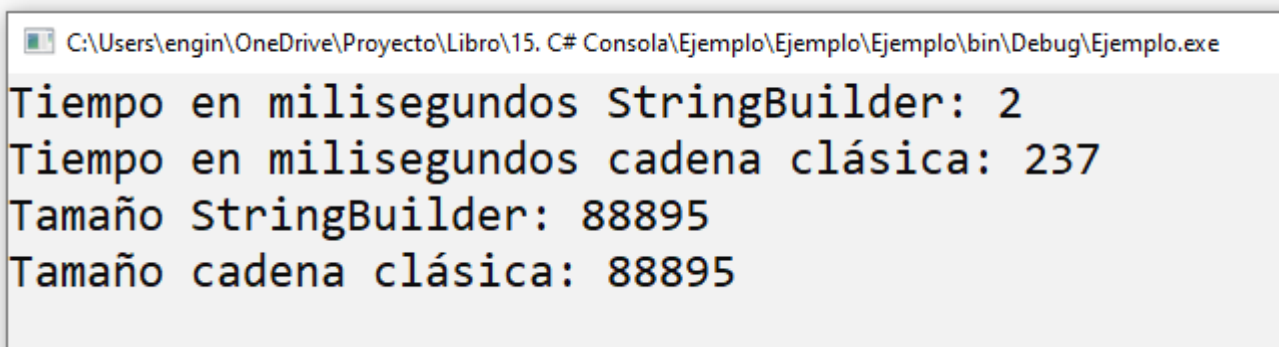


Ilustración 16: Prueba de desempeño en formato "debug"

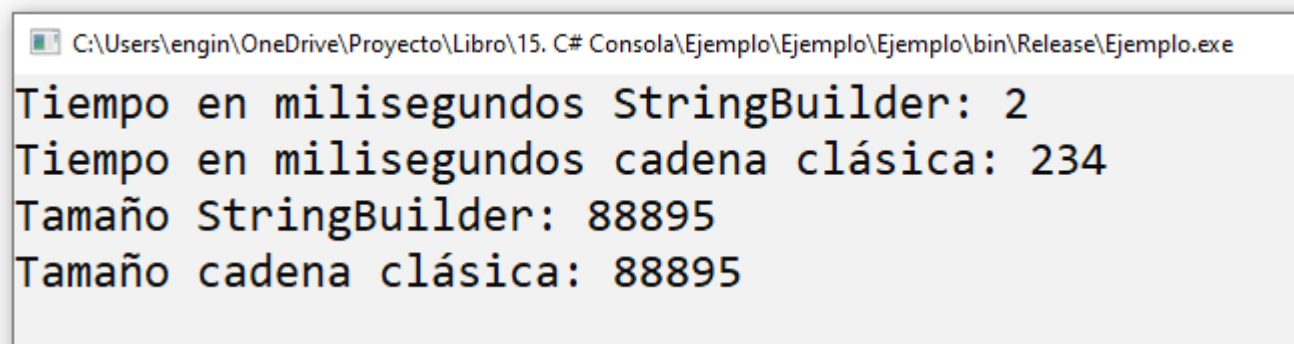


Ilustración 17: Prueba de desempeño en formato "release"

StringBuilder es más rápido que string.

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Cortando espacios
            string cadenaA = "    espacios al inicio y al final    ";
            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 al inicio y final\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 CHARACTER AL INICIO Y FINALaaaaaaaaaaaaa";
            Console.WriteLine "[" + cadenaA + "]");
            string cadenaG = cadenaA.Trim('a');
            Console.WriteLine "[" + cadenaG + "]");

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

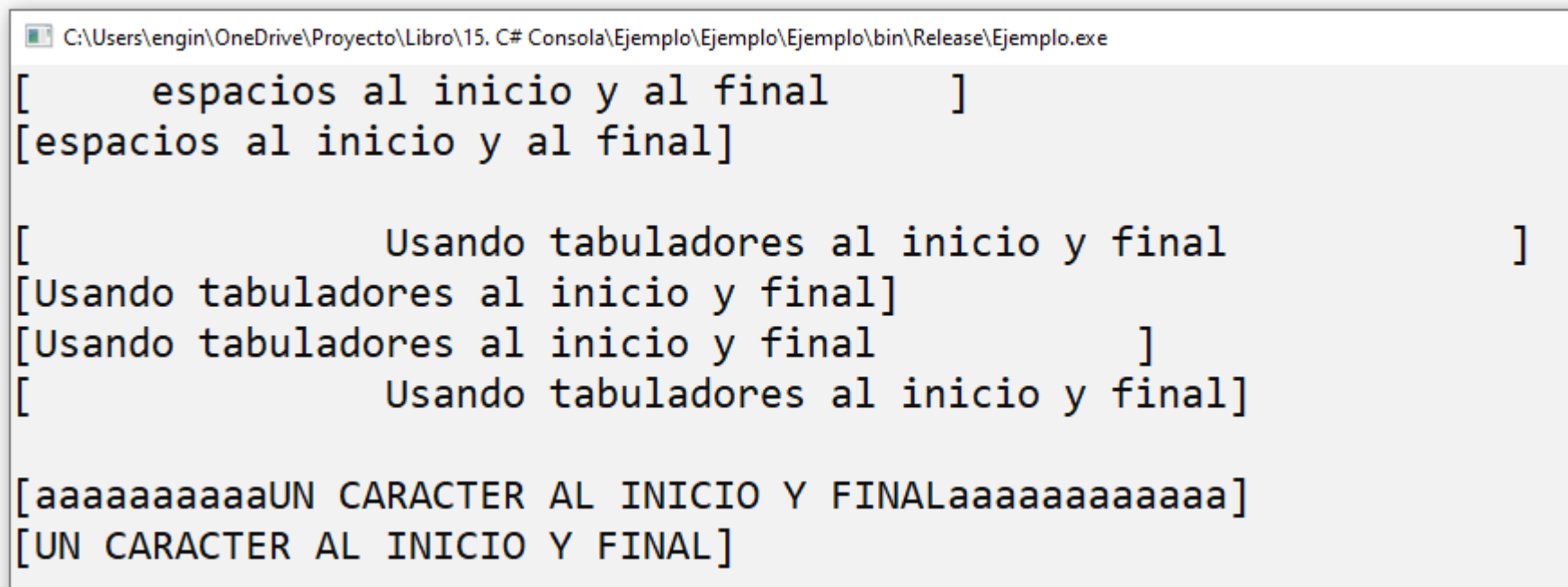


Ilustración 18: Eliminando caracteres al inicio y al final

Comparar cadenas. Forma no recomendada.

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

016.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Comparar cadenas
            string cadenaA = "abcdefghijkl";
            string cadenaB = "Abcdefghij";
            string cadenaC = "abcdefghijkl ";
            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");

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

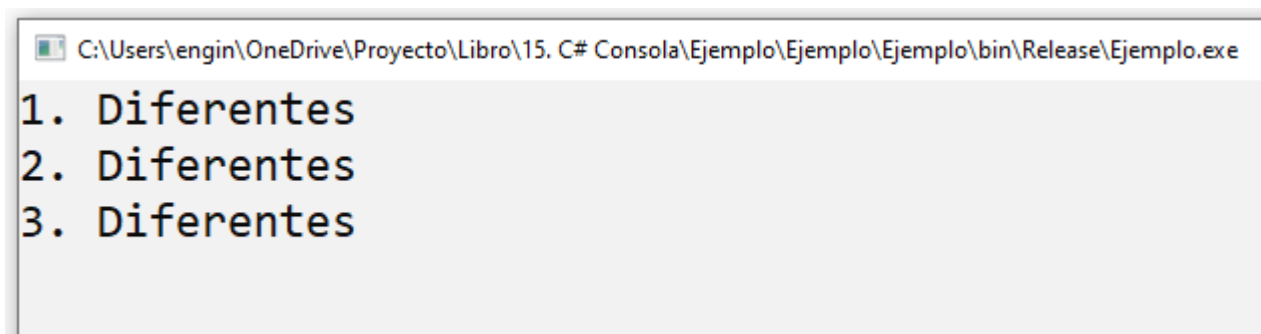


Ilustración 19: Comparación de cadenas. No recomendado.

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Comparar cadenas
            string cadenaA = "abcdefghijkl";
            string cadenaB = "Abcdefghij";
            string cadenaC = "abcdefghijkl ";
            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");

