**Resúmen Primer Parcial**

**Laboratorio II**

**Índice:**

**Clase Char: Pág 2**

**Clase String: Pág 5**

**Clase StringBuilder: Pág 9**

**Clase Array: Pág 12**

**Clase Char**

**.Compare to:**

|  |
| --- |
| Compara una instancia con otra y devuelve una indicación de sus valores relativos. Devuelve un entero. |

|  |  |
| --- | --- |
| char chA = ‘A’, chN = ‘N’;  int resultado = chA.**CompareTo**(‘4');  Console.WriteLine(resultado);  // Salida: “13”  Console.WriteLine(chA.**CompareTo**(chN));  // Output: "-13“ |  |

**.Equals:**

Compara una instancia con otra, indicando si son iguales o no. Devuelve un booleano.

|  |
| --- |
| char chA = ‘A’, chN = ‘N’, chA2 = 'A';  string str = “cadena”;  Console.WriteLine(chA.Equals('A')); //True  Console.WriteLine(chA.Equals(chA)); //True  Console.WriteLine(chA.Equals(chN)); // False |

**.GetNumericValue:**

Convierte un carácter Unicode numérico en un número de tipo double. Por ejemplo:

char ch1 = ‘1’; string str = “cadena 4”;

double numero = Char.GetNumericValue(ch1);

Console.WriteLine(numero); // Salida: “1“

Console.WriteLine(Char.GetNumericValue(str, 7)); // Salida: “4“

**.ToString:**

Convierte el valor del caracter en una cadena string equivalente. Por ejemplo:

char cadena = 'x'.ToString();

Console.WriteLine(cadena); // Salida: "x“

Console.WriteLine(Char.ToString('b')); // Salida: "b"

**.Parse:**

Convierte una cadena en el caracter equivalente. Por ejemplo:

string cadena = “s”;

char caracter = Char.Parse(cadena);

Console.WriteLine(caracter); // Salida: ‘S’

**.TryParse:**

Ídem Parse, pero devuelve un código indicando si la conversión fue exitosa o no. Por ejemplo:

char chObtenido; bool resultado;

resultado = char.TryParse("J", **out** chObtenido);

Console.WriteLine(resultado + " " + chObtenido); // Salida: “true J”

**.GetUnicodeCategory:**

Indica a qué categoría pertenece un carácter. Las posibles categorías están definidas en la enumeración UnicodeCateogry. Por ejemplo:

UnicodeCategory categoria = Char.GetUnicodeCategory('a');

Console.WriteLine(categoria); // Salida: "LowercaseLetter"

Console.WriteLine(Char.GetUnicodeCategory(‘1’)); // Salida : "DecimalDigitNumber“

***Nota***: para acceder a la enumeración *UnicodeCategory*, deberá referenciar el espacio de nombres **System.Globalization**

Para conocer si un caracter pertenece a una categoría particular, se utilizan métodos específicos, como por ejemplo:

***IsDigit***:

Para indica si se clasifica como un dígito decimal. Por ejemplo:

Console.WriteLine(Char.IsDigit(‘1’)); // Salida: "True“

***IsLetter***:

Para indicar si se clasifica como carácter alfabético. Por ejemplo:

Console.WriteLine(Char.IsLetter(',')); // Salida: "False“

***IsPunctuation***:

Para indicar si se clasifica como un signo de puntuación. Por ejemplo:

Console.WriteLine(Char.IsPunctuation('.')); // Salida: "True“

**Además: IsControl, IsHighSurrogate, IsLetterOrDigit, IsLower, IsLowSurrogate, IsNumber, IsSeparator, IsSurrogate, IsSurrogatePair, IsSymbol, IsUpper y IsWhiteSpacepara.**

**Clase String**

**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 caracter en una cadena:



char x = cadena[3]; // x = ‘n';

**Ver Word para saber como armar string!!**

**Funciones de String**

**.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: “-;;;;;;;;;;;;Contenido-”

Console.WriteLine(“-{0}-", cadena.PadRight(20, '.'));

// Salida: “-Contenido………… -”

**.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“

**.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

**.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”

**.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:



**.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

**.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

**Clase StringBuilder**

Representa una cadena de caracteres modificable.

Puede cambiarse una vez se ha creado mediante adición, sustracción, sustitución o inserción de caracteres.

StringBuilder MiStringBuilder = new StringBuilder("Hello World!”);

**Capacidad**: número máximo de caracteres que puede contener la cadena.

**Longitud**: Obtiene o establece la longitud del objeto actual.

Por ejemplo, se crea un objeto StringBuilder con la cadena "Hello", que tiene una longitud de 5, y especificar que el objeto tenga una capacidad máxima de 25.

