

C# Y .NET 10

Parte 03. Arreglos estáticos

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
Declaración y asignar valores	7
Otra forma de definir y asignar.....	9
Arreglo unidimensional de cadenas	10
Arreglo unidimensional. Tamaños.....	11
Recorrer arreglos de cadenas	12
Uso de la instrucción foreach para recorrer un arreglo	13
Ordenar un arreglo de cadenas.....	14
Ordenar un arreglo de cadenas. Las mayúsculas, minúsculas, tildes y diéresis	15
Ordenar un arreglo de tipo double	16
Funciones genéricas para arreglos unidimensionales	17
Algoritmos de ordenación	19
Métrica de velocidad: Algoritmos de ordenación	24
Arreglo bidimensional	29
Arreglo tridimensional.....	31
Arreglo de arreglos.....	32
Métrica de velocidad: arreglo bidimensional vs arreglo de arreglos	34
Métrica de velocidad: Arreglos multidimensionales.....	36
Tipo implícito de arreglo	39
Tipo implícito en arreglo de arreglos.....	40
Convertir una cadena en un arreglo de cadenas al dividirla	41

Tabla de ilustraciones

Ilustración 1: Declaración y asignar valores	8
Ilustración 2: Otra forma de definir y asignar.....	9
Ilustración 3: Arreglo unidimensional de cadenas	10
Ilustración 4: Arreglo unidimensional. Tamaños.....	11
Ilustración 5: Recorrer arreglos de cadenas	12
Ilustración 6: Uso de la instrucción foreach para recorrer un arreglo.....	13
Ilustración 7: Ordenar un arreglo de cadenas	14
Ilustración 8: Ordenar un arreglo de cadenas. Las mayúsculas, minúsculas, tildes y diéresis ...	15
Ilustración 9: Ordenar un arreglo de tipo double	16
Ilustración 10: Funciones genéricas para arreglos unidimensionales	18
Ilustración 11: Algoritmos de ordenación	23
Ilustración 12: Métrica de velocidad: Algoritmos de ordenación	28
Ilustración 13: Arreglo bidimensional.....	30
Ilustración 14: Arreglo tridimensional	31
Ilustración 15: Arreglo de arreglos	33
Ilustración 16: Métrica de velocidad: arreglo bidimensional vs arreglo de arreglos	35
Ilustración 17: Métrica de velocidad: arreglos multidimensionales	38
Ilustración 18: Tipo implícito de arreglo	39
Ilustración 19: Tipo implícito en arreglo de arreglos	40
Ilustración 20: Convertir una cadena en un arreglo de cadenas al dividirla.....	41
Ilustración 21: Cuando hay más de un espacio intermedio	43

Acerca del autor

Rafael Alberto Moreno Parra

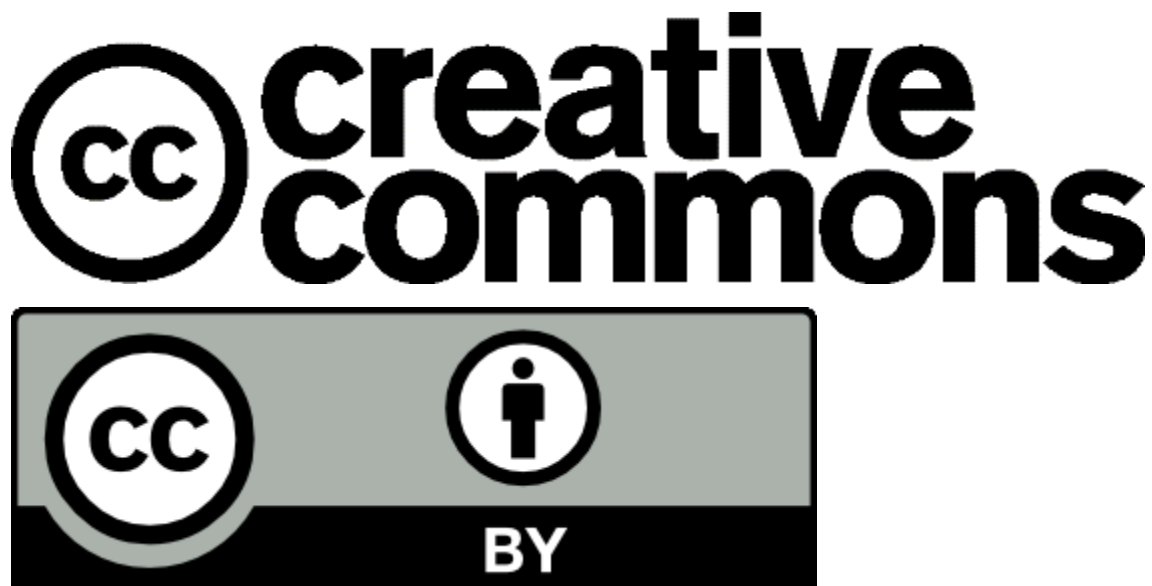
ramsoftware@gmail.com o enginelifelife@hotmail.com

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

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

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

Licencia de este libro



Licencia del software

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



Marcas registradas

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

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

Microsoft ® Visual Studio 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 y mis recordados Sally, Capuchina, Tinita, Tammy, Vikingo y Michu.

Declaración y asignar valores

Los arreglos unidimensionales empiezan en la posición 0

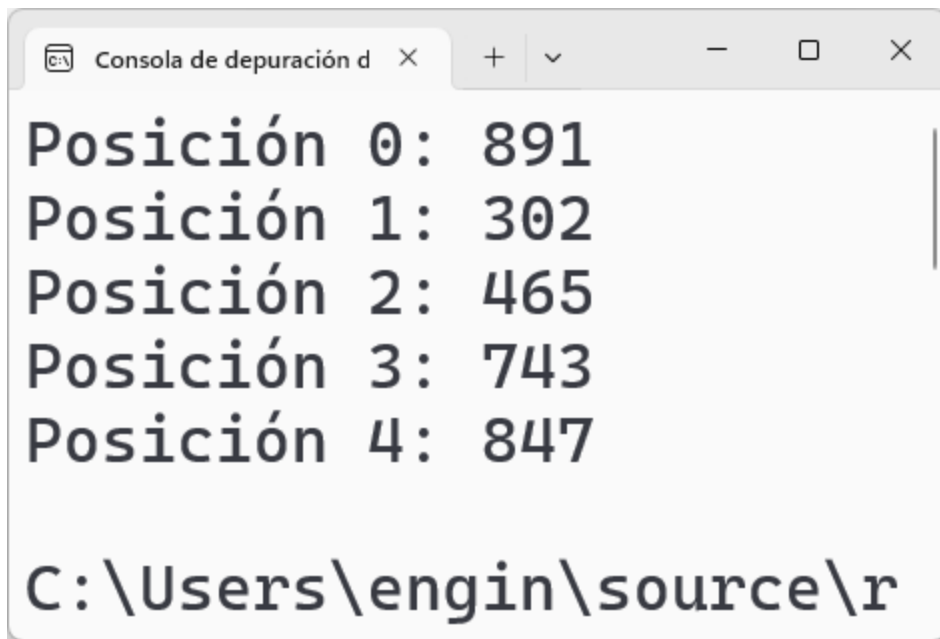
C/001.cs

```
namespace Ejemplo;

internal 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("Posición 0: " + arreglo[0]);
        Console.WriteLine("Posición 1: " + arreglo[1]);
        Console.WriteLine("Posición 2: " + arreglo[2]);
        Console.WriteLine("Posición 3: " + arreglo[3]);
        Console.WriteLine("Posición 4: " + arreglo[4]);
    }
}
```



