Índice

Ejercicios Unidad 8

Descargar estos ejercicios

Antes de empezar

Para realizar estos ejercicios, deberás descargar los recursos del enlace de proyecto_arrays. Como puedes ver, la solución está compuesta de varios proyectos. Cada uno de ellos corresponde con un ejercicio, deberás implementar todo el código, tanto de la Main como de los métodos que se piden en cada ejercicio. Cada proyecto contiene el test correspondiente, que deberás pasar para comprobar que has hecho el ejercicio correctamente.

Ejercicio 1. Números aleatorios en posiciones múltiplo de 4

Rellena un array de **10 números** de tipo **double**, de forma aleatoria, y visualiza los que estén en una **posición** que sea múltiplo de cuatro. Este ejercicio estará formado por varios métodos que definirás en el proyecto ejercicio1.

```
Ejercicio 1: Números aleatorios en posiciones múltiplo de 4

Array completo:
[0]: 45,67
[1]: 23,89
[2]: 78,45
[3]: 12,34
[4]: 89,23
[5]: 56,78
[6]: 34,56
[7]: 67,89
[8]: 91,23
[9]: 43,21

Números en posiciones múltiplo de 4:
Posición [0]: 45,67
Posición [4]: 89,23
Posición [8]: 91,23
```

Requisitos:

- Crear un método **GeneraNumerosAleatorios** que devuelva un array de 10 elementos de tipo `double`.
- Rellenar el array con números aleatorios entre 0 y 100.
- Crear un método **MuestraArrayCompleto** que muestre todos los elementos del array.
- Crear un método **MuestraMultiplosDe4** que muetre solo los elementos cuya posición sea múltiplo de 4 (posiciones 0, 4, 8).
- Usar `Random.NextDouble() * 100` para generar números reales aleatorios.
- Usar el operador módulo `%` para identificar posiciones múltiplo de 4.

Ejercicio 2. Conversión mayúsculas y minúsculas

Rellena un array de **10 caracteres** de forma aleatoria y luego modifica el mismo array de forma que los elementos que estén en mayúsculas pasen a ser minúsculas y los que estén en minúsculas pasen a mayúsculas. Este ejercicio estará formado por varios métodos que definirás en el proyecto ejercicio2.

```
Ejercicio 2: Conversión mayúsculas y minúsculas

Array original:
A b C d E f G h I j

Array modificado:
a B c D e F g H i J
```

Requisitos:

- Crear un método **GenerarCaracteresAleatorios** que devuelva un array de 10 caracteres generados aleatoriamente.
- Generar caracteres aleatorios (letras mayúsculas y minúsculas).
- Recuerda la correspondencia entre los caracteres ASCII y los números: 65-90 (A-Z) y 97-122 (a-z).
- Crea el método **ConvierteMayusculasMinusculas** que convierta las mayúsculas a minúsculas y viceversa, de un array pasado como parámetro, usando `char.IsLower()` y `char.IsUpper()` para detectar el tipo de carácte y `char.ToLower()` y `char.ToUpper()` para las conversiones.
- Visualizar el array con `foreach` antes y después de la modificación. Usando el método MuestraArray, al que le llegará el array a mostrar.

Ejercicio 3. Elemento mayor y su posición

Crea un array de 10 elementos, visualiza **el elemento mayor de la serie y la posición que ocupa**. Si hay varios iguales, muestra solo el primero. Este ejercicio estará formado por varios métodos que definirás en el proyecto ejercicio3.

```
Ejercicio 3: Elemento mayor y su posición

Array: [45, 23, 78, 12, 89, 56, 34, 67, 91, 43]

El elemento mayor es: 91
Posición del elemento mayor: 8
```

Requisitos:

- Método **GeneraArrayAleatorio** que Crea un array de 10 números enteros aleatorios y lo devuelve.

- Método **EncuentraMayor** al que le llega un array de enteros y encuentra y devuelve el valor máximo del array.
- Método **EncuentraPosicionMayor** al que le llega un array de enteros y encuentra la primera posición donde aparece ese valor máximo y la devuelve. Si hay valores repetidos que sean máximos, mostrar solo la primera ocurrencia.
- Método **MuestraArray** al que le llega un array de enteros y muestra su contenido como se muestra en la imagen.
- Programa principal que consiga una salida como la que se espera en el ejercicio.

