Caracteres y cadenas de texto

Uno de los elementos más utilizados, sino el que más que mostramos de cara al usuario cuando programamos son las cadenas de texto, comúnmente llamadas frases, por lo tanto en este video vamos a ver como operar con ellos.

Índice

1 - El tipo char

Lo primero que veremos es el tipo carácter, llamado tipo `char` , que es un único carácter

El cual tiene varias características:

- Son los datos predefinidos del tipo char y ocupan 1 byte en memoria.
- para indicar un carácter debemos ponerlo entre comilla simple. si utilizamos dos será considerado como una cadena.

```
char caracter = 'k';
```

esa es la forma de definir un carácter, pero en circunstancias normales no se utilizan nunca, lo que si utilizamos es el tipo string.

2 - El tipo string

Lo que si utilizamos, y mucho, son las cadenas de texto, osea varios elementos tipo char uno detrás de otro, haciendo frases. Esto se llama tipo `string` .

Hay varias formas de crear un string, la primera es directa y la más sencilla:

- Asignación de forma directa.

```
string test = "NetMentor";
```

- La segunda es utilizando un array de chars, veremos lo que son los arrays en el siguiente post.

```
char[] name = { 'N', 'E', 'T'};
string stringName = new string(name);
```

- La tercera es concatenando dos variables

```
var nombreConcatenado = name1 + name2
```

- finalmente podemos utilizar *string interpolation*, que podemos ver más detalladamente, en otro video, pero que de primeras diremos que puede tener variables dentro siempre y cuando las introducimos entre

llaves.

```
var stringConVariables = $"el nombre es {variable}.";
```

2.1 - Formatear un string

Normalmente cuando queremos mostrar datos, suelen ser datos de un tipo, ya sea porcentaje, dinero, un formato específico de fecha o muchas otras opciones, ya que es mucho mejor mostrar 50€ que un simple 50, o el símbolo de porcentaje cuando hablamos de porcentajes

Para ello utilizamos un metodo que esta dentro de la clase string, llamado "*format*" y lo Utilizamos de la siguiente manera:

```
string.Format("La temperatura actual es {0}°C.", 20.4);
```

Como primer parámetro pasamos la frase que queremos imprimir, dentro de la frase podemos incluir variables, como hemos indicado en el punto anterior. para ello introducimos entre corchetes el número de la variable {n}, ya que después como segundo parámetro, (y siguientes) incluimos las variables que queremos enseñar.

2.2 - Funciones de la clase string

Finalmente dentro de string veremos que por defecto string trae muchas funciones de uso común para así facilitar la vida de los desarrolladores y son las siguientes:

imaginémonos que tenemos un string como el siguiente:

```
string myString = "Tengo un vaso lleno de";
```

- Si lo queremos convertir a mayúsculas:

```
string mayusculas = myString.ToUpper();
```

- Si lo queremos convertir a minuscula

```
string minusculas = myString.ToLower();
```

- Si queremos comparar una frase con la otra, nos devolverá true or false

```
bool sonIguales = myString.Equals("Tengo un vaso lleno de");
```

- Si queremos saber la posición de un carácter

```
int posicion =  myString.IndexOf("o");
```

- Si queremos concatenar dos string

```
string frase =  myString + " zumo de naranja";
```

- Si queremos saber si una frase contiene cierta frase, nos devuelve true falso

```
bool contiene = myString.Contains("vaso");
```

3 - StringBuilder

Cuando creamos una variable string son inmutables, lo que significa que una vez se le asigna el valor esté no se puede cambiar. Si lo que queremos es actualizar un string, internamente lo que hace el código es crear un nuevo espacio de memoria, lo cual puede causar problemas de rendimiento.

Para evitar este problema o si necesitamos una cadena de texto multable tenemos el tipo `StringBuilder` .

```
StringBuilder sb = new StringBuilder();
```

En este caso si queremos añadir texto a la variable utilizamos el siguiente método:

```
sb.Append(“texto”);
```

el cual sí utiliza los mismos espacios de memoria y no crea duplicidades.

Para imprimirlo simplemente tenemos que llamar al método `.ToString()` dentro de `StringBuilder`

```
Console.Writeline(sb.ToString())
```

Comúnmente en la vida real utilizaremos la concatenación para añadir texto, ya que los pocos bytes que gasta no compensa la lógica de crear el objeto. pero si lo que vamos a hacer es montar un texto largo, como juntar varios xml, si nos puede interesar hacer un stringbuilder para que así utilice menos espacio en memoria.