The image shows a screenshot of a Windows Debug Console window. The title bar at the top reads "Consola de depuración d" followed by a close button. Below the title bar, there are five lines of text: "Posición 0: 891", "Posición 1: 302", "Posición 2: 465", "Posición 3: 743", and "Posición 4: 847". At the bottom of the window, there is a file path: "C:\Users\engin\source\r".

```
Posición 0: 891
Posición 1: 302
Posición 2: 465
Posición 3: 743
Posición 4: 847

C:\Users\engin\source\r
```

Ilustración 1: Declaración y asignar valores

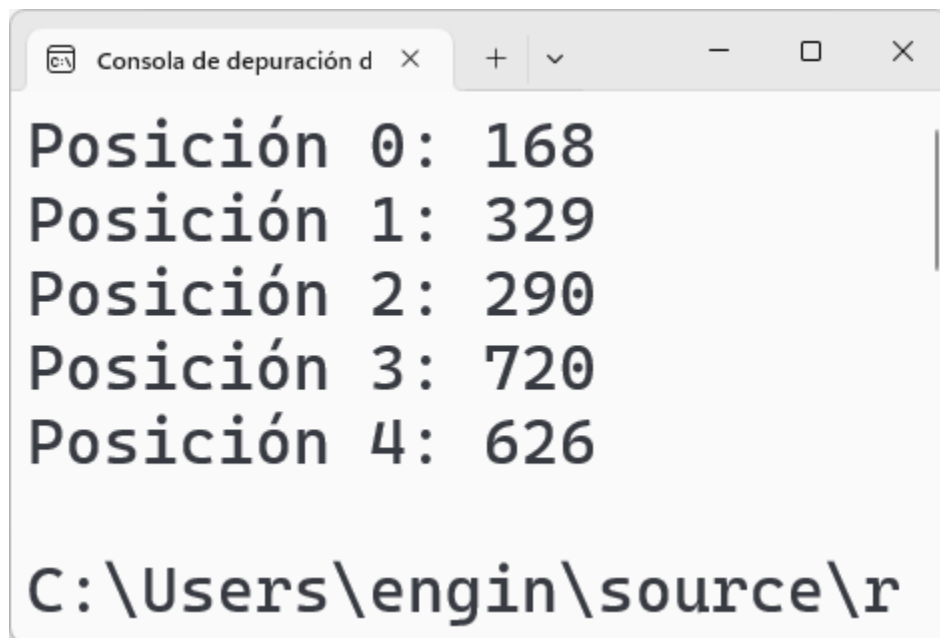
Otra forma de definir y asignar

C/002.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        //Arreglo unidimensional. Definición y asignación.
        int[] arreglo = { 168, 329, 290, 720, 626 };

        //Imprimiendo valores
        Console.WriteLine("Posición 0: " + arreglo[0]);
        Console.WriteLine("Posición 1: " + arreglo[1]);
        Console.WriteLine("Posición 2: " + arreglo[2]);
        Console.WriteLine("Posición 3: " + arreglo[3]);
        Console.WriteLine("Posición 4: " + arreglo[4]);
    }
}
```



```
Consola de depuración d
Posición 0: 168
Posición 1: 329
Posición 2: 290
Posición 3: 720
Posición 4: 626

C:\Users\engin\source\r
```

Ilustración 2: Otra forma de definir y asignar

Arreglo unidimensional de cadenas

C/003.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        //Arreglo unidimensional de cadenas.
        string[] cadenas = ["Sally", "Suini", "Grisú", "Arian"];

        //Imprimiendo valores
        Console.WriteLine("Posición 0: " + cadenas[0]);
        Console.WriteLine("Posición 1: " + cadenas[1]);
        Console.WriteLine("Posición 2: " + cadenas[2]);
        Console.WriteLine("Posición 3: " + cadenas[3]);
    }
}
```



Ilustración 3: Arreglo unidimensional de cadenas

Arreglo unidimensional. Tamaños

C/004.cs

```
namespace Ejemplo;

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

        //Tamaños de los arreglos
        int TCadenas = cadenas.Length;
        int TNumeros = numeros.Length;

        //Imprime
        Console.WriteLine("Tamaño arreglo cadenas: " + TCadenas);
        Console.WriteLine("Tamaño arreglo numeros: " + TNumeros);
    }
}
```

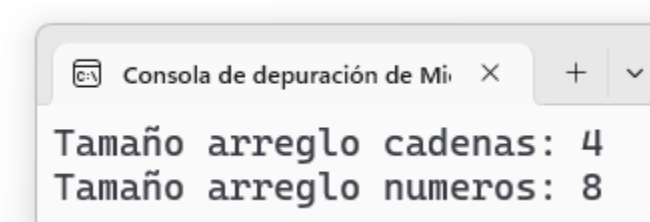


Ilustración 4: Arreglo unidimensional. Tamaños

Recorrer arreglos de cadenas

C/005.cs

```
namespace Ejemplo;

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

        //Tamaño
        int TCadenas = cadenas.Length;

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

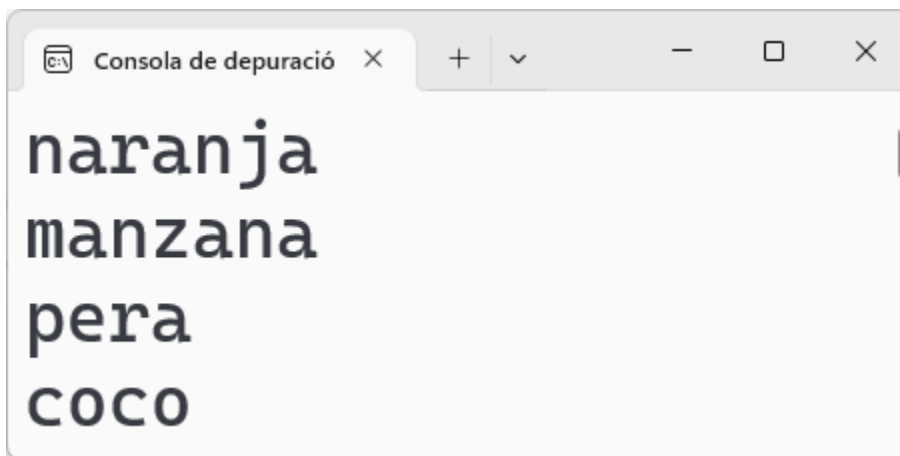


Ilustración 5: Recorrer arreglos de cadenas

Uso de la instrucción foreach para recorrer un arreglo

C/006.cs

