UT 3 - Componentes

INTRODUCCIÓN C#

Visual Studio 2019 - C#

Sintaxi C# :

- ✓ **Estructuras de control**
- ✓ **Funciones**
- ✓ **Arrays**
- ✓ **String**
- ✓ **StringsBuilder**
- ✓ **Excepciones**
- ✓ **Diccionary / Hash Table**

Visual Studio 2019 - C#

Sintaxi C# - UT3 - C# Estructuras de control

✓ Saltos incondicionales

- ❑ Cuando encuentre la llamada a otros métodos/funciones
- ❑ Con el uso de las palabras claves como `goto, break, continue, return y throw`

✓ Saltos condicionales

Visual Studio 2019 - C#

Sintaxi C# - UT3 - C# Estructuras de control

✓ Saltos incondicionales

- Goto

```
static void Main()
{
    int contador=0;
    REPETIR:
    Console.WriteLine ("La línea: {0}", contador);
    if (contador++ < 100)
        goto REPETIR;
}
```

Obsoleta y confusa

Visual Studio 2019 - C#

Sintaxi C# - UT3 - C# Estructuras de control

✓ Saltos condicionales

- ❑ If
- ❑ Switch
- ❑ Bucle for
- ❑ Bucle while
- ❑ Bucle do-while
- ❑ Bucle `foreach`
- ❑ Usando `break, continue, return y throw`

Visual Studio 2019 - C#

Sintaxi C# - UT3 - C# Estructuras de control

✓ Saltos condicionales

- ❑ If
- ❑ Switch
- ❑ Bucle for
- ❑ Bucle while
- ❑ Bucle do-while
- ❑ Bucle `foreach`
- ❑ Usando continue y break `break, continue, return y throw`

Diferencias?

Cuando uso...

Visual Studio 2019 - C#

Sintaxi C# - UT3 - C# Estructuras de control

✓ Saltos condicionales

- Bucle **`foreach`**

  se utiliza para hacer **iteraciones sobre elementos de una colección**, como pueden ser los enteros dentro de un arreglo de enteros.

```
int[,] arr = {{1,2},{3,4}};
foreach( int elem in arr )
{
    Console.Write( elem +", ");    //salida 1, 2, 3, 4,
}
```

Visual Studio 2019 - C#

Sintaxi C# - UT3 - C# Estructuras de control

✓ Saltos condicionales

- Usando `break, continue, return y throw`

```
for(int i = 0; i<20; i++ )
{
  if (i==5) continue;
  if (i==9) break;
  Console.Write("{0},",i);
}
//salida 0,1,2,3,4,6,7,8,
```

Visual Studio 2019 - C#

Sintaxi C# - UT3 - C# Funciones

✓ Funciones

```
<visibility> <return type> <name>(<parameters>)
{
        <function code>
}
```

```
//EJEMPLO
public void DoStuff()
{
    Console.WriteLine("I'm doing something...");
}
```

Visual Studio 2019 - C#

Sintaxi C# - UT3 - C# Funciones

✓ Funciones

```
<visibility> <return type> <name>(<parameters>)
{
        <function code>
}
```

- **<visibility>** PUBLIC, es la visibilidad, y es opcional. Si no se define ninguna, entonces la función será PRIVATE.
- **<return type>:** cualquier tipo válido en C# o VOID(que no retorna nada. Si tiene un tipo debemos tener siempre un **return**
- **<parameters>** aunq no tenga parámetros siempre -> paréntesis.

Visual Studio 2019 - C#

✓ Funciones

<visibility> → Modificadores de acceso

public: Es el modificador menos restrictivo. No existen restricciones para el acceso a los miembros o tipos que se hayan definido mediante public.

private: sólo son accesibles dentro de la clase en la que se definen. Es por tanto el modificador más restrictivo.

readonly: el campo no se puede modificar fuera de la propia declaración del campo o del contructor de la clase.

static: los miembros declarados con *static* tienen la cualidad de sólo existir una vez y no pueden coexistir múltiples instancias en memoria al mismo tiempo

https://docs.microsoft.com/es-es/dotnet/csharp/language-reference/keywords/access-modifiers

Visual Studio 2019 - C#

✓ Funciones

`<visibility>`

Accesibilidad declarada	Significado
public	El acceso no está restringido.
protected	El acceso está limitado a la clase contenedora o a los tipos derivados de la clase contenedora.
internal	El acceso está limitado al ensamblado actual.
protected internal	El acceso está limitado al ensamblado actual o a los tipos derivados de la clase contenedora.
private	El acceso está limitado al tipo contenedor.
private protected	El acceso está limitado a la clase contenedora o a los tipos derivados de la clase contenedora que hay en el ensamblado actual. Disponible desde la versión C# 7.2.