Ejercicio 4. Verificador de números capicúa

Implementa un programa que dado un número entero introducido por teclado, determine si es **capicúa** usando arrays. Este ejercicio estará formado por varios métodos que definirás en el proyecto ejercicio4.

```
Ejercicio 4: Verificador de números capicúa

Introduce un número: 1234321
El número 1234321 es capicúa.

Introduce un número: 12345
El número 12345 no es capicúa.
```

Requisitos:

- Método **LeeNumero** que lea un número como string y convertirlo a array de caracteres usando `ToCharArray()` y lo devuelva.
- Método **EsCapicua** al que le llega el array de caracteres y devuelve si es capicua:
  - Comparar el array original con su versión invertida.
  - Usar operadores de intervalo para resolverlo.
  - Determinar si el número es capicúa comparando caracteres desde los extremos hacia el centro.
- Mostrar el resultado de forma clara.

Ejercicio 5. Desplazamiento circular a la derecha

Introduce un array de 10 elementos y **desplaza** todos sus componentes una posición hacia la derecha, colocando el último en la primera posición. Visualiza el array antes y después del desplazamiento. Este ejercicio estará formado por varios métodos que definirás en el proyecto ejercicio5.

```
Ejercicio 5: Desplazamiento circular a la derecha

Introduce 10 números:
Número 1: 10
Número 2: 20
Número 3: 30
Número 4: 40
Número 5: 50
Número 6: 60
Número 7: 70
Número 8: 80
Número 9: 90
Número 10: 100

Array original:
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

Array después del PRIMER desplazamiento:
[100, 10, 20, 30, 40, 50, 60, 70, 80, 90]

Array después del SEGUNDO desplazamiento:

[90, 100, 10, 20, 30, 40, 50, 60, 70, 80]
```

Requisitos:

- Crear un array de 10 números enteros introducidos por el usuario usando un método **LeeNumeros** que lo devuelva.
- Crea el método **DesplazaDerechaCircular** que implemente el desplazamiento circular guardando el último elemento y moviendo todos los elementos una posición a la derecha mediante un bucle y colocando el último en la primera posición, la modificación se hará sobre el mismo array.
- Crea el método **DesplazaDerechaCircularConRangos** al que le llega un array y devuelva otro array con los elementos desplazado. Esta vez no se podrán usar bucles, se usarán los operadores de intervalos y propagación. *Valora el motivo por el cual se ha tenido que devolver el array creado y no se a asignado al mismo de entrada.*

- Crea el método **MuestraArray** al que le llega un array y lo muestra.
- Crea el programa principal, para que muestre el array antes y después de los dos desplazamientos como se ve en la salida.

Ejercicio 6. Proyecto gestor de calificaciones

Este ejercicio estará formado por varios métodos que definirás en el proyecto ejercicio6.

```
Ejercicio 6: Gestor de calificaciones
¿Cuántos alumnos hay en la clase? 4
Introduce las calificaciones:
Alumno 1: 8,5
Alumno 2: 6,2
Alumno 3: 9,1
Alumno 4: 7,8

--- ESTADÍSTICAS DE LA CLASE ---
Calificaciones: 8,50 - 6,20 - 9,10 - 7,80
Promedio de la clase: 7,90
Calificación más alta: 9,10
Calificación más baja: 6,20
Número de aprobados: 4
Número de suspensos: 0

Presiona cualquier tecla para salir...
```

Requisitos:

- Método llamado **LeeCalificaciones** que pida el número de alumnos y sus calificaciones, devolviendo un array de tipo `double[]`.
- Método llamado **CalculaPromedio** que reciba un array de calificaciones y devuelva el promedio como `double`. Usa foreach para resolverlo.
- Método llamado **ObtieneMaxima** que reciba un array de calificaciones y devuelva la calificación más alta. Usa foreach para resolverlo.
- Método llamado **ObtieneMinima** que reciba un array de calificaciones y devuelva la calificación más baja. Usa foreach para resolverlo.
- Método llamado **CuentaAprobados** que reciba un array de calificaciones y devuelva cuántos alumnos aprobaron (nota >= 5.0). Usa foreach para resolverlo.
- Método llamado **MuestraEstadisticas** que reciba el array de calificaciones y muestre toda la información formateada.