```
namespace Ejemplo;

internal 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);
        }
    }
}
```

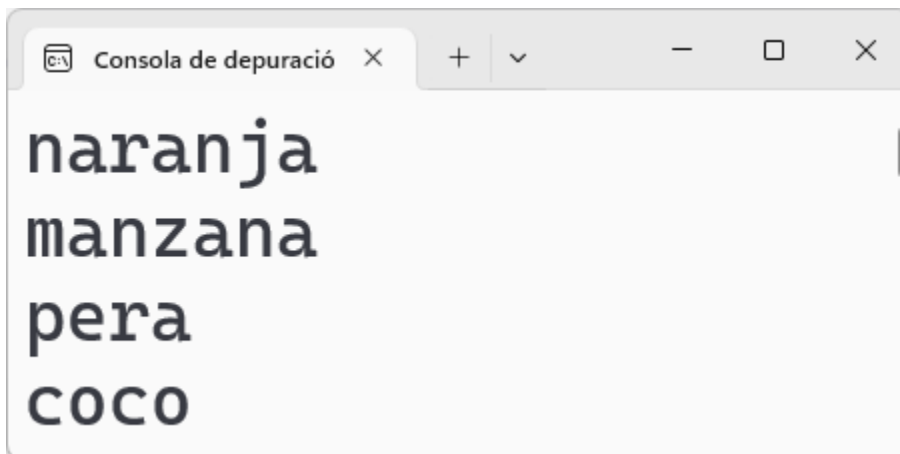


Ilustración 6: Uso de la instrucción foreach para recorrer un arreglo

Ordenar un arreglo de cadenas

C/007.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        //Arreglo unidimensional
        string[] cadenas = ["mil", "ufr", "ac", "ari", "an"];

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

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

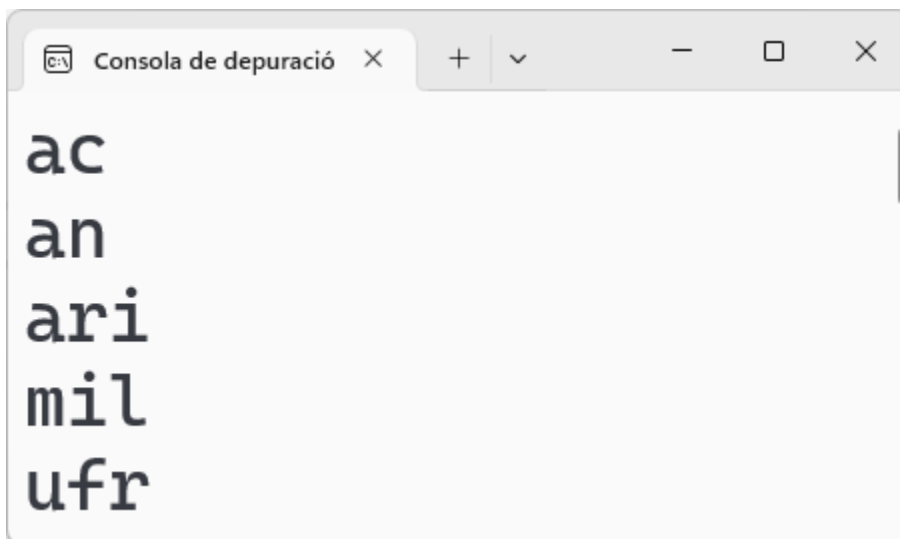


Ilustración 7: Ordenar un arreglo de cadenas

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

C/008.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        //Arreglo unidimensional
        string[] cadenas = ["áa", "aa", "äa", "Aa", "aá"];

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

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

En caso de probar las tildes, diéresis, mayúsculas y minúsculas, se obtendría este resultado:

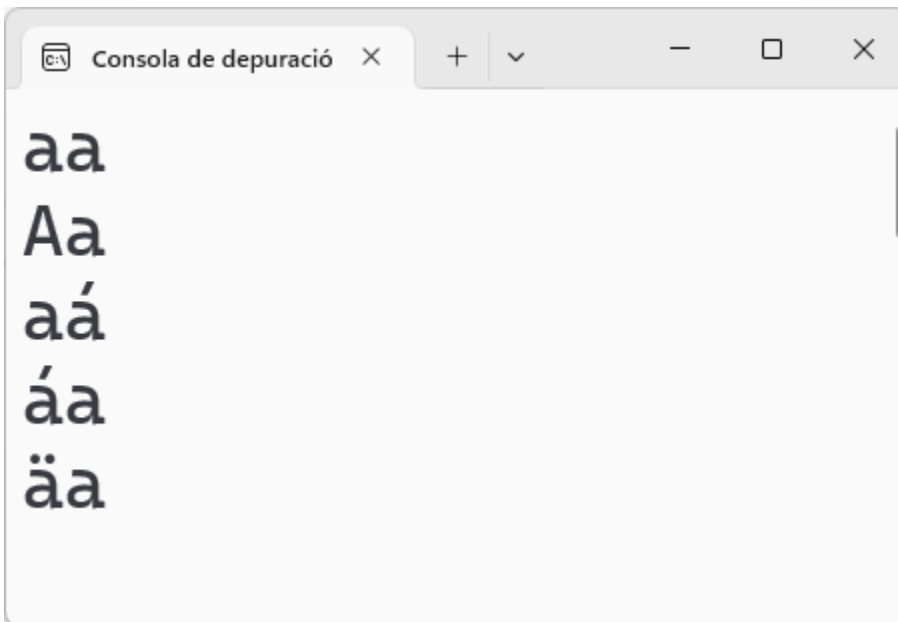


Ilustración 8: Ordenar un arreglo de cadenas. Las mayúsculas, minúsculas, tildes y diéresis

Ordenar un arreglo de tipo double

C/009.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        Random Azar = new();

        //Llena un arreglo con valores aleatorios
        double[] Numeros = new double[10];
        for (int cont = 0; cont < Numeros.Length; cont++)
            Numeros[cont] = Azar.NextDouble();

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

        //Recorre el arreglo y lo imprime
        foreach (double unvalor in Numeros) {
            Console.WriteLine(unvalor);
        }
    }
}
```

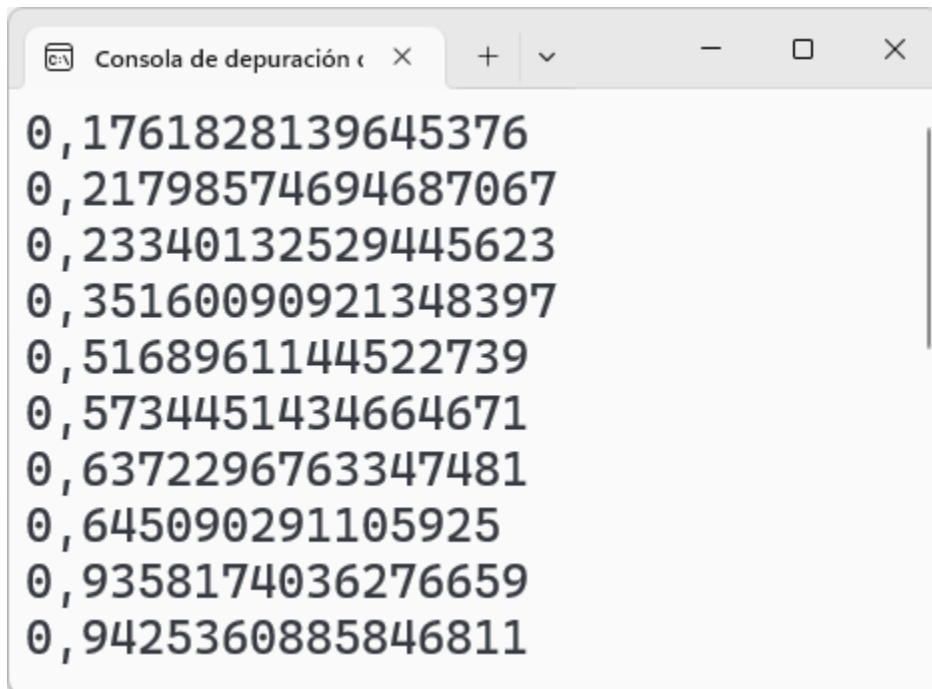


Ilustración 9: Ordenar un arreglo de tipo double

Funciones genéricas para arreglos unidimensionales

Se crean funciones que se envía como parámetro el arreglo

C/010.cs

```
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 pos = PosArregloDato(numeros, valorBusca);
        if (pos == -1)
            Console.WriteLine("Valor no encontrado");
        else {
            Console.Write("Valor " + valorBusca);
            Console.WriteLine(" en posición: " + pos);
        }
    }

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

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

    //Retorna la posición del dato en el arreglo
}
```

```

//o retorna -1 si no lo pos
static int PosArregloDato(int[] arreglo, int valor) {
    for (int pos = 0; pos < arreglo.Length; pos++) {
        if (arreglo[pos] == valor)
            return pos;
    }
    return -1;
}
}