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

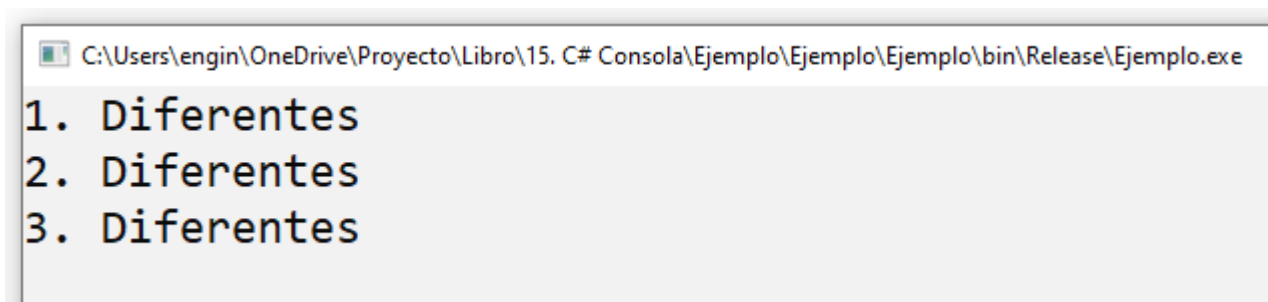


Ilustración 20: Comparación de cadenas. Forma recomendada.

```
using System;

namespace Ejemplo {
    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");

            Console.ReadKey();
        }
    }
}

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

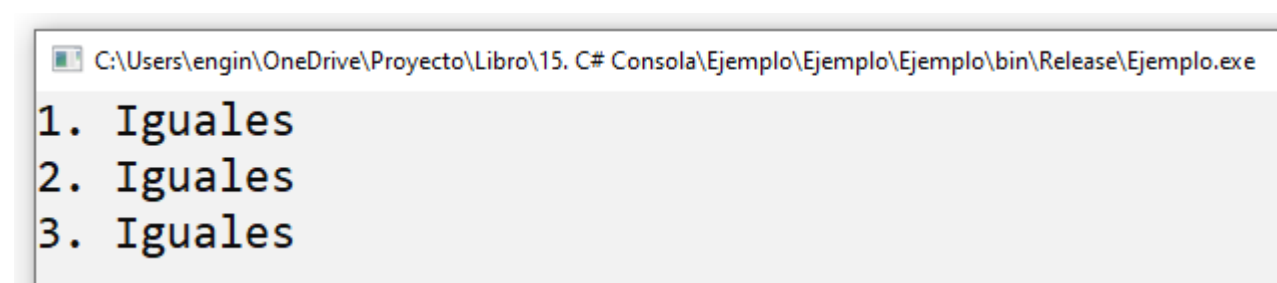


Ilustración 21: Comparación de cadenas. Forma recomendada.

Arreglos Unidimensionales

iOJO! Los arreglos unidimensionales empiezan en la posición 0.

Declaración y asignar valores

019.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Arreglo unidimensional. Definición.
            int[] arreglo = new int[5];

            //Asignando valores a cada posición del arreglo
            arreglo[0] = 891;
            arreglo[1] = 302;
            arreglo[2] = 465;
            arreglo[3] = 743;
            arreglo[4] = 847;

            //Imprimiendo valores
            Console.WriteLine("En posición 0 el valor es " + arreglo[0].ToString());
            Console.WriteLine("En posición 1 el valor es " + arreglo[1].ToString());
            Console.WriteLine("En posición 2 el valor es " + arreglo[2].ToString());
            Console.WriteLine("En posición 3 el valor es " + arreglo[3].ToString());
            Console.WriteLine("En posición 4 el valor es " + arreglo[4].ToString());

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

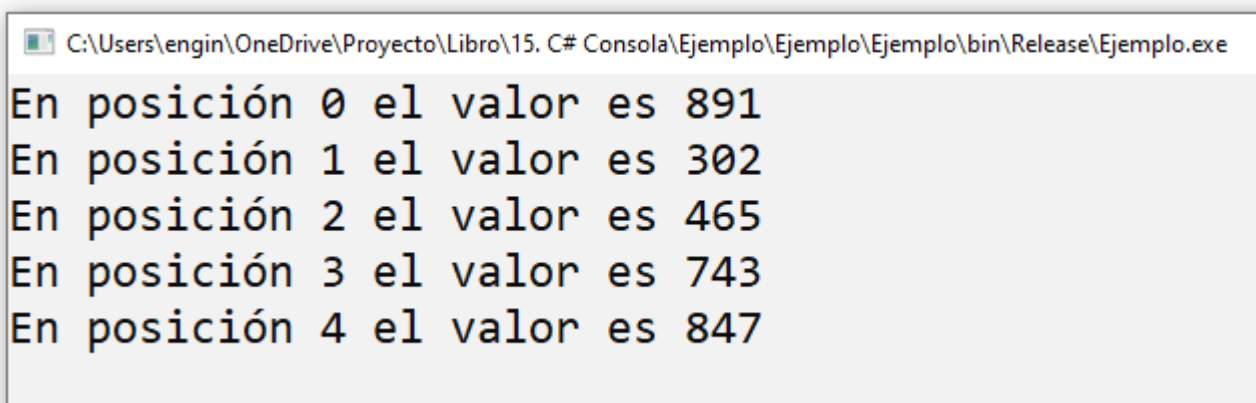


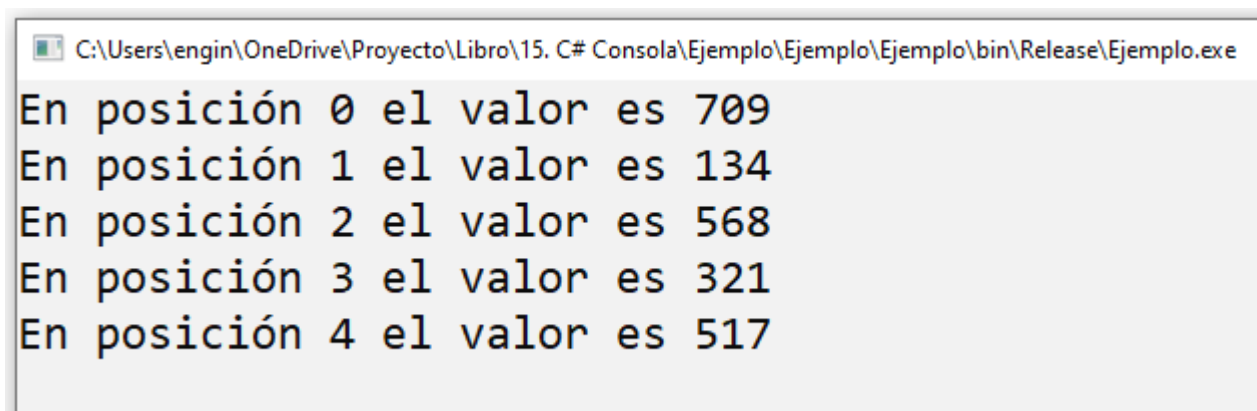
Ilustración 22: Arreglo unidimensional

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Arreglo unidimensional. Definición y asignación.
            int[] arreglo = { 709, 134, 568, 321, 517 };

            //Imprimiendo valores
            Console.WriteLine("En posición 0 el valor es " + arreglo[0].ToString());
            Console.WriteLine("En posición 1 el valor es " + arreglo[1].ToString());
            Console.WriteLine("En posición 2 el valor es " + arreglo[2].ToString());
            Console.WriteLine("En posición 3 el valor es " + arreglo[3].ToString());
            Console.WriteLine("En posición 4 el valor es " + arreglo[4].ToString());

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



