Programación

C#

TECNICATURA SUPERIOR EN PROGRAMACIÓN
LABORATORIO DE COMPUTACIÓN II

UTN
Facultad Regional
San Francisco

Clase String
funcionalidad
CADENAS DE RELLENO

- PadLeft y PadRight
  - Alinea a la derecha o izquierda los caracteres de la cadena e inserta espacios en blanco o un carácter especificado hasta alcanzar la longitud total deseada. Por ejemplo:

```
string cadena;
cadena = "Contenido";
cadena = cadena.PadLeft(20, ';');
Console.WriteLine("-{0}-", cadena);
// Salida: "-;;;;;;;;;;;;Cadena-"
Console.WriteLine("-{0}-", cadena.PadRight(20, '.'));
// Salida: "-Cadena..... -"
```

COMPARAR CADENAS

- **Compare y CompareTo**
  - Compara dos cadenas. Devuelve 0 (cero) si son iguales, 1 si la primera cadena es mayor que la segunda o -1 en caso contrario. Por ejemplo:

```
string str1 = "ABCD", str2 = "abcde", str3 = "abcd";
int comparacion = string.Compare(str1, str2);
Console.WriteLine(comparacion); // Salida: "-1"
Console.WriteLine(string.Compare(str2, str1)); // Salida: "1"
```

- **CompareOrdinal**
  - Compara dos cadenas especificadas, teniendo en cuenta los valores numéricos de cada caracter. Diferencia entre mayúsculas y minúsculas. Por ejemplo:

```
string str1 = "ABCD", str2 = "abcd";
int comparacion = string.CompareOrdinal(str1, str2);
Console.WriteLine(comparacion); // Salida: "-32"
Console.WriteLine(string.CompareOrdinal(str2, str1)); // Salida: "32"
```

COMPARAR CADENAS

- **Equals**
  - Determina si una cadena y un objeto (otra cadena) tienen el mismo valor. Por ejemplo:

```
string cadena1 = "mar", cadena2 = "río", cadena3 = "mar";
bool mismo_valor = cadena1.Equals(cadena2);
Console.WriteLine(mismo_valor);       // False
Console.WriteLine(cadena1.Equals(cadena3));       // True
Console.WriteLine(string.Equals(cadena1, cadena2)); // False
Console.WriteLine(string.Equals(cadena1, cadena3)); // True
```

  - Ejercicio 8: ¿qué diferencia encuentra entre los distintos métodos de comparación de cadenas? ¿Cuándo utilizaría unos y otros?

CAMBIAR MAYÚSCULAS Y MINÚSCULAS

- **ToLower y ToUpper**
  - Devuelve una copia de la cadena pero convertida a minúsculas o mayúsculas respectivamente. Por ejemplo:

```
string cadenaPrueba = "PrUeBa";
cadenaPrueba =  cadenaPrueba.ToLower();
Console.WriteLine(cadenaPrueba); // Salida: "prueba"
Console.WriteLine(cadenaPrueba.ToUpper()); // Salida: "PRUEBA"
```

- Ejercicio 9:

Dada la matriz:

nombres = {" JuAn", "JOse ", " ANA ", " maria", "andreS   ", "LuCas "}

Realizar un programa que le permita al usuario buscar un nombre ingresado por pantalla y si lo encuentra lo muestre en mayúsculas y minúsculas. Caso contrario que le pida al usuario que vuelva a ingresar, teniendo como máximo 3 intentos posibles para que su entrada coincida con algún nombre de la matriz.