```

The screenshot shows a debug console window titled 'Consola de depuración'. It displays a list of numbers arranged in a grid-like format, with semicolons separating them. The numbers are: 20, 26, 19, 12, 19, 30, 3, 0, 2, 17, 1, 7, 21, 15, 5, 3, 14, 27, 3, 8, 16, 8, 29, 14, 14, 21, 19, 29, 8, 22, 25, 26, 16, 10, 13, 3, 17, 15, 26, 21, 2, 9, 23, 3, 19, 6, 21, 23, 24, 3, 18, 11. Below the list, it says 'Valor 17 en posición: 8'.

Ilustración 10: Funciones genéricas para arreglos unidimensionales

Algoritmos de ordenación

Se prueban diversos algoritmos de ordenación enviando arreglos como parámetros y estos arreglos son modificados por las funciones mismas. 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.

C/011.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        //Ordenación
        int Limite = 10;
        int[] Listado = new int[Limite];

        int minimo = 10, maximo = 99;

        //Ordena por Inserción
        Console.WriteLine("\nOrdena por inserción");
        LlenaArreglo(Listado, minimo, maximo);
        ImprimeArreglo("Original", Listado);
        Insercion(Listado);
        ImprimeArreglo("Ordenado", Listado);

        //Ordena por Selección
        Console.WriteLine("\nOrdena por selección");
        LlenaArreglo(Listado, minimo, maximo);
        Array.Copy(Listado, 0, Listado, 0, Listado.Length);
        ImprimeArreglo("Original", Listado);
        Seleccion(Listado);
        ImprimeArreglo("Ordenado", Listado);

        //Ordena por Burbuja
        Console.WriteLine("\nOrdena por burbuja");
        LlenaArreglo(Listado, minimo, maximo);
        ImprimeArreglo("Original", Listado);
        Burbuja(Listado);
        ImprimeArreglo("Ordenado", Listado);

        //Ordena por Shell
        Console.WriteLine("\nOrdena por shell");
        LlenaArreglo(Listado, minimo, maximo);
        ImprimeArreglo("Original", Listado);
```

```

Shell(Listado);
ImprimeArreglo("Ordenado", Listado);

//Ordena por QuickSort
Console.WriteLine("\nOrdena por quicksort");
LlenaArreglo(Listado, minimo, maximo);
ImprimeArreglo("Original", Listado);
QuickSort(Listado, 0, Listado.Length - 1);
ImprimeArreglo("Ordenado", Listado);
}

//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 pos = 0; pos < arreglo.Length; pos++) {
        arreglo[pos] = azar.Next(min, max + 1);
    }
}

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

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

```

```

        }
    }
    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[] arr) {
    int incr = arr.Length;
    do {
        incr /= 2;
        for (int k = 0; k < incr; k++) {
            for (int i = incr + k; i < arr.Length; i += incr) {
                int j = i;
                while (j - incr >= 0 && arr[j] < arr[j - incr]) {
                    int tmp = arr[j];
                    arr[j] = arr[j - incr];
                    arr[j - incr] = tmp;
                    j -= incr;
                }
            }
        }
    } while (incr > 1);
}

//Ordenación por QuickSort
static void QuickSort(int[] arreglo, int primero, int ultimo) {
    int i, j, central;
    int pivote;

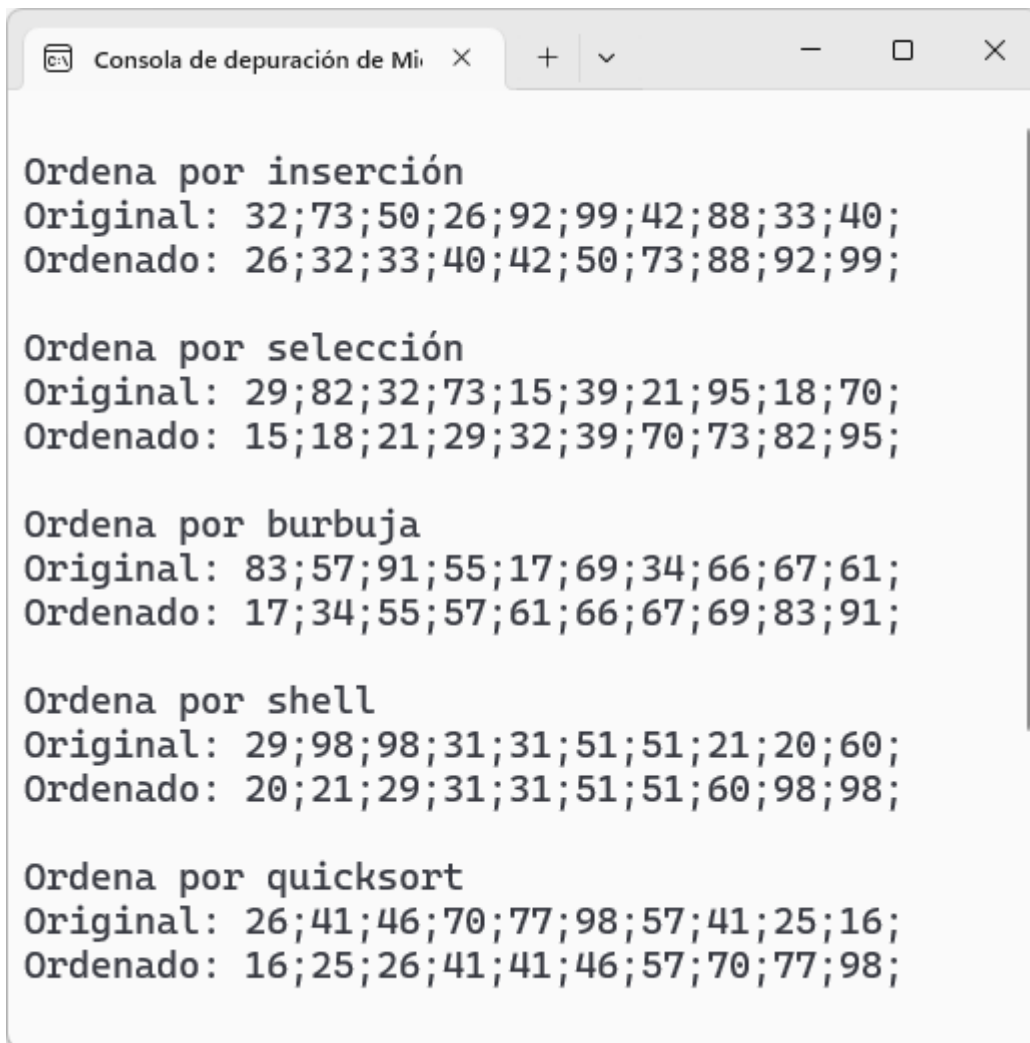
```

```

    central = (primero + ultimo) / 2;
    pivote = arreglo[central];
    i = primero;
    j = ultimo;
    do {
        while (arreglo[i] < pivote) i++;
        while (arreglo[j] > pivote) j--;
        if (i <= j) {
            int tmp = arreglo[i];
            arreglo[i] = arreglo[j];
            arreglo[j] = tmp;
            i++;
            j--;
        }
    } while (i <= j);

