



Programación

C #

TECNICATURA SUPERIOR EN PROGRAMACIÓN
LABORATORIO DE COMPUTACIÓN II



UTN
Facultad Regional
San Francisco

Clase String



- Es una **colección secuencial de objetos Char (caracteres)** que representan una cadena.
 - `string saludo = "Hola mundo";`
- Se usa **String** para contener varios caracteres sin la sobrecarga de administración de una matriz **Char[]**.
- Un objeto String es inmutable (de sólo lectura): **no se puede modificar su valor una vez que se ha creado**.
 - Los métodos que “modifican” objetos String, en realidad devuelven uno nuevo.
 - Para modificar realmente el contenido de una cadena, se utiliza la clase *System.Text.StringBuilder*.

FUNCIONALIDAD

- La clase String proporciona métodos para:
 - Comparar objetos String,
 - Devolver el índice de un carácter o una cadena en un objeto String,
 - Copiar el valor de un objeto String,
 - Dividir una cadena o combinar cadenas,
 - Modificar el valor de una cadena,

Clase String

funcionalidad

OPERADORES

- Operador +

- Permite concatenar cadenas. Por ejemplo:

```
string saludo = "Hola " + "mundo"; // saludo = "Hola mundo"
```

- De igualdad: == y !=

- Se definen para comparar los **valores** de objetos **string**, no las referencias. Por ejemplo:

```
string saludo = "Hola mundo", saludo2 = "Hola";  
saludo2 += " mundo"; // Agrego contenido a la saludo2  
Console.WriteLine(saludo == saludo2); // Salida: true  
Console.WriteLine((object)saludo == (object)saludo2); // Salida: false
```

Clase String

funcionalidad

LITERALES DE CADENA



- Son objetos de tipo **string** que se pueden escribir de dos formas:

- entre comillas, por ejemplo:

- "esto es un literal"

- entre comillas y precedidos de @, por ejemplo:

- @"c:\Docs\Source\a.txt" // En lugar de "c:\\Docs\\Source\\a.txt"

En este caso, las secuencias de escape *no* se procesan

- Pueden contener cualquier literal de carácter, incluidas las secuencias de escape. Por ejemplo:

- string a = "\\u0066"; // Salida: "f" → \\ = \, u0066 = f

Clase String


funcionalidad

OPERADORES



- El operador `[]` se puede utilizar para tener acceso a caracteres individuales de un objeto string a través de un índice. Un índice es la posición de un carácter en una cadena:

```
string cadena = "Contenido";
```



```
char x = cadena[3]; // x = 'n';
```

- Ejercicio 1:
 - Realizar un programa que corrobore la identidad de un usuario. Éste deberá ingresar su nombre, y el sistema deberá controlar que no haya ingresado una cadena vacía. Si no es así, se le debe pedir el apellido al usuario y una vez corroborado este dato también, se le pedirá una contraseña. Si ésta es "demo", entonces el sistema mostrará el mensaje "Acceso concedido a (*nombre y apellido del usuario*)" , caso contrario "No tiene permisos para acceder".

Clase String

funcionalidad

CREAR CADENAS NUEVAS



- **Format**

- Reemplaza un **elemento de formato** de una cadena por el **equivalente textual**. Por ejemplo:

```
string nombre = "Federico", apellido = "Perez";  
string.Format("Mi nombre es: {0}, {1}", apellido, nombre);  
// El resultado es: "Mi nombre es: Perez, Federico"
```

Elementos de formato:

- "{0}", índice: 0, se corresponde con: apellido
- "{1}", índice: 1, se corresponde con: nombre.

- **Ejercicio 2:**

- Modifique el ejercicio anterior y agregue una línea para mostrar "Acceso concedido a (*nombre y apellido del usuario*)" si el logueo es correcto, usando este método. ¿Qué diferencias encuentra? ¿Cuándo utilizaría el operador "+" y cuándo este método?

Clase String

funcionalidad

CREAR CADENAS NUEVAS



- **Concat**

- Concatena cadenas (o la representación en cadena de otros objetos) una a continuación de la otra. Por ejemplo:

```
int i = -123;
string cadena = "cad";
Object[] objs = new Object[] {-123, -456, -789};
string cadena_concatenada = String.Concat(i, cadena);
Console.WriteLine("1) {0}", cadena_concatenada); // Salida: "1) -123cad"
Console.WriteLine("2) {0}", String.Concat(i, i, i)); // Salida: "2) -123-123-123"
Console.WriteLine("3) {0}", String.Concat(objs)); // Salida: "3) -123-456-789"
```

- **Ejercicio 3:**

- Si tuviera que mostrar el nombre y apellido de una persona por pantalla, y éstos fueron ingresados de manera independiente, ¿cómo lo haría usando este método? ¿Y si tuviera que incluir ", DNI: (numero documento)"?

