Documentar el codi

Exercici nº17

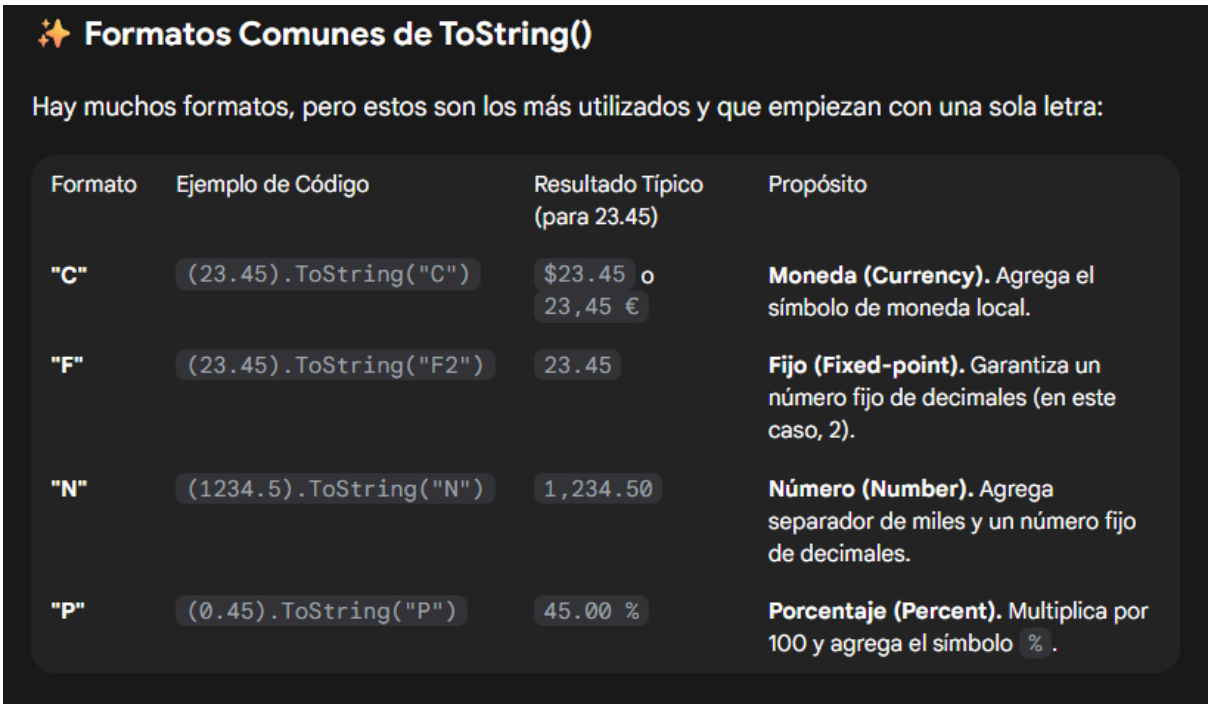
Enunciat:

/* Hi ha una secta d'adoradors dels decimals que volen que els hi feu un programa que a partir d'un nombre real,
ex. 4.56, només ens retorni els decimals, 0,56. Als números que només tenen decimals els anomenen "nombres meravellosos"

Número lleig: 23,45
Nombre meravellós: 0,45

Número lleig: 8,5
Nombre meravellós: 0,5
*/

Documentar
Utilització del ToString()



✨ **Formatos Comunes de ToString()**

Hay muchos formatos, pero estos son los más utilizados y que empiezan con una sola letra:

Formato	Ejemplo de Código	Resultado Típico (para 23.45)	Propósito
"C"	`(23.45).ToString("C")`	`$23.45` o `23,45 €`	**Moneda (Currency).** Agrega el símbolo de moneda local.
"F"	`(23.45).ToString("F2")`	`23.45`	**Fijo (Fixed-point).** Garantiza un número fijo de decimales (en este caso, 2).
"N"	`(1234.5).ToString("N")`	`1,234.50`	**Número (Number).** Agrega separador de miles y un número fijo de decimales.
"P"	`(0.45).ToString("P")`	`45.00 %`	**Porcentaje (Percent).** Multiplica por 100 y agrega el símbolo `%`.

Es una diferencia sutil pero importante entre cómo C# maneja los tipos de datos.
La verdad es que, en términos de **resultado numérico**, `(int)` y `Math.Truncate()` a menudo son **idénticos**: ambos eliminan la parte decimal, moviendo el número **hacia el cero** (truncamiento).

Operación	Valor de Prueba (5.9)	Valor de Prueba (-5.9)
`(int)5.9`	5	-5
`Math.Truncate(5.9)`	5.0	-5.0

🔑 La Diferencia Clave: El Tipo de Retorno

La distinción principal es el **tipo de dato que devuelven**:

- **Casting Explícito `(int)`**
  - Fuerza una **conversión de tipo**.
  - **Resultado:** Es un tipo **`int`** (entero sin decimales).
  - *Ejemplo:* Convierte un `double` a un `int`.
- **`Math.Truncate()`**
  - Es una **función matemática**.
  - **Resultado:** Devuelve la parte entera, pero mantiene el tipo de dato original (`double` o `decimal`).
  - *Ejemplo:* Si le pasas un `double`, te devuelve un `double` (ej. `5.0`).

Usar `(int)` es la forma más directa de obtener un entero, mientras que `Math.Truncate()` se usa a menudo si quieres manipular la parte entera de un número **sin perder su precisión decimal** o si quieres seguir trabajando con el tipo `double` después.

Ahora que entendemos que ambos truncan hacia cero, ¿te gustaría explorar la Opción 1 y ver cómo `Math.Floor()` y `Math.Ceiling()` manejan los números negativos de manera diferente a `Math.Truncate()`?

Es una función matemática. Devuelve la parte entera, pero mantiene el tipo de dato original (double o decimal).

```
PS C:\DevOp\Local\Prog_Coral\practica\1-Programes-basics\exercicis\exercici17> dotnet run

Exertcici n°. 17.

Escriu Un numero amb decimals: 23,45
Nombre meravellós: 0,45

Un altra solucio:
Segona metode
amb double.Parse(String) Convierte una representacio en cadena de texto en su valor numerico
amb (int) Conversión Explícita o Cast (Moldeado)
Nombre meravellós: 0,44999999999999993 format numero.

Nombre meravellós: 0,45 format String

Un altra solucio:
Tercer Metode
amb Math.Truncante  Es una función matemática. Devuelve la parte entera,
pero mantiene el tipo de dato original (double o decimal).

Nombre meravellós: 0,44999999999999993 format numero.

Nombre meravellós: 0,45

Un altra solucio:
Quart Metode
amb amb Poencia de 10. Math.Pow  Es una función matemática.
C# s'utilitza per calcular la potència d'un nombre.

Nombre meravellós: 0,45
Fi.
PS C:\DevOp\Local\Prog_Coral\practica\1-Programes-basics\exercicis\exercici17>
```