StringBuilder MyStringBuilder = new StringBuilder("Hello World!", 25);

MyStringBuilder.Capacity = 30; //Aquí se aumenta la capacidad

La capacidad *no puede ser menor* que el valor de la propiedad Length.

**.Append:**

Permite agregar texto al final de una cadena. El espacio se asigna automáticamente según sea necesario. Por ejemplo:

StringBuilder MyStringBuilder = new StringBuilder("¡Hola Mundo!");

MyStringBuilder.Append(" Qué lindo día.");

Console.WriteLine(MyStringBuilder); // “¡Hola mundo! Qué lindo día.”

**.AppendFormat:**

Igual que Append, pero acepta cadenas de formato que contienen cero o más especificaciones de formato. Por ejemplo:

int MyInt = 25;

StringBuilder MyStringBuilder = new StringBuilder("Su total es");

MyStringBuilder.AppendFormat(“${0} ", MyInt);

Console.WriteLine(MyStringBuilder); // Salida: “Su total es $25”

**.Insert:**

Agrega una cadena en una posición específica en la StringBuilder actual.

StringBuilder MyStringBuilder = new StringBuilder("¡Hola mundo!");

MyStringBuilder.Insert(5,"hermoso ");

Console.WriteLine(MyStringBuilder); // Salida: “¡Hola hermoso mundo!”

**.Remove:**

Permite quitar cierta cantidad de caracteres comenzando en la posición indicada.

StringBuilder MyStringBuilder = new StringBuilder("¡Hola Mundo!");

MyStringBuilder.Remove(5,7);

Console.WriteLine(MyStringBuilder); // Salida: “¡Hola”

**.Replace:**

Reemplaza todas las apariciones de una cadena o caracter en el objeto StringBuilder con otra cadena o caracter.

StringBuilder MyStringBuilder = new StringBuilder("!Hola Mundo!");

MyStringBuilder.Replace('!', '?');

Console.WriteLine(MyStringBuilder); // Salida: “?Hola mundo?”

**Clase Array**

**.Sort:**

Ordena los elementos de toda una matriz unidimensional. Por ejemplo:

string[] palabras = {"el", "verde", "lindo", "queda", "salta", "ejercitación", "y", "mira"};

Array.**Sort**(palabras);

// Salida: ejercitación – el – lindo – mira – queda – salta – verde – y

**.Reverse:**

**Invierte el orden** de los elementos de una matriz Array unidimensional o de una parte de Array. Por ejemplo, al arreglo anterior se le aplica:

Array.**Reverse**(palabras);

// Salida: y – verde – salta – queda – mira – lindo – el – ejercitación

**.Sort(arregloClaves,arregloElementos):**

Ordena un par de objetos **Array** unidimensionales donde el primero de ellos contiene las claves y el segundo los elementos correspondientes. La ordenación se realiza en función del arreglo que contiene las claves. Por ejemplo:

**.IndexOf:**

Busca un objeto determinado en todo el arreglo o parte de éste, y devuelve el índice de la primera aparición del mismo. Por ejemplo:

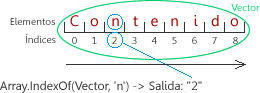
string[] claves = { "rojo", "verde", "naranja", "azul", “lila", "naranja" };

int indice = Array.IndexOf(claves, "naranja"); // 2

Array.IndexOf(claves, "naranja", 3); // 5

Array.IndexOf(claves, "naranja", 0, 3); // 2

// Array.IndexOf(arreglo, objeto, posicionInicial, cantElementosBuscar)



**.LastIndexOf:**

Devuelve el índice de la última aparición de un valor en una matriz unidimensional o en una parte de ésta. La búsqueda se realiza a partir del último elemento (o la posición indicada) hasta el inicio o lo especificado. Por ejemplo:

string[] claves = { "rojo", "verde", "naranja", "azul", “lila", "naranja" };

Array.LastIndexOf(claves, "naranja"); // Salida: 5

Array. LastIndexOf(claves, "naranja", 3); // Salida: 2

Array. LastIndexOf(claves, "naranja", 3, 3); // Salida: 2

// Array.IndexOf(arreglo, objeto, posicionInicial, cantElemHaciaAtrás)

**.CopyTo:**

Copia todos los elementos del arreglo actual en el arreglo destino especificado, empezando en el índice indicado del arreglo de destino. Por ejemplo:

Vector1.CopyTo(Vector2, 2);