Visual Studio 2019 - C#

✓ Arrays

Los arrays funcionan como colecciones de elementos.

```
static void Main(string[] args)
 {
    int[] numbers = { 4, 3, 8, 0, 5 };

    Array.Sort(numbers);
    foreach(int i in numbers)
        Console.WriteLine(i);
    Console.WriteLine(numbers.length );
 }
```

Visual Studio 2019 - C#

✓ Arrays - Métodos

Más **métodos** de esta clase:

- **Array.AsReadOnly**
- **Array.BinarySearch**
- **Array.Clear**
- **Array.ConstrainedCopy**
- **Array.ConvertAll**
- **Array.Copy**
- **Array.CreateInstance**
- **Array.Exists Array.Find**
- **Array.FindIndex**
- **Array.ForEach Array.IndexOf**
- **Array.LastIndexOf**
- **Array.Resize**
- **Array.Reverse**
- **Array.Sort**
- **Array.TrueForAll**

Visual Studio 2019 - C#

✓ Arrays - Propiedades

- Length
- Count IsFixedSize,
- IsReadOnly
- IsSynchronized

Info de arrays en C#:

- https://www3.gobiernodecanarias.org/medusa/eforma/campus/pluginfile.php/6077191/mod_resource/content/15/capitulo6.pdf
- https://www.dotnetperls.com/array
- https://docs.microsoft.com/es-es/dotnet/api/system.array?view=net-5.0

Visual Studio 2019 - C#

✓ Strings

- In .NET, the text is stored as a sequential read-only **collection of Char objects**.
- There is **no null-terminating character** at the end of a C# string; therefore a C# string can contain any number of embedded null characters ('\0').

System.String vs string

The string class is equivalent to the System.String in C# .
The string class inherits all the properties and methods of the System.String class.

Visual Studio 2019 - C#

✓ Strings

- In .NET, the text is stored as a sequential read-only **collection of Char objects**.
- There is **no null-terminating character** at the end of a C# string; therefore a C# string can contain any number of embedded null characters ('\0').
- Strings in .NET are **immutable**. Once a string is created, it cannot be modified. Every time you assign a new value to an existing string object, a new object is being created and old object is being released by the CLR. If you are doing lot of string operations, the StringBuilder class is recommended.

✓ Strings

- In .NET, the text is stored as a sequential read-only **collection of Char objects**.
- There is **no null-terminating character** at the end of a C# string; therefore a C# string can contain any number of embedded null characters ('\0').
- Strings in .NET are immutable. Once a string is created, it cannot be modified. Every time you assign a new value to an existing string object, a new object is being created and old object is being released by the CLR. If you are doing lot of string operations, the StringBuilder class is recommended.

System.String vs string

The string class is equivalent to the System.String in C# .

Visual Studio 2019 - C#

✓ Strings - Create

```
char[] chars = {'M','a','h','e','s','h'};
string name = new string(chars);
Console.WriteLine(name);
```

```
System.String firstName = "Mahesh";
```

System.String vs string

The string class is equivalent to the System.String in C# .
The string class inherits all the properties and methods of the System.String class.

Visual Studio 2019 - C#

✓ C# STRING FUNCTION

String Functions	Definitions
Clone()	Make clone of string.
CompareTo()	Compare two strings and returns integer value as output. It returns 0 for true and 1 for false.
Contains()	The C# Contains method checks whether specified character or string is exists or not in the string value.
EndsWith()	This EndsWith Method checks whether specified character is the last character of string or not.
Equals()	The Equals Method in C# compares two string and returns Boolean value as output.

