```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace estudio_array
{
    class Program
    {
        static void Main(string[] args)
        {
            // Declarar un array sin definir el máximo de valores que puede contener
            int[] int_variable = new int[] { };

            // Declarar un array definiendo el máximo de valores que puede contener
            int[] int_fijo = new int[4];

            // Declarar un array definiendo sus valores sin indicar el máximo de valores que
            puede contener
            // El valor máximo es equivalente al número de elementos insertados
            int[] int_definido = new int[] { 1, 2, 3, 4 };

            // Declarar un array definiendo sus valores indicando, además, el máximo de
            valores que puede contener
            // (Se debe insertar tantos valores como número máximo de valores definido)
            int[] int_definido_fijo = new int[3] { 1, 2, 3 };

            // Vamos a crear una array de valores para trabajar con él
            int[] numeros = new int[] { 1, 5, 9, 2, 3, 8, 4, 4, 2, 6, 9 };
            Console.WriteLine("Array inicial: {0}", String.Join(", ", numeros));
            Console.WriteLine();

            // Vamos a ordenar el array y a guardarlo en la misma
            int[] numeros_copia = (int[])numeros.Clone();  // Copia del array
            Array.Sort(numeros_copia);  // Ordenar el array
            Console.WriteLine("Array ordenador en la misma variable: {0}", String.Join(", ",
            numeros_copia));  // Mostrar el array

            // Vamos ordenar el array a la inversa en la misma variable
            Array.Reverse(numeros_copia);  // Invertir el orden del array
            Console.WriteLine("Array ordenador de manera descendente en la misma variable:
            {0}", String.Join(", ", numeros_copia));  // Mostrar el array

            // Vamos a ordenar el array almacenando el valor en otro array sin modificar el
            existente
            //     Dado que OrderBy devuelve un objeto de tipo IOrderedEnumerable<int> y
            nosotros queremos
            //     un array tenemos que convertirlo añadiendo al final con .ToArray()
            int[] numeros_ordenados = numeros.OrderBy(x => x).ToArray();  // Ordenar el array
            Console.WriteLine("Array ordenado en otra variable: {0}", String.Join(", ",
            numeros_ordenados));  // Mostrar el array

            // Vamos a ordenar el array de manera inversa almacenando el valor en otro array
            sin modificar el existente
            //     Dado que OrderByDescending devuelve un objeto de tipo
            IOrderedEnumerable<int> y nosotros queremos
```

```
//     un array tenemos que convertirlo añadiendo al final con .ToArray()
int[] numeros_ordenados_inverso = numeros.OrderByDescending(x => x).ToArray();
// Ordenar el array de manera descendente
Console.WriteLine("Array ordenado en orden descendente en otra variable: {0}",
String.Join(", ", numeros_ordenados_inverso));  // Mostrar el array

// Vamos a comprobar si todos los valores del array cumplen una condicion
bool todos_menor_que_10 = numeros.All(x => x < 10);  // ¿Son todos los valores
menor que 10?
bool todos_mayor_que_5 = numeros.All(x => x > 5);  // ¿Son todos los valores
mayor que 5?
Console.WriteLine("¿Son todos los valores menor que 10?: {0}", todos_menor_que_10
 ? "Sí" : "No");  // Mostrar el resultado de la primera comprobación
Console.WriteLine("¿Son todos los valores mayor que 5?: {0}", todos_mayor_que_5 ?
 "Sí" : "No");  // Mostrar el resultado de la segunda comprobación

// Ahora vamos a comprobar si hay algún valor que cumpla la condición
bool alguno_menor_que_2 = numeros.Any(x => x < 2);  // ¿Hay algún valor menor
que 2?
bool alguno_menor_que_1 = numeros.Any(x => x < 1);  // ¿Hay algún valor menor
que 1?
Console.WriteLine("¿Hay algún valor menor que 2?: {0}", alguno_menor_que_2 ? "Sí"
 : "No");  // Mostrar el valor de la primera comprobación
Console.WriteLine("¿Hay algún valor menor que 1?: {0}", alguno_menor_que_1 ? "Sí"
 : "No");  // Mostrar el valor de la segunda comprobación

// Vamos a eliminar los valores repetidos
//     Dado que Distinct devuelve un objeto de tipo IEnumerable<int> y nosotros
queremos
//     un array tenemos que convertirlo añadiendo al final con .ToArray()
int[] numeros_sin_repetir = numeros.Distinct().ToArray();  // Eliminar los
valores repetidos
Console.WriteLine("Array de números sin repetir: {0}", String.Join(", ",
numeros_sin_repetir));  // Mostrar el resultado

// Vamos a filtrar el array
//     Dado que Where devuelve un objeto de tipo IEnumerable<int> y nosotros
queremos
//     un array tenemos que convertirlo añadiendo al final con .ToArray()
int[] numeros_pares = numeros.Where(x => x % 2 == 0).ToArray();  // Obtener los
números pares
int[] numeros_impares = numeros.Where(x => x % 2 == 1).ToArray();  // Obtener
los números impares
int[] numeros_divisible_entre_4 = numeros.Where(x => x % 4 == 0).ToArray();  //
Obtener los números divisibles entre 4
int[] filtro_funcion = numeros.Where(FuncionFiltro).ToArray();  // Filtrar
número a través de función (Está definida al final del archivo)
Console.WriteLine("Números pares: {0}", String.Join(", ", numeros_pares));  //
Mostramos valor del primer resultado obtenido
Console.WriteLine("Números impares: {0}", String.Join(", ", numeros_impares));
// Mostramos valor del segundo resultado obtenido
Console.WriteLine("Números divisibles entre 4: {0}", String.Join(", ",
numeros_divisible_entre_4));  // Mostramos valor del tercer resultado obtenido
Console.WriteLine("Números filtrado por función: {0}", String.Join(", ",
filtro_funcion));  // Mostrar el valor del cuarto resultado
Console.ReadLine();
```

```
            // Vamos a aplicar un cambio a cada valor del array
            string[] numeros_entre_signos = numeros.Select(x => "<" + x + ">").ToArray();
            // Mostrar cada valor entre signos de menor-mayor y mayor-menor
            Console.WriteLine("Numéros entre signos: {0}", String.Join(" ",
            numeros_entre_signos));
        }

        // Filtra los números obteniendo sólo aquellos entre 4 y 8 (estos incluidos)
        private static bool FuncionFiltro(int x)
        {
            return x >= 4 && x <= 8;
        }
    }
}
```