```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\bin\Release\Ejemplo.exe
En posición 0 el valor es 709
En posición 1 el valor es 134
En posición 2 el valor es 568
En posición 3 el valor es 321
En posición 4 el valor es 517
```

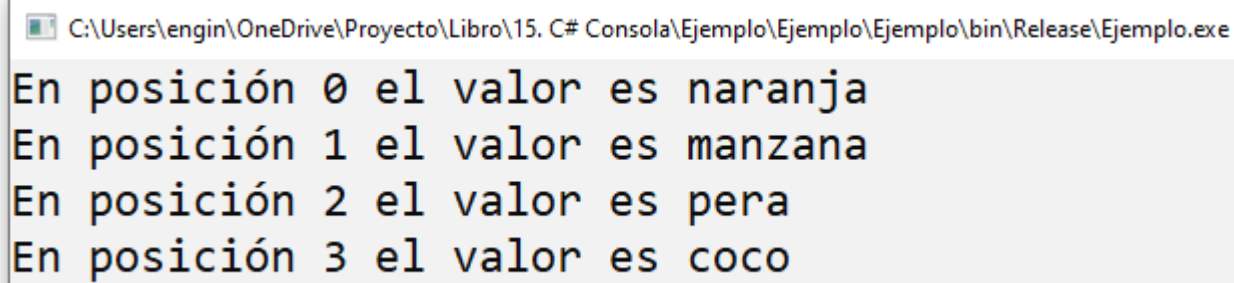
Ilustración 23: Arreglo unidimensional

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Arreglo unidimensional de cadenas.
            string[] cadenas = { "naranja", "manzana", "pera", "coco" };

            //Imprimiendo valores
            Console.WriteLine("En posición 0 el valor es " + cadenas[0]);
            Console.WriteLine("En posición 1 el valor es " + cadenas[1]);
            Console.WriteLine("En posición 2 el valor es " + cadenas[2]);
            Console.WriteLine("En posición 3 el valor es " + cadenas[3]);

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



C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\bin\Release\Ejemplo.exe

En posición 0 el valor es naranja
En posición 1 el valor es manzana
En posición 2 el valor es pera
En posición 3 el valor es coco

Ilustración 24: Arreglo unidimensional de cadenas

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Arreglo unidimensional. Tamaños.
            string[] cadenas = { "naranja", "manzana", "pera", "coco" };
            int[] numeros = new int[8];

            //Tamaños de los arreglos
            int TamanoCadenas = cadenas.Length;
            int TamanoNumeros = numeros.Length;

            //Imprime
            Console.WriteLine("Tamaño de arreglo cadenas:" + TamanoCadenas.ToString());
            Console.WriteLine("Tamaño de arreglo numeros:" + TamanoNumeros.ToString());

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

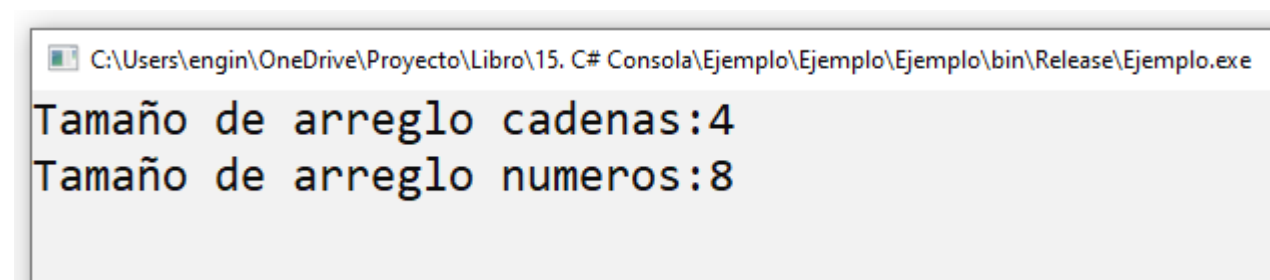


Ilustración 25: Tamaño de arreglos

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Arreglo unidimensional
            string[] cadenas = { "naranja", "manzana", "pera", "coco" };

            //Tamaño
            int TamanoCadenas = cadenas.Length;

            //Recorre el arreglo y lo imprime
            for (int posicion=0; posicion<TamanoCadenas; posicion++) {
                Console.WriteLine(cadenas[posicion]);
            }

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

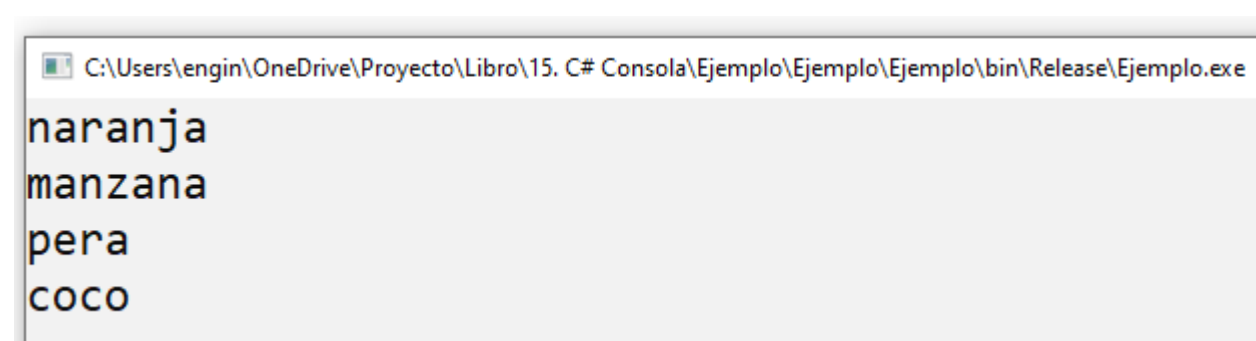


Ilustración 26: Recorrer arreglo de cadenas

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Arreglo unidimensional
            string[] cadenas = { "naranja", "manzana", "pera", "coco" };

            //Recorre el arreglo y lo imprime
            foreach(string texto in cadenas) {
                Console.WriteLine(texto);
            }

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

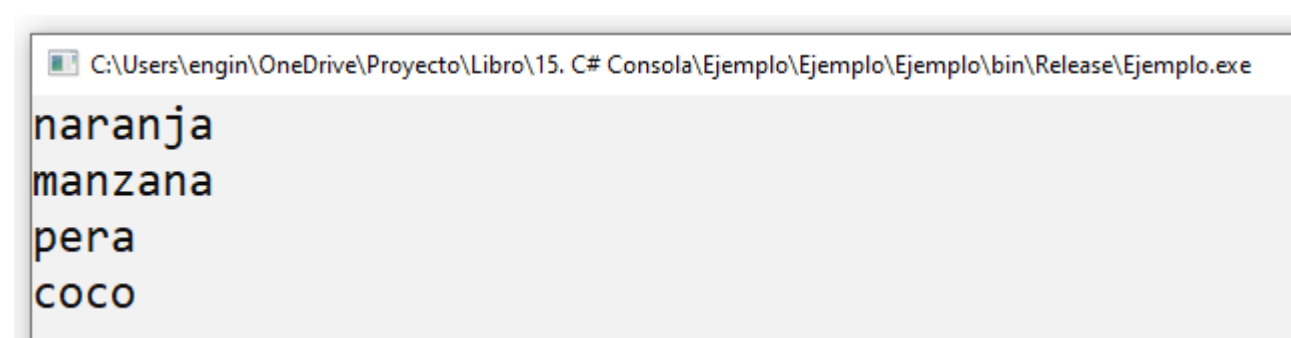


Ilustración 27: Recorrer arreglo con foreach

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Arreglo unidimensional
            string[] cadenas = { "def", "agh", "abc", "efg", "bcd", "klm", "iao", "hñu" };

            //Ordena el arreglo
            Array.Sort(cadenas);

            //Recorre el arreglo y lo imprime
            foreach(string texto in cadenas) {
                Console.WriteLine(texto);
            }

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

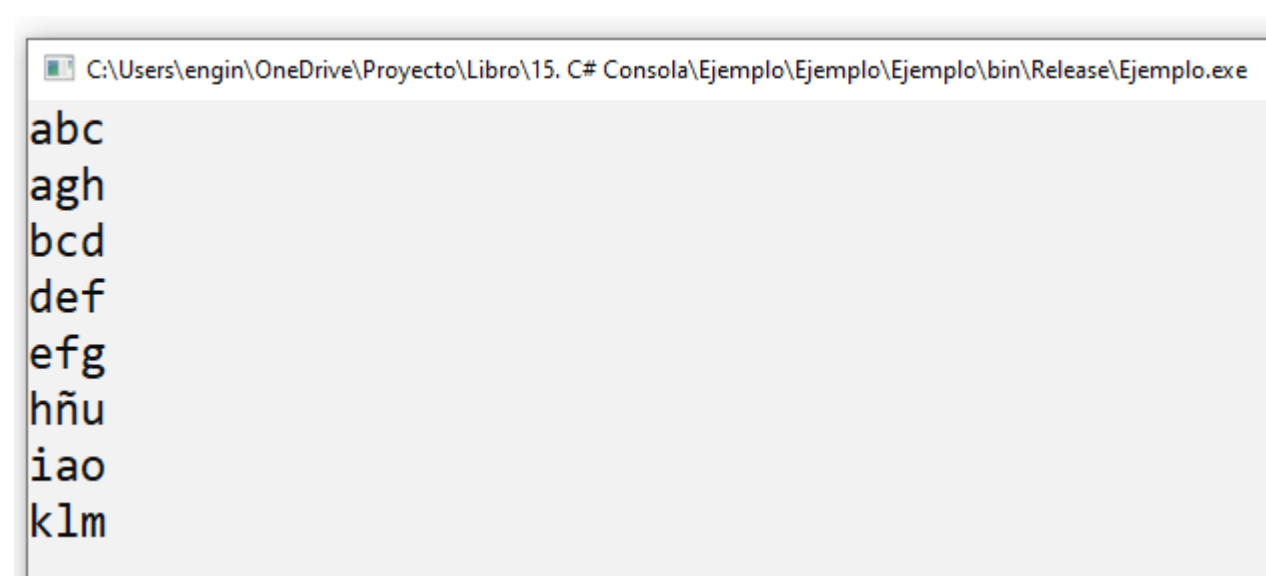


Ilustración 28: Ordenar un arreglo de cadenas


```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Arreglo unidimensional
            string[] cadenas = { "áa", "aa", "äa", "Aa", "Äa", "Áa", "aaa", "aáa", "aAa", "aÁa" };

            //Ordena el arreglo
            Array.Sort(cadenas);

            //Recorre el arreglo y lo imprime
            foreach(string texto in cadenas) {
                Console.WriteLine(texto);
            }

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

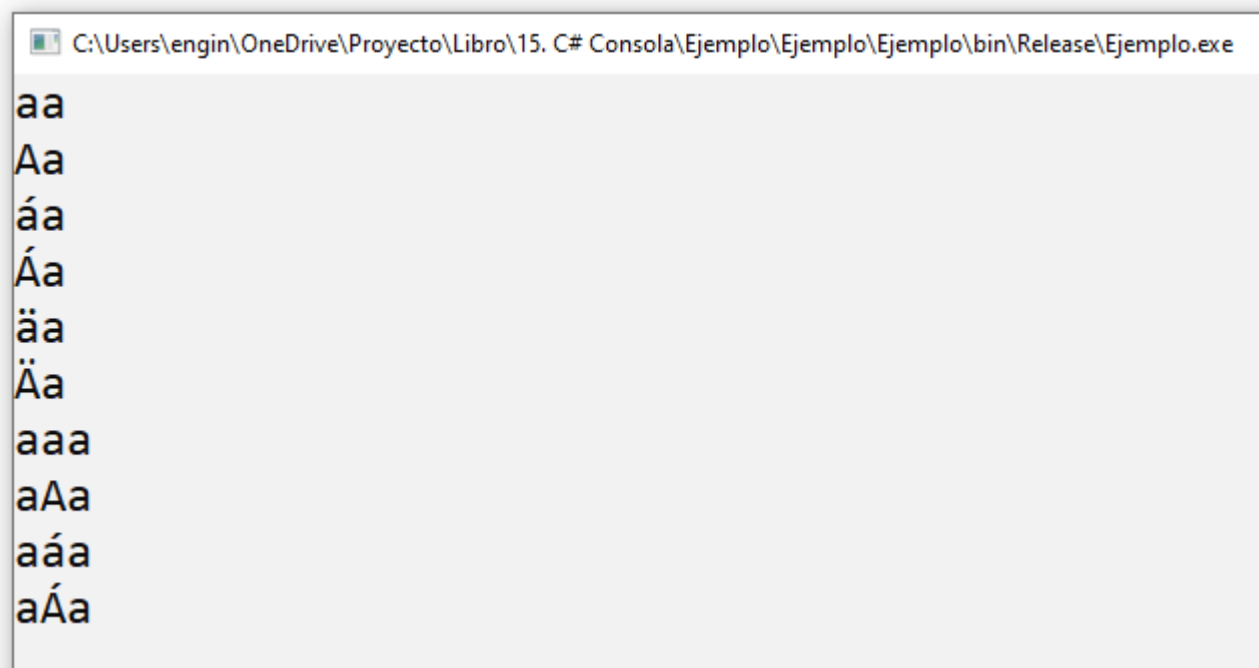


Ilustración 29: Ordenando una cadena con tildes, mayúsculas, diéresis

En caso de probar las tildes, diéresis, mayúsculas y minúsculas, se obtendría este resultado con cadenas de una sola letra.

```
string[] cadenas = { "á", "é", "í", "ó", "ú", "Á", "É", "Í", "Ó", "Ú", "A", "E", "I", "O", "U", "a", "e",  
"i", "o", "u", "Ä", "Ë", "Ï", "Ö", "Ü", "ä", "ë", "ï", "ö", "ü" };
```