Ejercicio 7. Proyecto análisis de temperaturas

Escribe un programa en el proyecto ejercicio7 que gestione las temperaturas de una semana.

```
Ejercicio 7: Análisis de temperaturas

Temperaturas de la semana:
Lunes: 5,3°C
Martes: 28,4°C
Miércoles: 12,4°C
Jueves: 6,8°C
Viernes: 25,6°C
Sábado: 30,6°C
Domingo: 32,3°C

--- ANÁLISIS SEMANAL ---
Temperaturas: L:5,30 M:28,44 X:12,38 J:6,78 V:25,61 S:30,62 D:32,25
Temperatura media: 20,20°C
Temperatura máxima: 32,25°C (Domingo)
Temperatura mínima: 5,30°C (Lunes)
Días con temperatura superior a la media: 4

--- TEMPERATURAS POR ENCIMA DE 25°C ---
Martes: 28,44°C
Viernes: 25,61°C
Sábado: 30,62°C
Domingo: 32,25°C

Presiona cualquier tecla para salir...
```

Requisitos:

- Método **LeeTemperaturas** que devuelva un array `double[7]` con las temperaturas de la semana, estás serán generadas aleatoriamente con números que irán desde 4 a 35 y serán de tipo double, usa el método `NextDouble` de la clase `Random`.
- Método **MuestraTemperaturas** al que le llega un array y muestra las temperaturas como en la salida. Crea un array de string para guardar los nombres de los días de la semana.
- Método **CalculaMedia** que calcule la temperatura media de la semana y la devuelva.
- Método **BuscaMaximo** que devuelva una tupla `(double temperatura, int dia)` con la temperatura máxima y el día.
- Método **BuscaMinimo** que devuelva una tupla `(double temperatura, int dia)` con la temperatura mínima y el día.
- Método **CuentaDiasSuperioresA** que reciba un array de temperaturas y un valor umbral, devolviendo cuántos días superaron ese valor.

- Método **TemperaturasAltas** que devuelve un array con las temperatura superior a 25°C.
- En la Main crea dos array de `string[]` uno con los nombres de los días de la semana y otro con los nombres abreviados (primera letra). Estos array se usarán para conseguir una salida como la que se expone en el ejercicio.

Ejercicio 8. Proyecto estadísticas de ventas

El proyecto 8 analizará las ventas mensuales de una empresa durante un año. Este ejercicio estará formado por varios métodos que definirás en el proyecto ejercicio8.

```
Ejercicio 8: Estadísticas de ventas

Introduce las ventas mensuales del año:
Enero: 15000,50
Febrero: 18500,75
Marzo: 22000,00
Abril: 19500,25
Mayo: 25000,80
Junio: 28000,90
Julio: 32000,60
Agosto: 29500,45
Septiembre: 26500,30
Octubre: 24000,15
Noviembre: 27500,85
Diciembre: 35000,20

--- TOTAL ANUAL ---
Ventas totales: 302.559,75€

--- ANÁLISIS TRIMESTRAL ---
Q1 (Ene-Mar): 55.501,25€
Q2 (Abr-Jun): 72.501,95€
Q3 (Jul-Sep): 88.001,35€
Q4 (Oct-Dic): 86.501,20€

Mejor trimestre: Q3 con 88.001,35€

--- MESES CON VENTAS SUPERIORES AL PROMEDIO ---
Promedio mensual: 25.213,31€
Mayo: 25.000,80€
Junio: 28.000,90€
Julio: 32.000,60€
Agosto: 29.500,45€
Septiembre: 26.500,30€
Noviembre: 27.500,85€
Diciembre: 35.000,20€
```

Requisitos:

- Como puedes ver en el ejercicio que se pasa como base, tienes un array de strings, estático y de solo lectura con los meses del año, deberás usarlo en los métodos necesarios para conseguir la salida esperada.
- Método **LeeVentasMensuales** que devuelva un array `double[]` con las ventas de cada mes.
- Método **CalculaVentasTrimestrales** que reciba las ventas mensuales y devuelva un array `double[]` con las ventas por trimestre. Deberás usar foreach con intervalos para calcular cada trimestre.