DIVIDIR CADENAS

- **Split**
  - Devuelve un arreglo String que contiene las subcadenas de una cadena que están delimitadas por ciertos caracteres especificados en una matriz Char. Por ejemplo:

Dividir Cadenas

- **Ejercicio 10:**
  - En un arreglo de cadenas se tiene la siguiente información (una línea en cada elemento del vector) :

    Gonzalez, Ricardo;ricardogon@gmail.com;35;467987

    Aima, Lucas;lucas_opt@gmail.com;25;497457

    Mariano, Andrea;andremari@gmail.com;30;4546788

    Gutierrez, Manuel;manugu9@gmail.com;23;-

    Rossi, Marcos;m_rossi@gmail.com;24;476899
  - Se necesita construir un programa que liste los nombres de los contactos en un menú para que el usuario seleccione la opción correspondiente, de acuerdo al contacto sobre el que desea ver más información. Por ejemplo:

    0. Primer contacto.

    1. Segundo contacto.

    2. Tercer contacto.
  - El usuario selecciona la opción "1" y el sistema muestra su mail, edad y teléfono.
  - El usuario debe tener la opción de volver a consultar los datos de otro contacto las veces que lo desee.

Buscar en Cadenas

- **StartsWith y EndsWith**
  - Determina si el principio de la cadena (o el final) coincide con otra pasada como parámetro. Devuelve true si coincide o false en caso contrario. Por ejemplo:

```
string buscarPrincipio, cadenaOriginal, buscarFinal;
cadenaOriginal = "Se busca al comienzo";
buscarPrincipio = "Se busca";
buscarFinal = "enzo";
bool coincide =  cadenaOriginal.StartsWith(buscarPrincipio);

Console.WriteLine(coincide);  // True
Console.WriteLine(cadenaOriginal.EndsWith(buscarFinal));      // True
Console.WriteLine(cadenaOriginal.StartsWith(buscarFinal));    // False
Console.WriteLine(cadenaOriginal.EndsWith(buscarPrincipio));   // False
```

BUSCAR EN CADENAS

- **Ejercicio 11:**

Se tiene la siguiente lista de carreras y universidades que las ofrecen:

Arquitectura, UTN Córdoba / Diseño gráfico, UTN Santa Fe / Diseño industrial, UTN Córdoba / Administración de empresas, UTN Santa Fe / Contador, UTN San Francisco / Turismo, UTN Buenos Aires / Medicina, UTN Córdoba / Química, UTN Mendoza / Historia, UTN La Plata / Ingeniería electrónica, UTN San Francisco / Ingeniería mecánica, UTN San Francisco / Ingeniería en sistemas, UTN San Francisco.

Se le solicita que realice un programa que le permita al usuario ingresar la carrera que busca y mostrar dónde se la dicta, o que ingrese una localidad y mostrar las carreras que ofrece, o busque y liste las carreras que empiecen con determinada letra y muestre dónde se pueden cursar.

Debe poder realizar tantas búsquedas como prefiera.

Buscar en Cadenas

- **IndexOf y LastIndexOf**
  - IndexOf devuelve el índice de la primera aparición de una cadena o un caracter dentro de la cadena que invoca al método; mientras que LastIndexOf es similar pero devuelve la posición de la última aparición.
  - Por ejemplo:

```
string oracion = "caminante no hay camino, se hace camino al andar";
string cadenaBuscada = "camino";
int indice = oracion.IndexOf(cadenaBuscada);
Console.WriteLine(indice);    // 17
Console.WriteLine(oracion.LastIndexOf(cadenaBuscada));        // 33
Console.WriteLine(oracion.IndexOf('m'));                      // 2
Console.WriteLine(oracion.LastIndexOf('m'));                  // 35
```

    - ¿Qué pasa si no encuentra coincidencia?

EJERCICIO REVISIÓN

- **Ejercicio 12:**
  - Se requiere un pequeño programa que le permita al usuario cargar los datos básicos de una factura: nombre y apellido del cliente, dirección y datos del producto (cantidad, nombre, precio sin IVA por unidad). El precio unitario con IVA (21%) incluido y el total los calcula el sistema. A modo de simplificación, se carga un único producto por factura. Debe poder cargar tantas facturas como requiera. El usuario podrá indicar si quiere aplicar un descuento (si es así, indica de cuánto, caso contrario, 0%). Al final el programa deberá mostrar todas las facturas con los datos del cliente, y el monto sin y con descuento, respetando el formato siguiente:

Cant. Productos: X unidades

Monto s/descuento: $XX,XX
Monto c/descuento: $XX,XX