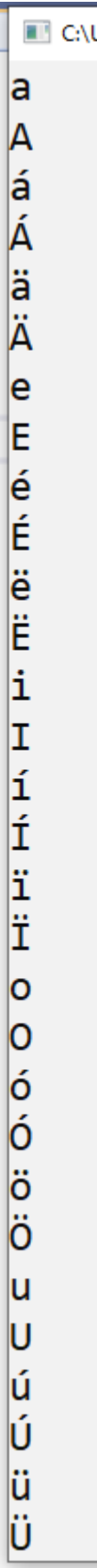


Ilustración 30: Ordenamiento con cadenas de una sola letra y uso de tildes, diéresis, mayúsculas y minúsculas

```
using System;
using System.Linq; //Requiere esta librería

namespace Ejemplo {
    class Program {
        static void Main() {
            //Arreglo unidimensional
            int[] numeros = { 27, 18, 52, 42, 90, 12 };

            //Máximo, mínimo, suma
            int maximo = numeros.Max();
            int minimo = numeros.Min();
            int suma = numeros.Sum();

            //Imprime
            Console.WriteLine("Máximo: " + maximo.ToString());
            Console.WriteLine("Mínimo: " + minimo.ToString());
            Console.WriteLine("Suma: " + suma.ToString());

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

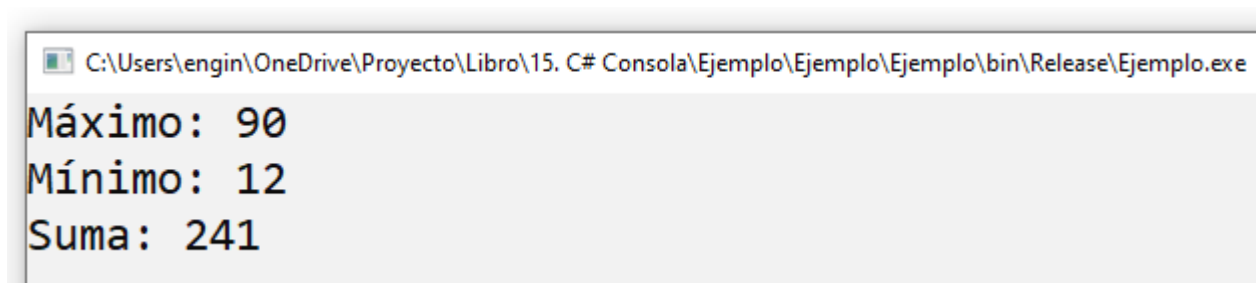


Ilustración 31: Máximo, mínimo, suma usando Linq

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Arreglo unidimensional. Funciones genéricas
            int[] numeros = new int[50];

            //Llena con valores al azar
            int minimo = 1, maximo = 30;
            LlenaArreglo(numeros, minimo, maximo);
            ImprimeArreglo(numeros);

            //Busca un valor determinado y retorna su posición
            int valorBusca = 17;
            int encuentra = PosArregloDato(numeros, valorBusca);
            if (encuentra == -1)
                Console.WriteLine("Valor no encontrado");
            else
                Console.WriteLine("Valor " + valorBusca.ToString() + " encontrado en posición: " +
encuentra.ToString());

            Console.ReadKey();
        }

        //Llena el arreglo con valores al azar entre min y max (ambos incluidos)
        static void LlenaArreglo(int[] arreglo, int min, int max) {
            Random azar = new Random();
            for (int posicion = 0; posicion < arreglo.Length; posicion++) {
                arreglo[posicion] = azar.Next(min, max+1);
            }
        }

        //Imprime el arreglo en consola
        static void ImprimeArreglo(int [] arreglo) {
            for (int posicion = 0; posicion < arreglo.Length; posicion++) {
                Console.Write(arreglo[posicion].ToString() + " ; ");
            }
            Console.WriteLine(" ");
        }

        //Retorna la posición del dato en el arreglo o retorna -1 si no lo encuentra
        static int PosArregloDato(int[] arreglo, int valor) {
            for (int posicion = 0; posicion < arreglo.Length; posicion++) {
                if (arreglo[posicion] == valor)
                    return posicion;
            }
            return -1;
        }
    }
}
```

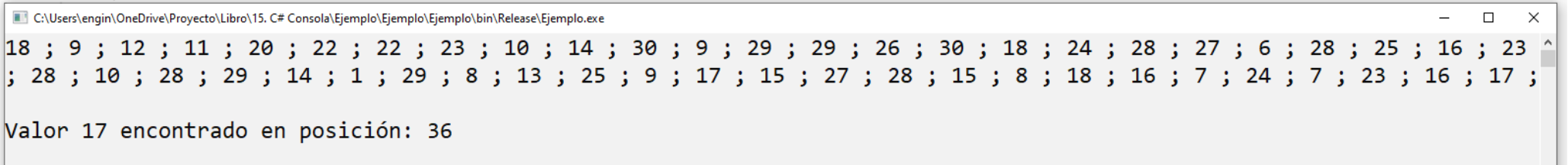


Ilustración 32: Funciones genéricas para arreglos unidimensionales

Se prueban diversos algoritmos de ordenación. Obsérvese que se envían los arreglos como parámetros y estos arreglos son modificados por las funciones mismas. Eso significa que, al enviar un arreglo como parámetro a una función o procedimiento, este es enviado por referencia (**no se copian los valores** como sucede con los tipos de datos nativos como int, double, float, string). Luego se debe tener especial cuidado con esto porque si la función o procedimiento modifica el arreglo, quedaría modificado también desde el sitio en que se llamó a la función o procedimiento.

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Ordenación
            int Limite = 35;
            int[] numA = new int[Limite];
            int[] numB = new int[Limite];

            int minimo = 0, maximo = 9;
            LlenaArreglo(numB, minimo, maximo);

            //Ordena por inserción
            Console.WriteLine("\nOrdena por inserción");
            Array.Copy(numB, 0, numA, 0, numB.Length); //Copia arreglo numB ==> numA
            ImprimeArreglo(numA);
            Insercion(numA);
            ImprimeArreglo(numA);

            //Ordena por selección
            Console.WriteLine("\nOrdena por selección");
            Array.Copy(numB, 0, numA, 0, numB.Length);
            ImprimeArreglo(numA);
            Seleccion(numA);
            ImprimeArreglo(numA);

            //Ordena por burbuja
            Console.WriteLine("\nOrdena por burbuja");
            Array.Copy(numB, 0, numA, 0, numB.Length);
            ImprimeArreglo(numA);
            Burbuja(numA);
            ImprimeArreglo(numA);

            //Ordena por shell
            Console.WriteLine("\nOrdena por shell");
            Array.Copy(numB, 0, numA, 0, numB.Length);
            ImprimeArreglo(numA);
            Shell(numA);
            ImprimeArreglo(numA);

            //Ordena por QuickSort
            Console.WriteLine("\nOrdena por quicksort");
            Array.Copy(numB, 0, numA, 0, numB.Length);
            ImprimeArreglo(numA);
            QuickSort(numA, 0, numA.Length - 1);
            ImprimeArreglo(numA);

            Console.ReadKey();
        }

        //Llena el arreglo con valores al azar entre min y max (ambos incluidos)
        static void LlenaArreglo(int[] arreglo, int min, int max) {
            Random azar = new Random();
            for (int posicion = 0; posicion < arreglo.Length; posicion++) {
                arreglo[posicion] = azar.Next(min, max+1);
            }
        }

        //Imprime el arreglo en consola
        static void ImprimeArreglo(int [] arreglo) {
            for (int posicion = 0; posicion < arreglo.Length; posicion++) {
                Console.Write(arreglo[posicion].ToString() + ";");
            }
            Console.WriteLine(" ");
        }

        //Ordenamiento por Insert
        static void Insercion(int[] arreglo) {
            int tmp;
            int j;
        }
    }
}
```

```

        for (int i = 1; i < arreglo.Length; i++) {
            tmp = arreglo[i];
            for (j = i - 1; j >= 0 && arreglo[j] > tmp; j--) {
                arreglo[j + 1] = arreglo[j];
            }
            arreglo[j + 1] = tmp;
        }
    }

//Ordenamiento por Selección
static void Seleccion(int[] arreglo) {
    for (int i = 0; i < arreglo.Length - 1; i++) {
        int min = i;
        for (int j = i + 1; j < arreglo.Length; j++) {
            if (arreglo[j] < arreglo[min]) {
                min = j;
            }
        }
        if (i != min) {
            int aux = arreglo[i];
            arreglo[i] = arreglo[min];
            arreglo[min] = aux;
        }
    }
}

//Ordenamiento por Burbuja
static void Burbuja(int[] Arreglo) {
    int n = Arreglo.Length;
    int tmp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - 1; j++) {
            if (Arreglo[j] > Arreglo[j + 1]) {
                tmp = Arreglo[j];
                Arreglo[j] = Arreglo[j + 1];
                Arreglo[j + 1] = tmp;
            }
        }
    }
}

//Ordenamiento por Shell
static void Shell(int[] arreglo) {
    int N = arreglo.Length;
    int incremento = N;
    do {
        incremento = incremento / 2;
        for (int k = 0; k < incremento; k++) {
            for (int i = incremento + k; i < N; i += incremento) {
                int j = i;
                while (j - incremento >= 0 && arreglo[j] < arreglo[j - incremento]) {
                    int tmp = arreglo[j];
                    arreglo[j] = arreglo[j - incremento];
                    arreglo[j - incremento] = tmp;
                    j -= incremento;
                }
            }
        }
    } while (incremento > 1);
}

//Ordenación por QuickSort
static void QuickSort(int[] vector, int primero, int ultimo) {
    int i, j, central;
    double pivote;
    central = (primero + ultimo) / 2;
    pivote = vector[central];
    i = primero;
    j = ultimo;
    do {
        while (vector[i] < pivote) i++;
        while (vector[j] > pivote) j--;
        if (i <= j) {
            int temp;
            temp = vector[i];
            vector[i] = vector[j];
            vector[j] = temp;
            i++;
            j--;
        }
    }
}

```