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

```



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

Ordena por inserción
Original: 32;73;50;26;92;99;42;88;33;40;
Ordenado: 26;32;33;40;42;50;73;88;92;99;

Ordena por selección
Original: 29;82;32;73;15;39;21;95;18;70;
Ordenado: 15;18;21;29;32;39;70;73;82;95;

Ordena por burbuja
Original: 83;57;91;55;17;69;34;66;67;61;
Ordenado: 17;34;55;57;61;66;67;69;83;91;

Ordena por shell
Original: 29;98;98;31;31;51;51;21;20;60;
Ordenado: 20;21;29;31;31;51;51;60;98;98;

Ordena por quicksort
Original: 26;41;46;70;77;98;57;41;25;16;
Ordenado: 16;25;26;41;41;46;57;70;77;98;
```

Ilustración 11: Algoritmos de ordenación

```
using System.Diagnostics;

namespace Ejemplo;

internal class Program {
    static void Main() {
        int Limite = 30000;
        int[] Original = new int[Limite];
        int[] LShell = new int[Limite];
        int[] LInsercion = new int[Limite];
        int[] LSeleccion = new int[Limite];
        int[] LBurbuja = new int[Limite];
        int[] LQuickSort = new int[Limite];

        //Medidor de tiempos
        double 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 = 20;
        for (int prueba = 1; prueba <= TotalPruebas; prueba++) {
            LlenaArreglo(Original, 10, 90);

            //Ordenación por método Shell
            Array.Copy(Original, 0, LShell, 0, Original.Length);
            long Inicia = Stopwatch.GetTimestamp();
            Shell(LShell);
            TimeSpan Transcurrido = Stopwatch.GetElapsedTime(Inicia);
            TPShell += Transcurrido.TotalMilliseconds;

            //Ordenación por método Inserción
            Array.Copy(Original, 0, LInsercion, 0, Original.Length);
            Inicia = Stopwatch.GetTimestamp();
            Insercion(LInsercion);
            Transcurrido = Stopwatch.GetElapsedTime(Inicia);
            TPIns += Transcurrido.TotalMilliseconds;

            //Ordenación por método Selección
            Array.Copy(Original, 0, LSeleccion, 0, Original.Length);
            Inicia = Stopwatch.GetTimestamp();
            Seleccion(LSeleccion);
            Transcurrido = Stopwatch.GetElapsedTime(Inicia);
            TPSel += Transcurrido.TotalMilliseconds;
        }
    }
}
```



```

//Ordenación por método Burbuja
Array.Copy(Original, 0, LBurbuja, 0, Original.Length);
Inicia = Stopwatch.GetTimestamp();
Burbuja(LBurbuja);
Transcurrido = Stopwatch.GetElapsedTime(Inicia);
TPBur += Transcurrido.TotalMilliseconds;

//Ordenación por método QuickSort
Array.Copy(Original, 0, LQuickSort, 0, Original.Length);
Inicia = Stopwatch.GetTimestamp();
QuickSort(LQuickSort, 0, LQuickSort.Length - 1);
Transcurrido = Stopwatch.GetElapsedTime(Inicia);
TPQuick += Transcurrido.TotalMilliseconds;

//Verifica que los arreglos ordenados coinciden
for (int cont = 0; cont < Original.Length; cont++) {
    if (LShell[cont] != LInsercion[cont] ||
        LInsercion[cont] != LSeleccion[cont] ||
        LSeleccion[cont] != LBurbuja[cont] ||
        LBurbuja[cont] != LQuickSort[cont])
        Console.WriteLine("Error en la prueba");
}
}

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("Número de elementos a ordenar: " + Limite);
Console.WriteLine("Tiempo promedio en milisegundos");
Console.WriteLine("ShellSort: " + TS);
Console.WriteLine("InsertSort: " + TI);
Console.WriteLine("Selección: " + TL);
Console.WriteLine("Burbuja: " + TB);
Console.WriteLine("QuickSort: " + TQ);
}

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

//Ordenamiento por Insert

```

```

static void Insercion(int[] arreglo) {
    int j;
    for (int i = 1; i < arreglo.Length; i++) {
        int 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[] arr) {
    int incr = arr.Length;
    do {
        incr /= 2;
    }
}

```

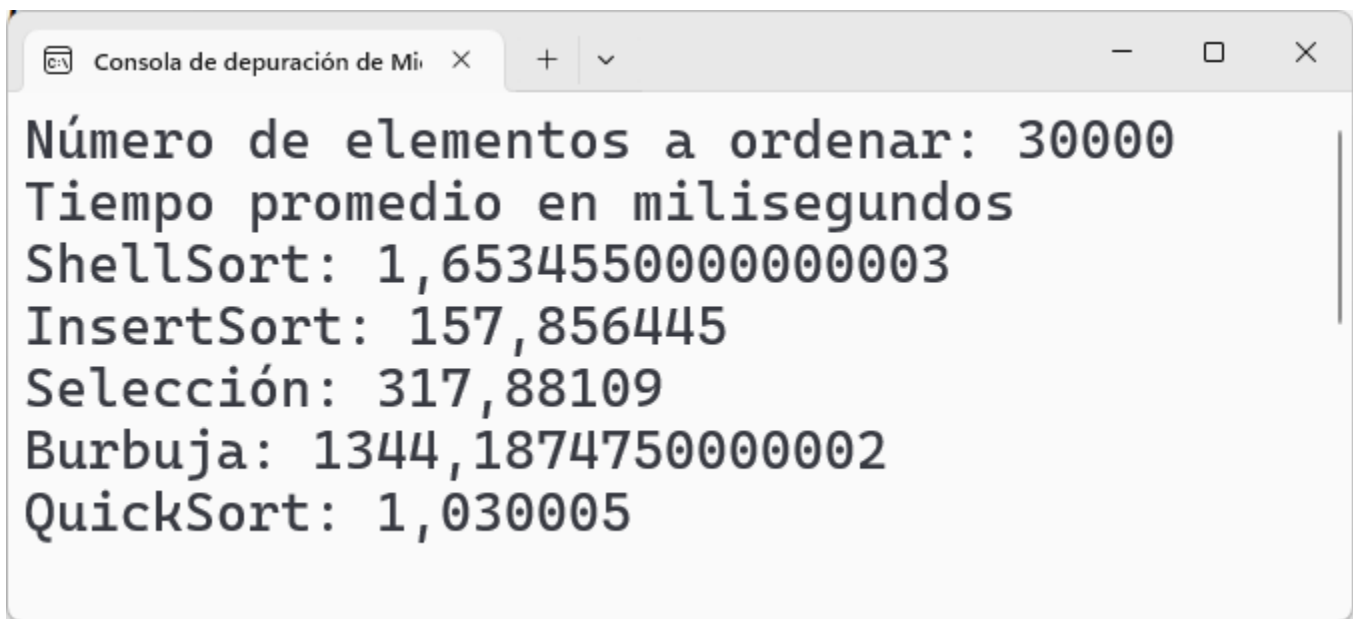
```

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

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

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