Clase String

funcionalidad

CREAR CADENAS NUEVAS

- Join

- Concatena una cadena (separador) entre cada uno de los elementos de una matriz **String**, generando una sola cadena concatenada. Por ejemplo:

```
string[] saludo = { "¡Hola", "y", "bienvenido!"};  
Console.WriteLine(String.Join("-", saludo));  
// Salida: "¡Hola-y-bienvenido!"
```

- Ejercicio 4:

- Hacer un programa que le permita al usuario ingresar palabras al azar, guardarlas en un arreglo y luego permitirle al usuario que las visualice separadas por espacio, punto y coma, guión, asterisco o punto, según la opción que elija (1, 2, 3, 4 o 5) en un menú que el programa visualice. Repetir las veces que el usuario desee.

Clase String


funcionalidad

CREAR CADENAS NUEVAS

- Insert

- Inserta una cadena String dentro de otra, en la posición especificada. Por ejemplo:

```
string cadena = "Hace tiempo";  
                0 1 2 3 4 5 6 7 8 9 10  
string cadenaResultado = cadena.Insert(4, "mucho");  
Console.WriteLine(cadenaResultado); // Salida: Hace mucho tiempo.
```



- Ejercicio 5:

- Realice un cuadro comparativo con todos los métodos para crear cadenas vistos hasta el momento. Indique en qué casos utilizaría cada uno, ventajas y ejemplos.

Clase String

funcionalidad

CREAR CADENAS NUEVAS

- CopyTo

- Copia cierta cantidad de caracteres dentro de una matriz, en la posición especificada. Por ejemplo:

cadenaOrigen.CopyTo (int posicionCadenaOrigen, char[] destino, int posicionDestino, int cantidadElementos)



// Salida: "Dhole"

Clase String

funcionalidad

RECORTAR Y QUITAR CARACTERES

- Trim

- Quita un conjunto de caracteres, del principio y del final de una cadena.

```
string nombre = "Juan";
```

```
string nombreIngresado = " Juan ";
```

```
Console.WriteLine(nombre == nombreIngresado); // Falso
```

```
Console.WriteLine(nombre == nombreIngresado.Trim()); // True
```

```
string str5 = "*;|@123***456@|;*";
```

```
char[] delim = new char[] { '*', ';', '|', '@' };
```

```
Console.WriteLine(str5.Trim(delim)); // Salida: 123***456
```

Clase String

funcionalidad

RECORTAR Y QUITAR CARACTERES



- TrimStart

- Quita un conjunto de caracteres especificado en una matriz, sólo del **principio** de una cadena.

- TrimEnd

- Quita un conjunto de caracteres especificado en una matriz, sólo del **final** de una cadena.

Por ejemplo:

```
string caracteres, nombre, nombre2;  
caracteres = ";1@ ";  
nombre1 = "Andrés @1 @1;; "; nombre2 = " @1 @1;; Andrés";  
Console.WriteLine(nombre.TrimStart(caracteres.ToCharArray()));  
// Salida: "Andrés"  
Console.WriteLine(nombre.TrimEnd(caracteres.ToCharArray()));  
// Salida: "Andrés"
```

Clase String

funcionalidad

RECORTAR Y QUITAR CARACTERES

- Ejercicio 6:

Se parte de la siguiente matriz:

Usuarios = {"Juan", "José", "Andrés", "Ana", "Silvana", "Romina"}

Se requiere realizar un programa que le pida al usuario ingresar su nombre para corroborar su identidad y, si lo que ingresa es correcto (coincide con alguno de los valores de la matriz) se le solicita que ingrese una nueva clave de seguridad. Ésta debe cumplir lo siguiente:

- ✦ No tener espacios ni al inicio ni final,
- ✦ No comenzar con guiones ni puntos,
- ✦ No tener números al final de la cadena.

Controlar la clave ingresada, quitar los caracteres no válidos e imprimir por pantalla la clave correcta resultante.