```

        } while (i <= j);

        if (primero < j) {
            QuickSort(vector, primero, j);
        }
        if (i < ultimo) {
            QuickSort(vector, i, ultimo);
        }
    }
}
}

```

C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\Ejemplo\bin\Release\Ejemplo.exe

Ordena por inserción

2;8;1;0;3;2;6;6;5;1;9;7;0;8;2;3;5;5;0;8;1;5;6;5;6;4;9;5;6;2;1;1;5;0;6;
0;0;0;0;1;1;1;1;1;2;2;2;2;3;3;4;5;5;5;5;5;5;5;5;6;6;6;6;6;6;7;8;8;8;9;9;

Ordena por selección

2;8;1;0;3;2;6;6;5;1;9;7;0;8;2;3;5;5;0;8;1;5;6;5;6;4;9;5;6;2;1;1;5;0;6;
0;0;0;0;1;1;1;1;1;2;2;2;2;3;3;4;5;5;5;5;5;5;5;5;6;6;6;6;6;6;7;8;8;8;9;9;

Ordena por burbuja

2;8;1;0;3;2;6;6;5;1;9;7;0;8;2;3;5;5;0;8;1;5;6;5;6;4;9;5;6;2;1;1;5;0;6;
0;0;0;0;1;1;1;1;1;2;2;2;2;3;3;4;5;5;5;5;5;5;5;5;6;6;6;6;6;6;7;8;8;8;9;9;

Ordena por shell

2;8;1;0;3;2;6;6;5;1;9;7;0;8;2;3;5;5;0;8;1;5;6;5;6;4;9;5;6;2;1;1;5;0;6;
0;0;0;0;1;1;1;1;1;2;2;2;2;3;3;4;5;5;5;5;5;5;5;5;6;6;6;6;6;6;7;8;8;8;9;9;

Ordena por quicksort

2;8;1;0;3;2;6;6;5;1;9;7;0;8;2;3;5;5;0;8;1;5;6;5;6;4;9;5;6;2;1;1;5;0;6;
0;0;0;0;1;1;1;1;1;2;2;2;2;3;3;4;5;5;5;5;5;5;5;5;6;6;6;6;6;6;7;8;8;8;9;9;

Ilustración 33: Diversos algoritmos de ordenación

```

using System;
using System.Diagnostics;

namespace Ejemplo {
    class Program {
        static void Main() {
            int Limite = 18000;
            int[] numerosA = new int[Limite];
            int[] numerosB = new int[Limite];

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

            long TPShell = 0, TPIns = 0, TPSel = 0, TPBur = 0, TPQuick = 0;

            //Para disminuir oscilaciones en el tiempo, se hacen
            //N pruebas con cada grupo de pruebas
            int TotalPruebas = 5;
            int minimo = 10, maximo = 300;
            for (int prueba = 1; prueba <= TotalPruebas; prueba++) {
                LlenaArreglo(numerosA, minimo, maximo);

                //Ordenación por método Shell
                Array.Copy(numerosA, 0, numerosB, 0, numerosA.Length);
                cronometro.Reset();
                cronometro.Start();
                Shell(numerosB);
                TPShell += cronometro.ElapsedMilliseconds;

                //Ordenación por método Inserción
                Array.Copy(numerosA, 0, numerosB, 0, numerosA.Length);
                cronometro.Reset();
                cronometro.Start();
                Insercion(numerosB);
                TPIns += cronometro.ElapsedMilliseconds;

                //Ordenación por método Selección
                Array.Copy(numerosA, 0, numerosB, 0, numerosA.Length);
                cronometro.Reset();
                cronometro.Start();
                Seleccion(numerosB);
                TPSel += cronometro.ElapsedMilliseconds;

                //Ordenación por método Burbuja
                Array.Copy(numerosA, 0, numerosB, 0, numerosA.Length);
                cronometro.Reset();
                cronometro.Start();
                Burbuja(numerosB);
                TPBur += cronometro.ElapsedMilliseconds;

                //Ordenación por método QuickSort
                Array.Copy(numerosA, 0, numerosB, 0, numerosA.Length);
                cronometro.Reset();
                cronometro.Start();
                QuickSort(numerosB, 0, numerosB.Length - 1);
                TPQuick += cronometro.ElapsedMilliseconds;
            }

            double TS = (double)TPShell / TotalPruebas;
            double TI = (double)TPIns / TotalPruebas;
            double TL = (double)TPSel / TotalPruebas;
            double TB = (double)TPBur / TotalPruebas;
            double TQ = (double)TPQuick / TotalPruebas;

            Console.WriteLine("=====");
            Console.WriteLine("Número de elementos: " + Limite.ToString());
            Console.WriteLine("ShellSort, tiempo promedio en milisegundos: " + TS.ToString());
            Console.WriteLine("InsertSort, tiempo promedio en milisegundos: " + TI.ToString());
            Console.WriteLine("Seleccion, tiempo promedio en milisegundos: " + TL.ToString());
            Console.WriteLine("Burbuja, tiempo promedio en milisegundos: " + TB.ToString());
            Console.WriteLine("QuickSort, tiempo promedio en milisegundos: " + TQ.ToString());

            Console.ReadKey();
        }

        //Llena el arreglo con valores al azar entre min y max (ambos incluidos)
        static void LlenaArreglo(int[] arreglo, int min, int max) {

```



```

Random azar = new Random();
for (int posicion = 0; posicion < arreglo.Length; posicion++) {
    arreglo[posicion] = azar.Next(min, max+1);
}

//Ordenamiento por Insert
static void Insercion(int[] arreglo) {
    int tmp;
    int j;
    for (int i = 1; i < arreglo.Length; i++) {
        tmp = arreglo[i];
        for (j = i - 1; j >= 0 && arreglo[j] > tmp; j--) {
            arreglo[j + 1] = arreglo[j];
        }
        arreglo[j + 1] = tmp;
    }
}

//Ordenamiento por Selección
static void Seleccion(int[] arreglo) {
    for (int i = 0; i < arreglo.Length - 1; i++) {
        int min = i;
        for (int j = i + 1; j < arreglo.Length; j++) {
            if (arreglo[j] < arreglo[min]) {
                min = j;
            }
        }
        if (i != min) {
            int aux = arreglo[i];
            arreglo[i] = arreglo[min];
            arreglo[min] = aux;
        }
    }
}

//Ordenamiento por Burbuja
static void Burbuja(int[] Arreglo) {
    int n = Arreglo.Length;
    int tmp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - 1; j++) {
            if (Arreglo[j] > Arreglo[j + 1]) {
                tmp = Arreglo[j];
                Arreglo[j] = Arreglo[j + 1];
                Arreglo[j + 1] = tmp;
            }
        }
    }
}

//Ordenamiento por Shell
static void Shell(int[] arreglo) {
    int N = arreglo.Length;
    int incremento = N;
    do {
        incremento = incremento / 2;
        for (int k = 0; k < incremento; k++) {
            for (int i = incremento + k; i < N; i += incremento) {
                int j = i;
                while (j - incremento >= 0 && arreglo[j] < arreglo[j - incremento]) {
                    int tmp = arreglo[j];
                    arreglo[j] = arreglo[j - incremento];
                    arreglo[j - incremento] = tmp;
                    j -= incremento;
                }
            }
        }
    } while (incremento > 1);
}

//Ordenación por QuickSort
static void QuickSort(int[] vector, int primero, int ultimo) {
    int i, j, central;
    double pivote;
    central = (primero + ultimo) / 2;
    pivote = vector[central];
    i = primero;
    j = ultimo;
    do {

```

```

while (vector[i] < pivote) i++;
while (vector[j] > pivote) j--;
if (i <= j) {
    int temp;
    temp = vector[i];
    vector[i] = vector[j];
    vector[j] = temp;
    i++;
    j--;
}
} while (i <= j);

if (primero < j) {
    QuickSort(vector, primero, j);
}
if (i < ultimo) {
    QuickSort(vector, i, ultimo);
}
}
}
}

```

```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\Ejemplo\bin\Release\Ejemplo.exe
=====
Número de elementos: 18000
ShellSort, tiempo promedio en milisegundos: 1,4
InsertSort, tiempo promedio en milisegundos: 96,4
Selección, tiempo promedio en milisegundos: 193,2
Burbuja, tiempo promedio en milisegundos: 633,8
QuickSort, tiempo promedio en milisegundos: 1
```

Ilustración 34: Tiempo ordenando 18 elementos