```



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

```
Número de elementos a ordenar: 30000  
Tiempo promedio en milisegundos  
ShellSort: 1,6534550000000003  
InsertSort: 157,856445  
Selección: 317,88109  
Burbuja: 1344,1874750000002  
QuickSort: 1,030005
```

Ilustración 12: Métrica de velocidad: Algoritmos de ordenación

Arreglo bidimensional

C/013.cs

```
namespace Ejemplo;

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

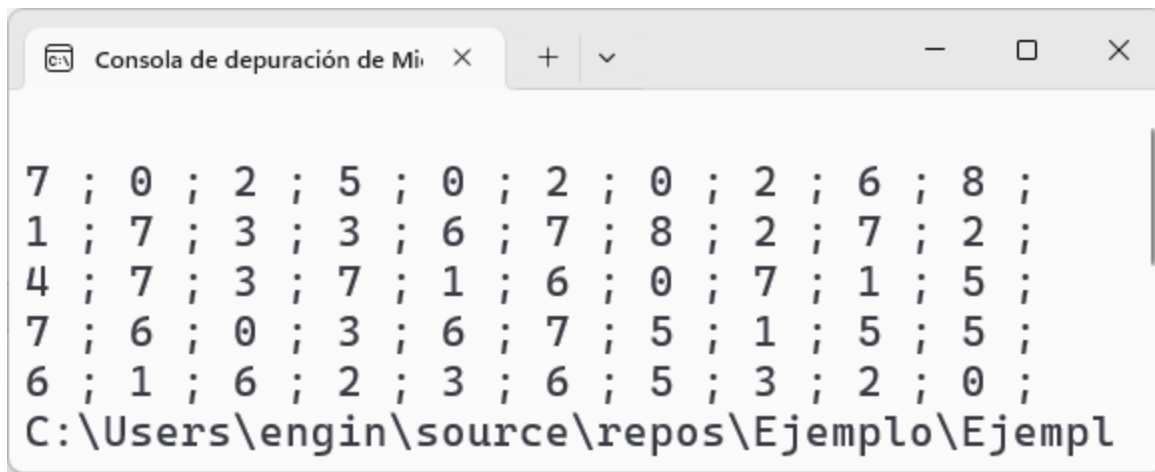
        //Declara un arreglo bidimensional
        int[,] Tablero = new int[TotalFilas, TotalColumnas];

        /* Llena ese arreglo bidimensional
           Retorna el número de filas (la primera dimensión)
           Tablero.GetLength(0)

           Retorna el número de columnas (la segunda dimensión)
           Tablero.GetLength(1)

           Un arreglo bidimensional inicia en [0,0] y
           termina en [TotalFilas-1, TotalColumnas-1]
        */
        Random azar = new();
        for (int fila = 0; fila < Tablero.GetLength(0); fila++)
            for (int col = 0; col < Tablero.GetLength(1); col++)
                Tablero[fila, col] = azar.Next(0, 9);

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



The image shows a screenshot of a Windows Debug Console window. The title bar reads "Consola de depuración de Mi" followed by a close button. The window contains a 5x10 grid of numbers separated by semicolons, and a file path below it.

```
7 ; 0 ; 2 ; 5 ; 0 ; 2 ; 0 ; 2 ; 6 ; 8 ;  
1 ; 7 ; 3 ; 3 ; 6 ; 7 ; 8 ; 2 ; 7 ; 2 ;  
4 ; 7 ; 3 ; 7 ; 1 ; 6 ; 0 ; 7 ; 1 ; 5 ;  
7 ; 6 ; 0 ; 3 ; 6 ; 7 ; 5 ; 1 ; 5 ; 5 ;  
6 ; 1 ; 6 ; 2 ; 3 ; 6 ; 5 ; 3 ; 2 ; 0 ;  
C:\Users\engin\source\repos\Ejemplo\Ejempl
```

Ilustración 13: Arreglo bidimensional

Arreglo tridimensional

C/014.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        int TamanoX = 3;
        int TamanoY = 4;
        int TamanoZ = 5;

        //Declara un arreglo Tridimensional
        int[, ,] Solido = new int[TamanoX, TamanoY, TamanoZ];

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

           Un arreglo tridimensional inicia en [0,0,0]
        */
        for (int posX = 0; posX < Solido.GetLength(0); posX++)
            for (int posY = 0; posY < Solido.GetLength(1); posY++)
                for (int posZ = 0; posZ < Solido.GetLength(2); posZ++)
                    Solido[posX, posY, posZ] = (posX + 1) * 100 + posY * 10 +
posZ;

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

Consola de depuración de Mi

[100;101;102;103;104;]	[110;111;112;113;114;]	[120;121;122;123;124;]	[130;131;132;133;134;]
[200;201;202;203;204;]	[210;211;212;213;214;]	[220;221;222;223;224;]	[230;231;232;233;234;]
[300;301;302;303;304;]	[310;311;312;313;314;]	[320;321;322;323;324;]	[330;331;332;333;334;]

Ilustración 14: Arreglo tridimensional

Arreglo de arreglos

C/015.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        /* Arreglo de arreglos
        * NO confundirlos con arreglos bidimensionales.
        * Se entiende mejor haciendo analogía con
        * un tronco y ramas */

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

        //Defino las ramas
        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();
        for (int tronco = 0; tronco < arreglo.Length; tronco++)
            for (int rama = 0; rama < arreglo[tronco].Length; rama++)
                arreglo[tronco][rama] = azar.Next(0, 9);

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




A screenshot of a web browser's developer console. The console has a tab labeled 'Consola de depuración de Mi' with a close button (X) and expand/collapse icons (+ and v). The console displays a grid of numbers arranged in five rows and ten columns, separated by semicolons. The numbers are: Row 1: 3, 8, 4, 1, 1, 0, 8; Row 2: 5, 6, 2; Row 3: 6, 4, 6, 6, 1, 8, 8, 7, 1; Row 4: 5, 0, 6, 0; Row 5: 2, 8, 2, 0, 5, 2, 5, 2.

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

Ilustración 15: Arreglo de arreglos

Métrica de velocidad: arreglo bidimensional vs arreglo de arreglos

C/016.cs

```
using System.Diagnostics;

namespace Ejemplo;

internal 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[,] tablero = new int[Limite, Limite];

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

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

        //Llenando un arreglo bidimensional
        int valor = 0;
        cronometro.Reset();
        cronometro.Start();
        for (int fila = 0; fila < tablero.GetLength(0); fila++)
            for (int col = 0; col < tablero.GetLength(1); col++)
                tablero[fila, col] = 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 rama = 0; rama < arreglo[conjunto].Length; rama++)
                arreglo[conjunto][rama] = valor++;
        long TArreglo = cronometro.ElapsedTicks;