- Método **BuscaMejorTrimestre** que devuelva una tupla `(int trimestre, double ventas)` con el mejor trimestre.
- Método **MuestraMesesSuperioresAlPromedio** que calcule el promedio de venta de todos los meses y los muestre, después que muestre solo los meses con ventas superiores al promedio.

Ejercicio 9. Proyecto procesador de texto simple

Crea un programa que analice un texto ingresado por el usuario. Este ejercicio estará formado por varios métodos que definirás en el proyecto ejercicio9.

```
Ejercicio 9: Procesador de texto simple

Introduce un texto: La programación es divertida y la programación es útil

--- ANÁLISIS DEL TEXTO ---
Texto original: La programación es divertida y la programación es útil
Número de palabras: 9
Palabra más larga: programación (12 caracteres)
Palabra más corta: y (1 caracteres)

--- FRECUENCIA DE PALABRAS ---
1. la: 2 veces
2. programación: 2 veces
3. es: 2 veces
4. divertida: 1 vez
5. y: 1 vez
6. útil: 1 vez

--- PALABRAS ÚNICAS (sin repetir) ---
La, programación, es, divertida, y, útil

Total de palabras únicas: 6

Presiona cualquier tecla para salir...
```

Requisitos:

- Método **SeparaPalabras** que reciba un texto y devuelva un array `string[]` con las palabras (usar `Split`).
- Método **BuscaPalabraMasLarga** que devuelva la palabra con más caracteres.
- Método **BuscaPalabraMasCorta** que devuelva la palabra con menos caracteres.

- Método **ObtenPalabrasUnicas** que devuelva un array con las palabras sin repetir usando `operadores de intervalo y propagación` para añadir una palabra al array. Dos palabras serán iguales si sus caracteres son los mismos sin tener en cuenta las minúsculas y mayúsculas.
- Método **CuentaFrecuenciaPalabras** que muestre cuántas veces aparece cada palabra diferente. Usará el método anterior para obtener las palabras únicas del texto. Mostrará la frecuencia en el propio método.
- Todas las comparaciones deben ser insensibles a mayúsculas y minúsculas.

Ejercicio 10. Proyecto calculadora de vectores

Implementa operaciones básicas con vectores (arrays de números). Este ejercicio estará formado por varios métodos que definirás en el proyecto ejercicio10.

```
Ejercicio 10: Calculadora de vectores

Introduce el tamaño de los vectores: 4

Vector A:
Componente 1: 1,5
Componente 2: 2,3
Componente 3: -0,8
Componente 4: 3,2

Vector B:
Componente 1: 2,1
Componente 2: -1,4
Componente 3: 4,5
Componente 4: 0,7

--- OPERACIONES CON VECTORES ---
Vector A: [1,50, 2,30, -0,80, 3,20]
Vector B: [2,10, -1,40, 4,50, 0,70]

Suma (A + B): [3,60, 0,90, 3,70, 3,90]
Resta (A - B): [-0,60, 3,70, -5,30, 2,50]
Producto escalar (A · B): -1,43
Magnitud del vector A: 4,29
Magnitud del vector B: 5,21

Presiona cualquier tecla para salir...
```

Requisitos:

- Método **LeeVector** al que le llega el nombre del vector (ejemplo: A) y el tamaño de este y que pida al usuario los componentes de este y los devuelva en un array `double[]`.
- Método **MuestraVector** que muestre un vector en formato `[x, y, z, ...]`. Como se ve en la salida
- Método **SumaVectores** al que le llegan dos vectores y devuelve un nuevo vector suma de los dos de entrada. Sumando cada uno de sus componentes (por posición).
- Método **RestaVectores** igual que el anterior pero con la resta.
- Método **ProductoEscalar** que calcule el producto escalar entre dos vectores. Será la suma de los productos de los componentes de los vectores de entrada (por posición). Este método devolverá el producto escalar como doble.
- Método **CalculaMagnitud** que calcule la magnitud (norma) de un vector usando `Math.Sqrt`. Para ello se deberán sumar todos los componentes al cuadrado del vector. La magnitud será la raíz cuadrada del total de la suma.
- **MuestraOperacionVector**, método al que le llega el vector resultado de una operación y el nombre de la operación y lo muestra como en la salida.