```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\Ejemplo\bin\Release\Ejemplo.exe
=====
Número de elementos: 28000
ShellSort, tiempo promedio en milisegundos: 2,6
InsertSort, tiempo promedio en milisegundos: 225,8
Selección, tiempo promedio en milisegundos: 456,4
Burbuja, tiempo promedio en milisegundos: 1568,2
QuickSort, tiempo promedio en milisegundos: 1,2
```

Ilustración 35: Tiempo ordenando 28000 elementos

Arreglo bidimensional

Definición

031.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            int TotalFilas = 5;
            int TotalColumnas = 10;

            //Declara un arreglo bidimensional
            int[,] Tablero = new int[TotalFilas, TotalColumnas];

            /* Llena ese arreglo bidimensional
            Tablero.GetLength(0) Retorna el número de filas (la primera dimensión)
            Tablero.GetLength(1) Retorna el número de columnas (la segunda dimensión)
            Un arreglo bidimensional inicia en [0,0]
            Un arreglo bidimensional termina en [TotalFilas-1, TotalColumnas-1]
            */
            Random azar = new Random();
            for (int fila = 0; fila < Tablero.GetLength(0); fila++)
                for (int columna = 0; columna < Tablero.GetLength(1); columna++)
                    Tablero[fila, columna] = azar.Next(0, 9);

            //Imprime ese arreglo bidimensional
            for (int fila = 0; fila < Tablero.GetLength(0); fila++) {
                Console.WriteLine(" ");
                for (int columna = 0; columna < Tablero.GetLength(1); columna++)
                    Console.Write(Tablero[fila, columna].ToString() + " ; ");
            }

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

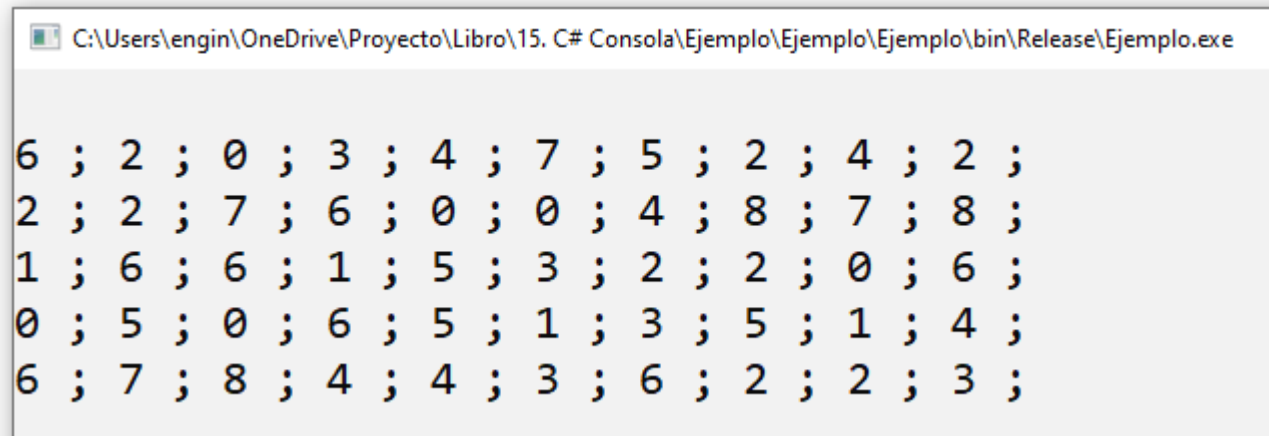


Ilustración 36: Arreglo bidimensional

Arreglo tridimensional

Definición

032.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            int DimensionX = 5;
            int DimensionY = 8;
            int DimensionZ = 3;

            //Declara un arreglo Tridimensional
            int[, ,] Tablero = new int[DimensionX, DimensionY, DimensionZ];

            /* Llena ese arreglo tridimensional
            Tablero.GetLength(0) Retorna la primera dimensión
            Tablero.GetLength(1) Retorna la segunda dimensión
            Tablero.GetLength(2) Retorna la tercera dimensión

            Un arreglo tridimensional inicia en [0,0,0]
            */
            Random azar = new Random();
            for (int posX = 0; posX < Tablero.GetLength(0); posX++)
                for (int posY = 0; posY < Tablero.GetLength(1); posY++)
                    for (int posZ = 0; posZ < Tablero.GetLength(2); posZ++)
                        Tablero[posX, posY, posZ] = azar.Next(0, 9);

            //Imprime ese arreglo tridimensional
            for (int posX = 0; posX < Tablero.GetLength(0); posX++) {
                Console.WriteLine(" ");
                for (int posY = 0; posY < Tablero.GetLength(1); posY++) {
                    Console.Write("[");
                    for (int posZ = 0; posZ < Tablero.GetLength(2); posZ++)
                        Console.Write(Tablero[posX, posY, posZ].ToString() + ";");
                    Console.Write("]  ");
                }

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

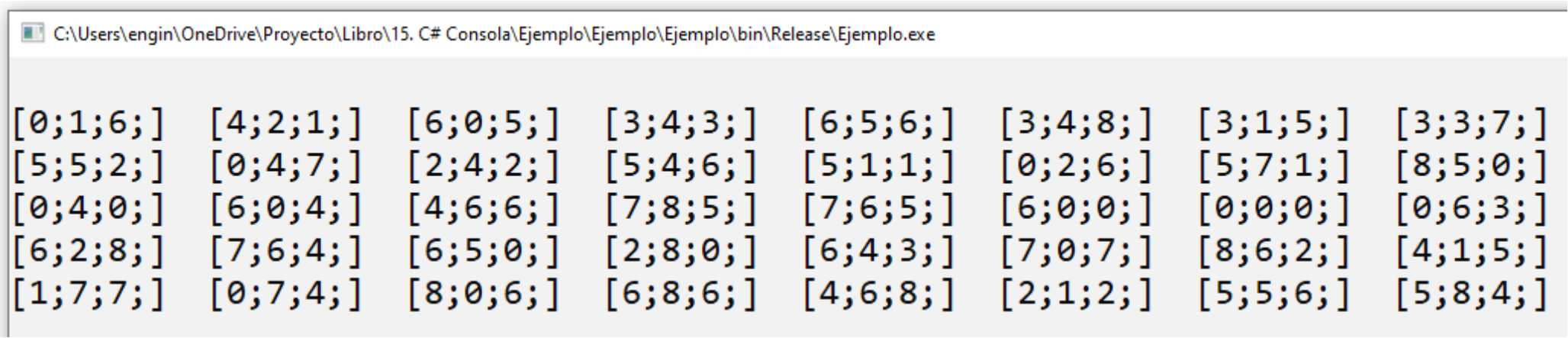


Ilustración 37: Arreglo tridimensional

Arreglo de arreglos

Definición

033.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            /* Arreglo de arreglos (NO confundirlos con arreglos bidimensionales).
             * Se entiende mejor haciendo analogía con un conjunto y subconjuntos */

            //Defino un arreglo (el conjunto)
            int[][] arreglo = new int[5][];

            //Defino los subconjuntos
            arreglo[0] = new int[7]; //Tendrá 7 elementos
            arreglo[1] = new int[3]; //Tendrá 3 elementos
            arreglo[2] = new int[9]; //Tendrá 9 elementos
            arreglo[3] = new int[4]; //Tendrá 4 elementos
            arreglo[4] = new int[8]; //Tendrá 8 elementos

            //Llenando un arreglo de arreglos
            Random azar = new Random();
            for (int conjunto = 0; conjunto < arreglo.Length; conjunto++)
                for (int subconjunto = 0; subconjunto < arreglo[conjunto].Length; subconjunto++)
                    arreglo[conjunto][subconjunto] = azar.Next(0, 9);

            //Imprime ese arreglo de arreglos
            for (int conjunto = 0; conjunto < arreglo.Length; conjunto++) {
                Console.WriteLine(" ");
                for (int subconjunto = 0; subconjunto < arreglo[conjunto].Length; subconjunto++)
                    Console.Write(arreglo[conjunto][subconjunto].ToString() + " ; ");
            }

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



Ilustración 38: Arreglo de arreglos

```
using System;
using System.Diagnostics;

namespace Ejemplo {
    class Program {
        static void Main() {
            /* ¿Qué es más rápido? ¿Un arreglo bidimensional o un arreglo de arreglos */

            //Limite ancho*alto de ambos arreglos
            int Limite = 80;

            //Arreglo Bidimensional
            int[,] bidimensional = new int[Limite, Limite];

            //Arreglo de arreglos
            int[][] arreglo = new int[Limite][];
            for (int subconjunto = 0; subconjunto < arreglo.Length; subconjunto++)
                arreglo[subconjunto] = new int[Limite];

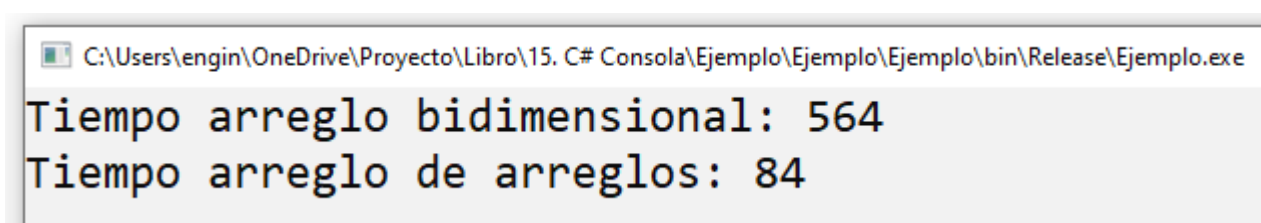
            //Medidor de tiempos
            Stopwatch cronometro = new Stopwatch();

            //Llenando un arreglo bidimensional
            int valor = 0;
            cronometro.Reset();
            cronometro.Start();
            for (int fila = 0; fila < bidimensional.GetLength(0); fila++)
                for (int columna = 0; columna < bidimensional.GetLength(1); columna++)
                    bidimensional[fila, columna] = valor++;
            long TBidim = cronometro.ElapsedTicks;

            //Llenando un arreglo de arreglos
            valor = 0;
            cronometro.Reset();
            cronometro.Start();
            for (int conjunto = 0; conjunto < arreglo.Length; conjunto++)
                for (int subconjunto = 0; subconjunto < arreglo[conjunto].Length; subconjunto++)
                    arreglo[conjunto][subconjunto] = valor++;
            long TArreglo = cronometro.ElapsedTicks;

            //Imprime los tiempos
            Console.WriteLine("Tiempo arreglo bidimensional: " + TBidim.ToString());
            Console.WriteLine("Tiempo arreglo de arreglos: " + TArreglo.ToString());

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



```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\bin\Release\Ejemplo.exe
Tiempo arreglo bidimensional: 564
Tiempo arreglo de arreglos: 84
```

Ilustración 39: Comparando el desempeño entre arreglo bidimensional y arreglo de arreglos

```
using System;
using System.Diagnostics;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Limite de todos los arreglos
            int Limite = 10;

            //Arreglos
            int[,] bidimensional = new int[Limite, Limite];
            int[,,] tridimensional = new int[Limite, Limite, Limite];
            int[,,,] cuatrodimensiones = new int[Limite, Limite, Limite, Limite];
            int[,,,,] cincodimensiones = new int[Limite, Limite, Limite, Limite, Limite];

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

            //Llenando un arreglo bidimensional
            int valor = 0;
            cronometro.Reset();
            cronometro.Start();
            for (int a = 0; a < bidimensional.GetLength(0); a++)
                for (int b = 0; b < bidimensional.GetLength(1); b++)
                    bidimensional[a, b] = valor++;
            long TBidim = cronometro.ElapsedTicks;

            //Llenando un arreglo tridimensional
            valor = 0;
            cronometro.Reset();
            cronometro.Start();
            for (int a = 0; a < tridimensional.GetLength(0); a++)
                for (int b = 0; b < tridimensional.GetLength(1); b++)
                    for (int c = 0; c < tridimensional.GetLength(2); c++)
                        tridimensional[a, b, c] = valor++;
            long TTridim = cronometro.ElapsedTicks;

            //Llenando un arreglo de cuatro dimensiones
            valor = 0;
            cronometro.Reset();
            cronometro.Start();
            for (int a = 0; a < cuatrodimensiones.GetLength(0); a++)
                for (int b = 0; b < cuatrodimensiones.GetLength(1); b++)
                    for (int c = 0; c < cuatrodimensiones.GetLength(2); c++)
                        for (int d = 0; d < cuatrodimensiones.GetLength(3); d++)
                            cuatrodimensiones[a, b, c, d] = valor++;
            long TCuatrodim = cronometro.ElapsedTicks;

            //Llenando un arreglo de cinco dimensiones
            valor = 0;
            cronometro.Reset();
            cronometro.Start();
            for (int a = 0; a < cincodimensiones.GetLength(0); a++)
                for (int b = 0; b < cincodimensiones.GetLength(1); b++)
                    for (int c = 0; c < cincodimensiones.GetLength(2); c++)
                        for (int d = 0; d < cincodimensiones.GetLength(3); d++)
                            for (int e = 0; e < cincodimensiones.GetLength(4); e++)
                                cincodimensiones[a, b, c, d, e] = valor++;
            long TCincodim = cronometro.ElapsedTicks;

            //Imprime los tiempos
            Console.WriteLine("Tiempo arreglo bidimensional: " + TBidim.ToString());
            Console.WriteLine("Tiempo arreglo tridimensional: " + TTridim.ToString());
            Console.WriteLine("Tiempo arreglo cuatro dimensiones: " + TCuatrodim.ToString());
            Console.WriteLine("Tiempo arreglo cinco dimensiones: " + TCincodim.ToString());

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