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

```
}  
}
```

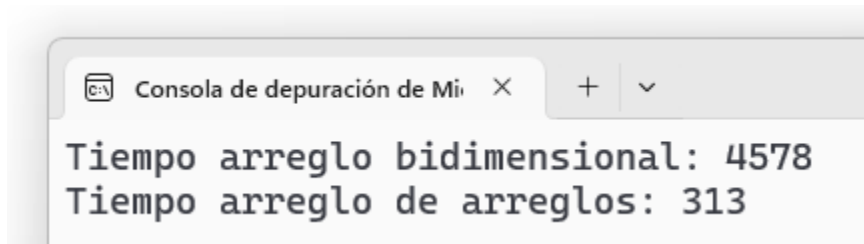


Ilustración 16: Métrica de velocidad: arreglo bidimensional vs arreglo de arreglos

```
using System.Diagnostics;

namespace Ejemplo;

internal class Program {
    static void Main() {
        Random Azar = new();

        //Limite de todos los arreglos
        int Lim = 60;

        //Arreglos
        int[,] dosDim = new int[Lim, Lim];
        int[,,,] tresDim = new int[Lim, Lim, Lim, Lim];
        int[,,,] cuatroDim = new int[Lim, Lim, Lim, Lim];
        int[,,,] cincoDim = new int[Lim, Lim, Lim, Lim, Lim];

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

        //Evitar optimización de ciclos
        //haciendo que acumule los valores
        long Acumula = 0;

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

        int posA = Azar.Next(dosDim.GetLength(0));
        int posB = Azar.Next(dosDim.GetLength(1));
        Acumula += dosDim[posA, posB];

        //Llenando un arreglo tridimensional
        valor = 0;
        cronometro.Reset();
        cronometro.Start();
        for (int a = 0; a < tresDim.GetLength(0); a++)
            for (int b = 0; b < tresDim.GetLength(1); b++)
                for (int c = 0; c < tresDim.GetLength(2); c++)
```

```

        tresDim[a, b, c] = valor++;
long TTridim = cronometro.ElapsedMilliseconds;

posA = Azar.Next(tresDim.GetLength(0));
posB = Azar.Next(tresDim.GetLength(1));
int posC = Azar.Next(tresDim.GetLength(2));
Acumula += tresDim[posA, posB, posC];

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

posA = Azar.Next(cuatroDim.GetLength(0));
posB = Azar.Next(cuatroDim.GetLength(1));
posC = Azar.Next(cuatroDim.GetLength(2));
int posD = Azar.Next(cuatroDim.GetLength(3));
Acumula += cuatroDim[posA, posB, posC, posD];

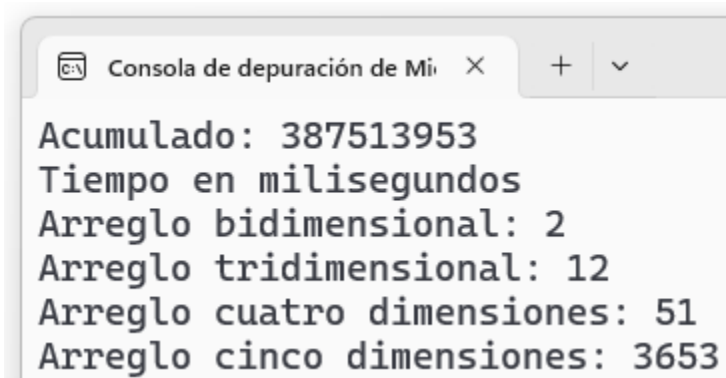
//Llenando un arreglo de cinco dimensiones
valor = 0;
cronometro.Reset();
cronometro.Start();
for (int a = 0; a < cincoDim.GetLength(0); a++)
    for (int b = 0; b < cincoDim.GetLength(1); b++)
        for (int c = 0; c < cincoDim.GetLength(2); c++)
            for (int d = 0; d < cincoDim.GetLength(3);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

posA = Azar.Next(cincoDim.GetLength(0));
posB = Azar.Next(cincoDim.GetLength(1));
posC = Azar.Next(cincoDim.GetLength(2));
posD = Azar.Next(cincoDim.GetLength(3));
int posE = Azar.Next(cincoDim.GetLength(4));
Acumula += cincoDim[posA, posB, posC, posD, posE];

//Imprime los tiempos
Console.WriteLine("Acumulado: " + Acumula);
Console.WriteLine("Tiempo en milisegundos");
Console.WriteLine("Arreglo bidimensional: " + TBidim);

```

```
Console.WriteLine("Arreglo tridimensional: " + TTridim);  
Console.WriteLine("Arreglo cuatro dimensiones: " + TCuatrodin);  
Console.WriteLine("Arreglo cinco dimensiones: " + TCincodin);  
}  
}
```



The screenshot shows a Windows Debug Console window titled 'Consola de depuración de Mi'. It displays the following output:

```
Acumulado: 387513953  
Tiempo en milisegundos  
Arreglo bidimensional: 2  
Arreglo tridimensional: 12  
Arreglo cuatro dimensiones: 51  
Arreglo cinco dimensiones: 3653
```

Ilustración 17: Métrica de velocidad: arreglos multidimensionales

Tipo implícito de arreglo

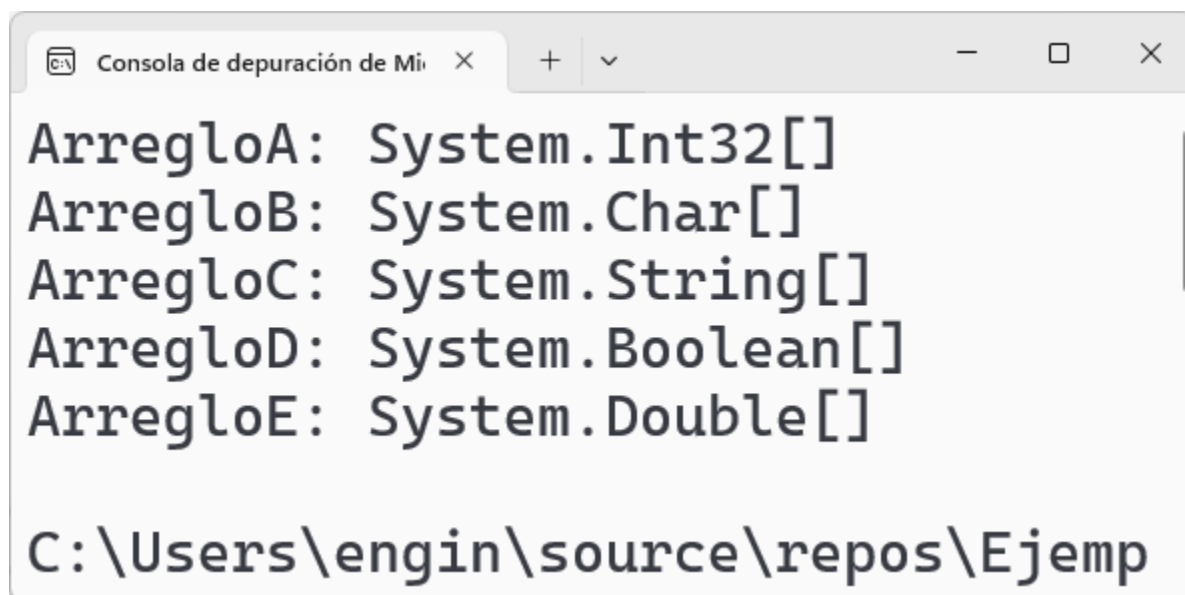
Al inicializar un arreglo que se ha declarado de tipo "var", este es capaz de determinar el tipo de dato al inicializarlo.