```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\bin\Release\Ejemplo.exe

Tiempo arreglo bidimensional: 52
Tiempo arreglo tridimensional: 112
Tiempo arreglo cuatro dimensiones: 2006
Tiempo arreglo cinco dimensiones: 15412
```

Ilustración 40: Tiempo tomado en llenar cada arreglo

¡OJO! La memoria puede agotarse con arreglos multidimensionales grandes.

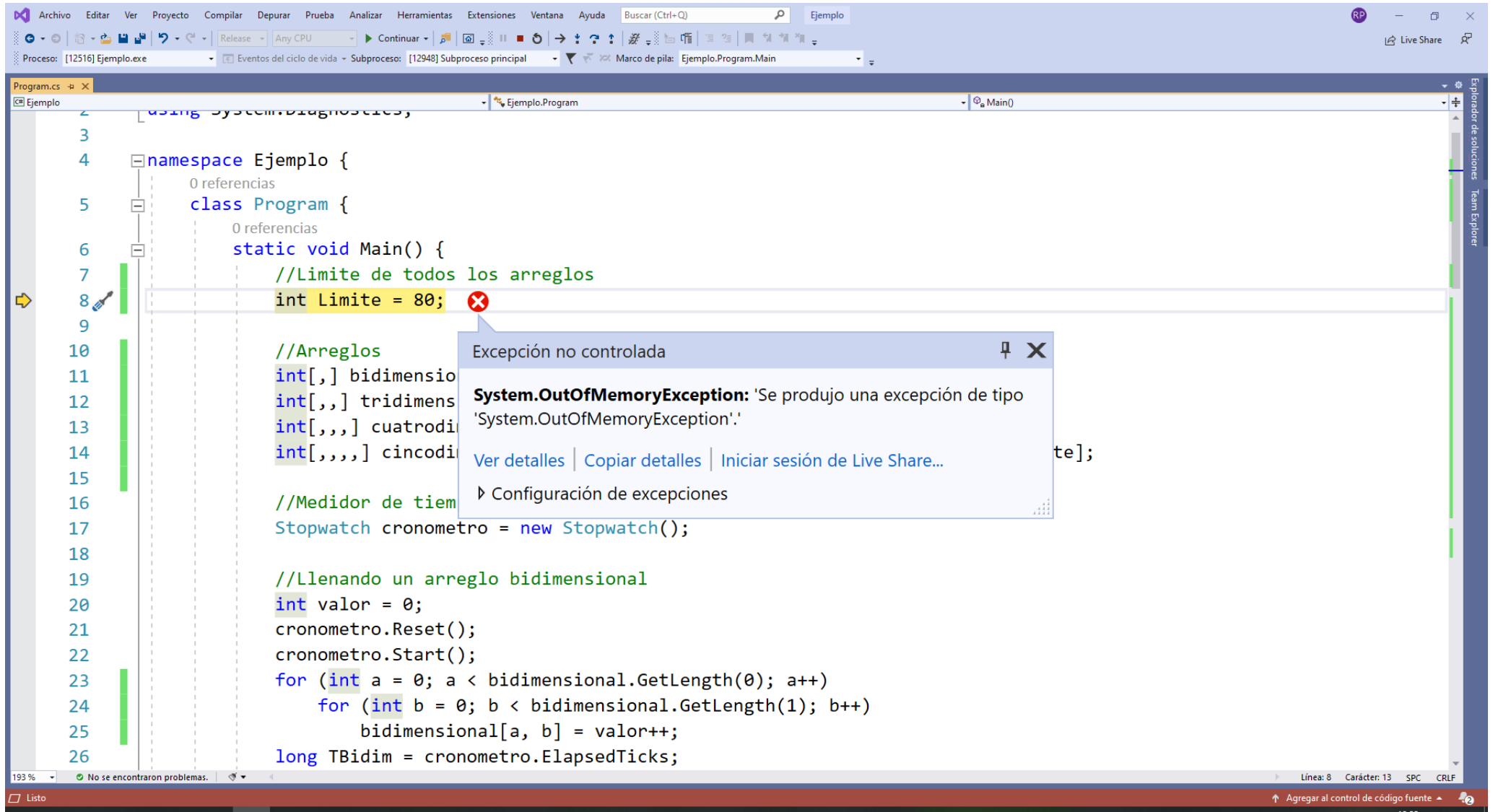


Ilustración 41: Se supera el límite de memoria

Tipo implícito de arreglo

Al inicializar un arreglo que se ha declarado de tipo “var”, este es capaz de determinar el tipo de dato al inicializarlo.

036.cs

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Tipo implícito de arreglo
            var arregloA = new[] { 16, 83, 29, 29 };
            var arregloB = new[] { 'R', 'a', 'f', 'a', 'e', 'l' };
            var arregloC = new[] { "Esta", "es", "una", "prueba" };
            var arregloD = new[] { true, false, false, true };
            var arregloE = new[] { 3.1, 8.9, 2.3 };

            //Imprime los tipos
            Console.WriteLine("Tipo ArregloA: " + arregloA.GetType().ToString());
            Console.WriteLine("Tipo ArregloB: " + arregloB.GetType().ToString());
            Console.WriteLine("Tipo ArregloC: " + arregloC.GetType().ToString());
            Console.WriteLine("Tipo ArregloD: " + arregloD.GetType().ToString());
            Console.WriteLine("Tipo ArregloE: " + arregloE.GetType().ToString());

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

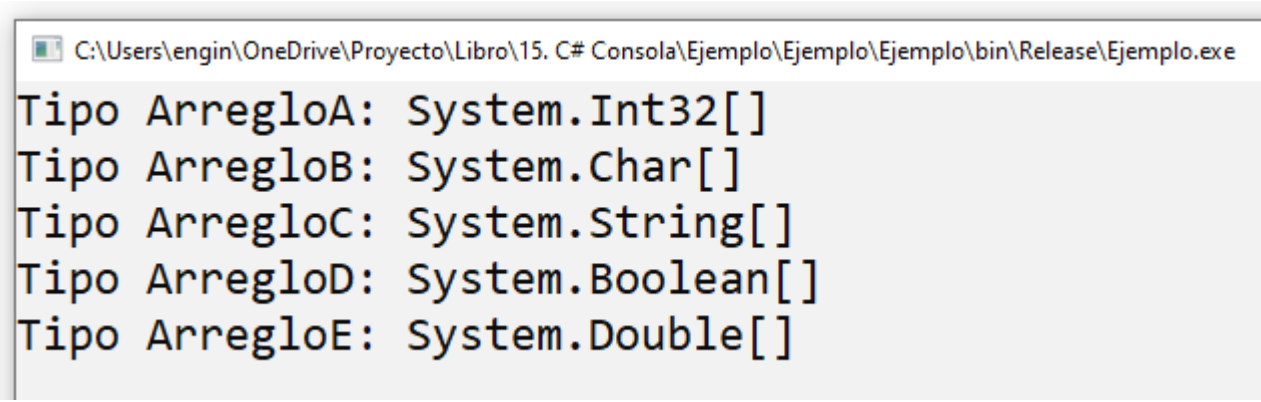


Ilustración 42: Tipo implícito de arreglo

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Tipo implícito en arreglo de arreglos
            var arregloA = new[] {
                new[] { 16, 83, 29, 29 },
                new[] { 72, 6, 26 }
            };

            var arregloB = new[] {
                new[] { 'a', 'e', 'i', 'o' },
                new[] { 'q', 'w', 'e' },
                new[] { 'r', 't', 'y', 'u', 'o', 'p' }
            };

            //Imprime los tipos
            Console.WriteLine("Tipo ArregloA: " + arregloA.GetType().ToString());
            Console.WriteLine("Tipo ArregloB: " + arregloB.GetType().ToString());

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

C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejemplo\Ejemplo\bin\Release\Ejemplo.exe

Tipo ArregloA: System.Int32[][]
 Tipo ArregloB: System.Char[][]

Ilustración 43: Tipo implícito en arreglo de arreglos

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Una cadena
            string Cadena = "Esta es una frase para probar la división por palabras";

            //Se divide en un arreglo de palabras
            string[] Palabras = Cadena.Split(' ');

            foreach (string elemento in Palabras) {
                Console.WriteLine "[" + elemento + ""];
            }

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

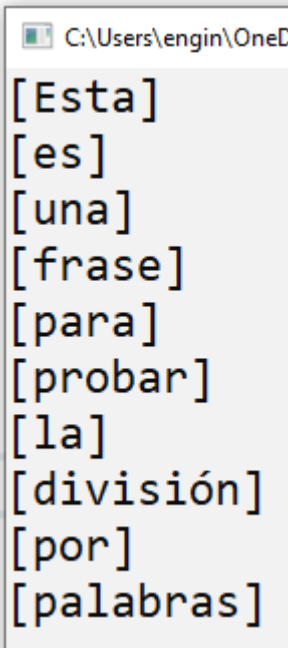


Ilustración 44: Divide la cadena en un arreglo de cadenas al poner el separador de espacio

Sin embargo, si hay más de un espacio intermedio.

```
using System;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Una cadena
            string Cadena = "  abc  def  ghij  klmn  opqrstu  vw  x  yz  ";

            //Se divide en un arreglo de palabras
            string[] Palabras = Cadena.Split(' ');

            //¡OJO! que cuando hay más de un espacio intermedio, este se interpreta como una palabra
            foreach (string elemento in Palabras) {
                Console.WriteLine "[" + elemento + "]");
            }

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

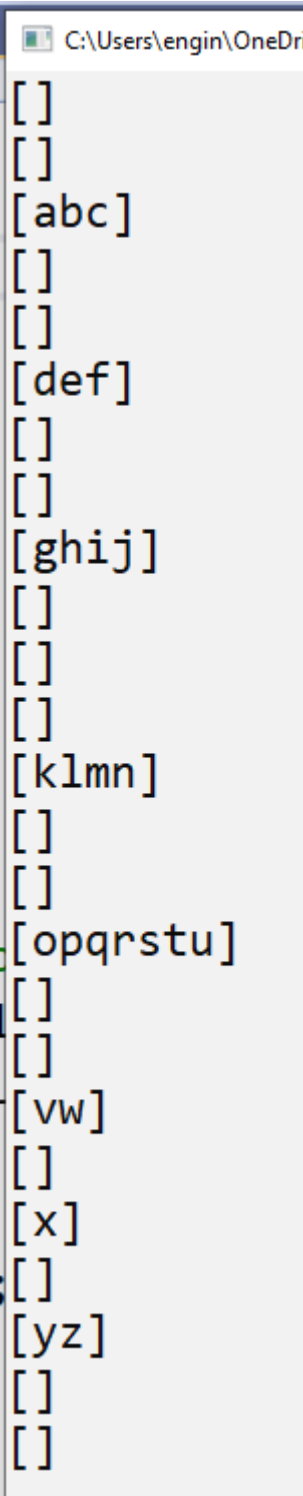


Ilustración 45: El problema es si se ponen muchos espacios entre palabras

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Una cadena
            string Cadena = "esta-es-una-prueba-de-ordenamiento-del-interior-de-una-cadena";
            Console.WriteLine("Cadena: " + Cadena);

            /* Se ordena usando Linq
             * Si desea ordenar los elementos dentro de una secuencia, deberá pasar
             * un método keySelector de identidad que indique que cada elemento de
             * la secuencia es, en sí mismo, una clave. */
            IEnumerable<char> resultado = Cadena.OrderBy(str => str);
            Console.WriteLine("Arreglo con las letras ordenadas");
            foreach (int valor in resultado) {
                Console.Write "[" + (char)valor + " ] ");
            }

            /* Y convierte ese arreglo en cadena */
            string Ordenado = String.Concat(resultado);

            //Imprime
            Console.WriteLine("\r\nOrdenado por letra: " + Ordenado);

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

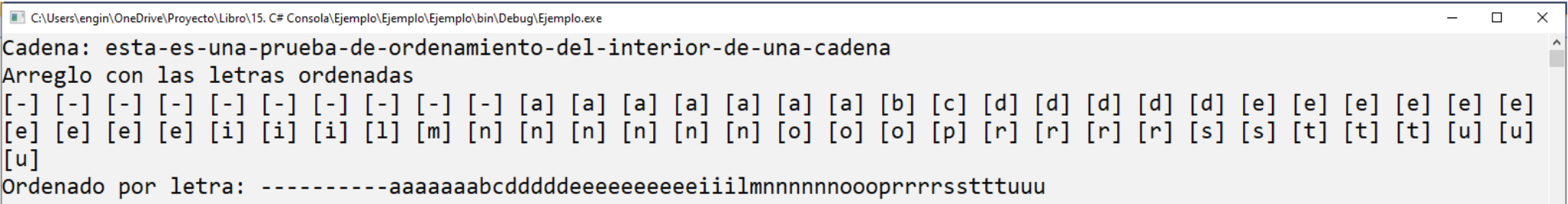


Ilustración 46: Ordenamiento de letras de una cadena.

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Una cadena
            string Cadena = "El-ñandú-es-un-ave-de-Sudamérica.-La-cigüeña-blanca-es-una-especie-de-ave-Ciconiiforme-de-gran-tamaño.";
            Console.WriteLine("Cadena: " + Cadena);

            /* Se ordena usando Linq
            * Si desea ordenar los elementos dentro de una secuencia, deberá pasar
            * un método keySelector de identidad que indique que cada elemento de
            * la secuencia es, en sí mismo, una clave. */
            IEnumerable<char> resultado = Cadena.OrderBy(str => str);
            Console.WriteLine("Arreglo con las letras ordenadas");
            foreach (int valor in resultado) {
                Console.Write "[" + (char)valor + " ] ");
            }

            /* Y convierte ese arreglo en cadena */
            string Ordenado = String.Concat(resultado);

            //Imprime
            Console.WriteLine("\r\nOrdenado por letra: " + Ordenado);

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

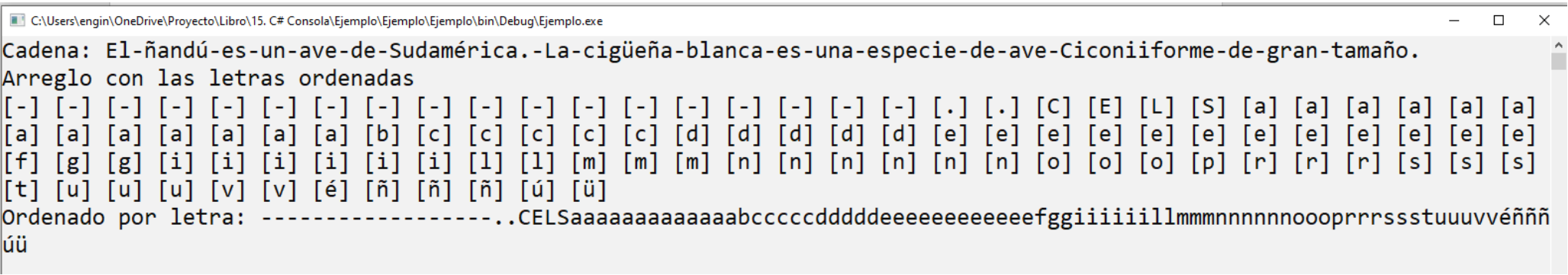


Ilustración 47: El orden alfabético que tiene en cuenta mayúsculas, minúsculas, tildes y diéresis no es lo esperado

Ordenando un arreglo de cadenas con Linq

Orden por tamaño

042.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Una cadena
            string[] Cadenas = { "gato", "perro", "avestruz", "toro", "ballena", "kakapo" };

            /* Se ordena usando Linq
             * Ordena por tamaño de las cadenas. */
            IEnumerable<string> resultado = Cadenas.OrderBy(str => str.Length);
            Console.WriteLine("Ordenado por tamaño");
            foreach (string valor in resultado) {
                Console.WriteLine "[" + valor + " ] ");
            }

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

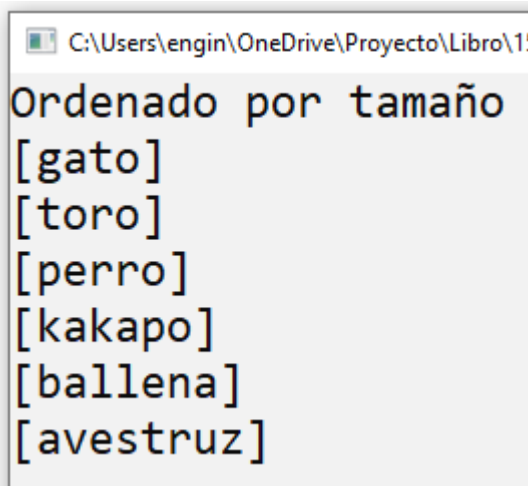


Ilustración 48: Ordenado por tamaño de cada cadena

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Una cadena
            string[] Cadenas = { "gato", "perro", "avestruz", "toro", "ballena", "kakapo" };

            /* Se ordena usando Linq
             * Ordena por la segunda letra. */
            IEnumerable<string> resultado = Cadenas.OrderBy(str => str[1]);
            Console.WriteLine("Ordenado por la segunda letra");
            foreach (string valor in resultado) {
                Console.WriteLine($"[" + valor + "] ");
            }

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

```
C:\Users\engin\OneDrive\Proyecto\Libro\15. C# Consola\Ejemplo\Ejen
Ordenado por la segunda letra
[gato]
[ballena]
[kakapo]
[perro]
[toro]
[avestruz]
```

Ilustración 49: Ordenado por la segunda letra de cada cadena


```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Ejemplo {
    class Program {
        static void Main() {
            //Una cadena
            string[] Cadenas = { "gato", "perro", "avestruz", "toro", "ballena", "kakapo" };

            /* Se ordena usando Linq
             * Invierte el orden */
            IEnumerable<string> resultado = Cadenas.Reverse();
            Console.WriteLine("Invierte el orden");
            foreach (string valor in resultado) {
                Console.WriteLine "[" + valor + " ] ");
            }

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

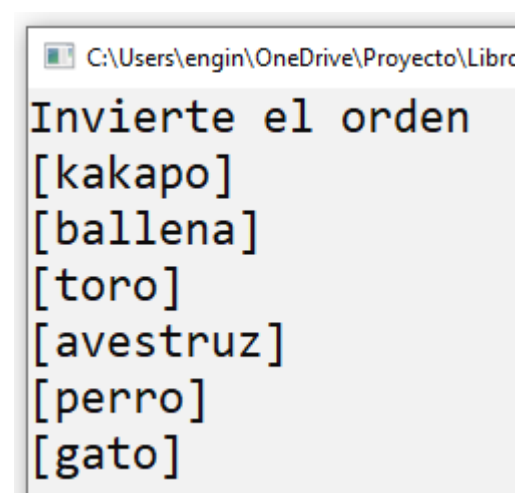


Ilustración 50: Invierte el orden del arreglo