C/018.cs

```
namespace Ejemplo;

internal 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("ArregloA: " + arregloA.GetType());
        Console.WriteLine("ArregloB: " + arregloB.GetType());
        Console.WriteLine("ArregloC: " + arregloC.GetType());
        Console.WriteLine("ArregloD: " + arregloD.GetType());
        Console.WriteLine("ArregloE: " + arregloE.GetType());
    }
}
```



```
Consola de depuración de Mi  +  -  □  ×

ArregloA: System.Int32[]
ArregloB: System.Char[]
ArregloC: System.String[]
ArregloD: System.Boolean[]
ArregloE: System.Double[]

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

Ilustración 18: Tipo implícito de arreglo

Tipo implícito en arreglo de arreglos

C/019.cs

```
namespace Ejemplo;

internal 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());
        Console.WriteLine("Tipo ArregloB: " + arregloB.GetType());
    }
}
```

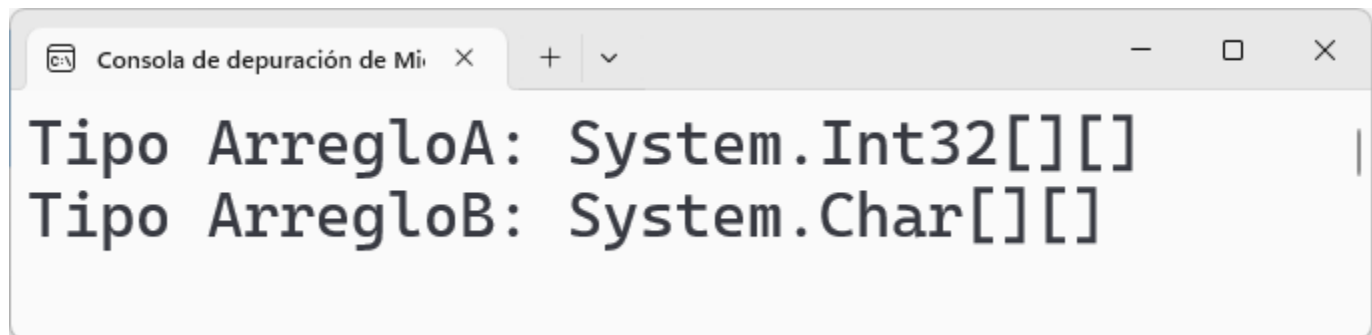


Ilustración 19: Tipo implícito en arreglo de arreglos

Convertir una cadena en un arreglo de cadenas al dividirla

C/020.cs

```
namespace Ejemplo;

internal class Program {
    static void Main() {
        //Una cadena
        string Cadena = "Esta es una frase";

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

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

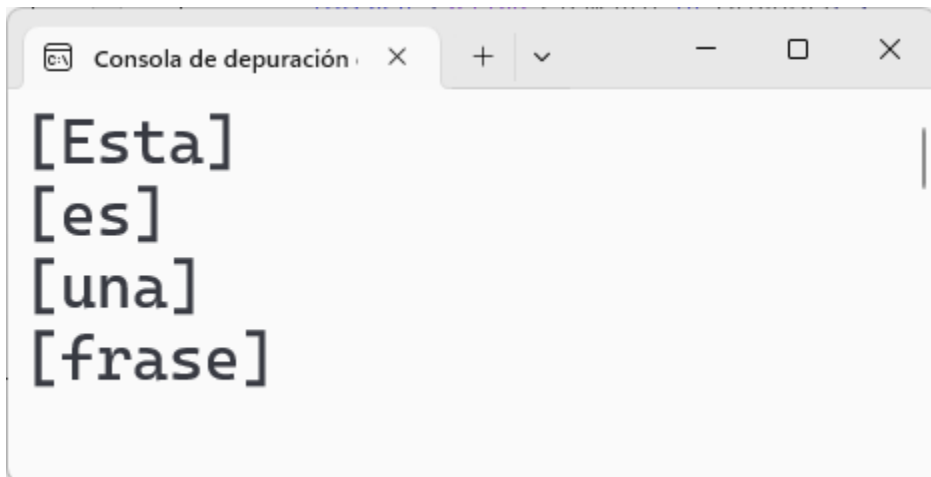


Ilustración 20: Convertir una cadena en un arreglo de cadenas al dividirla

Sin embargo, si hay más de un espacio intermedio.

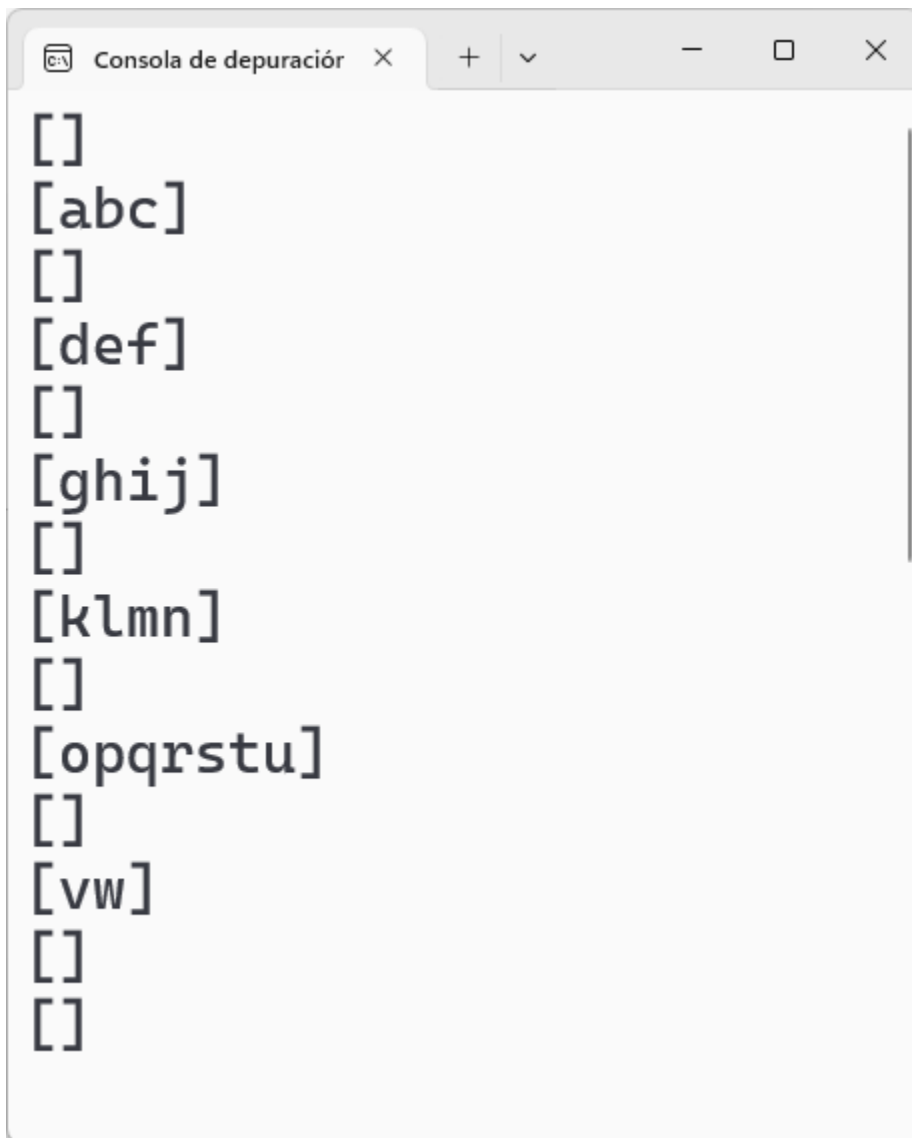
C/021.cs

```
namespace Ejemplo;

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

        //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 + "]");
        }
    }
}
```



```
Consola de depuraci3n X + v - □ X  
[]  
[abc]  
[]  
[def]  
[]  
[ghij]  
[]  
[klmn]  
[]  
[opqrstu]  
[]  
[vw]  
[]  
[]
```

Ilustraci3n 21: Cuando hay m1s de un espacio intermedio