String Functions	Definitions
Equals()	The Equals Method in C# compares two string and returns Boolean value as output.
GetHashCode()	This method returns HashValue of specified string.
GetType()	It returns the System.Type of current instance.
GetTypeCode()	It returns the Stystem.TypeCode for class System.String.
IndexOf()	Returns the index position of first occurrence of specified character.
ToLower()	Converts String into lower case based on rules of the current culture.
ToUpper()	Converts String into Upper case based on rules of the current culture.
Insert()	Insert the string or character in the string at the specified position

String Functions	Definitions
IsNormalized()	This method checks whether this string is in Unicode normalization form C.
LastIndexOf()	Returns the index position of last occurrence of specified character.
Length	It is a string property that returns length of string.
Remove()	This method deletes all the characters from beginning to specified index position.
Replace()	This method replaces the character.
Split()	This method splits the string based on specified value.
StartsWith()	It checks whether the first character of string is same as specified character.
Substring()	This method returns substring.
ToCharArray()	Converts string into char array.
Trim()	It removes extra whitespaces from beginning and ending of string.

Visual Studio 2019 - C#

✓ Strings - null -- Empty

A null string is a string variable that has not been initialized yet and has a null value. If you try to call any methods or properties of a null string, you will get an exception

```
//null
string nullStr = null;

//Empty-> object that contains zero characte
string empStr = string.Empty;
string empStr2 = "";
```

Visual Studio 2019 - C#

✓ Strings - Recorrer

```
//recorrer string
string nameString = "Mahesh Chand";
for (int counter = 0; counter <= nameString.Length - 1; counter++)
 Console.WriteLine(nameString[counter]);
```

✓ Copiar

```
string autName = firstName + " " + lastName;
string copyAuthor = string.Copy(autName);
```

```
char[] copiedStr = {'n','e','e','l'};
autName.CopyTo(2, copiedString, 2, 6);
```

Visual Studio 2019 - C#

✓ Strings - Rellenar

```
string str = "forty-two";
char pad = '.';

str.PadLeft(15, pad);     //......forty-two
str.PadRigth(15, pad);    //forty-two......
```

MAS SOBRE Strings:

- https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/strings/

Visual Studio 2019 - C#

✓ StringsBuilder

Representa una cadena de caracteres mutable. Esta clase no puede heredarse.

```
// Create a StringBuilder that expects to hold 50 characters.
// Initialize the StringBuilder with "ABC".
StringBuilder sb = new StringBuilder("ABC", 50);

// Append three characters (D, E, and F) to the end of the StringBuilder.
sb.Append(new char[] { 'D', 'E', 'F' });

// Append a format string to the end of the StringBuilder.
```

Visual Studio 2019 - C#

✓ ArrayList

```
ArrayList firstList = new ArrayList ();
firstList.Add (5);
firstList.Add (7);

ArrayList secondList = new ArrayList ();
secondList.Add (10);
secondList.Add (13);

//Añadir secondList dentro de firstList
firstList.AddRange (secondList);

//Recorrer la lista
foreach (int i en la firstList)
{
    Console.WriteLine (i);
}
```

Visual Studio 2019 - C#

✓ **Excepciones**

- la validación de entrada del usuario
- la comprobación de los objetos nulos
- la verificación de otros valores...

Visual Studio 2019 - C#

✓ **Excepciones** *muy parecido a Java*

```
Try
{
    Console.WriteLine("El primero:");
    Int primero = int.Parse(Console.ReadLine());
}
catch (Exception ex)
{
   Console.WriteLine("No es un número: " + ex.Message);
}
```

Visual Studio 2019 - C#

✓ Excepciones *muy parecido a Java*

También podemos usar el finally para añadir texto después de la excepción

```
Try
{
    Console.WriteLine("El primero:");
    Int primero = int.Parse(Console.ReadLine());
}
catch (Exception ex)
{
   Console.WriteLine("No es un número: " + ex.Message);
}
finally
{
 //añadimos aqui lo que se ejecutará al final aunque haya saltado la excepción
}
```

Visual Studio 2019 - C#

✓ Excepciones *muy parecido a Java*

Por ejemplo con: **System.IO.File.OpenRead (),** podría pasar alguna de las siguientes excepciones:

- ArgumentException
- ArgumentNullException
- PathTooLongException
- DirectoryNotFoundException
- UnauthorizedAccessException
- FileNotFoundException
- NotSupportedException
- SecurityException

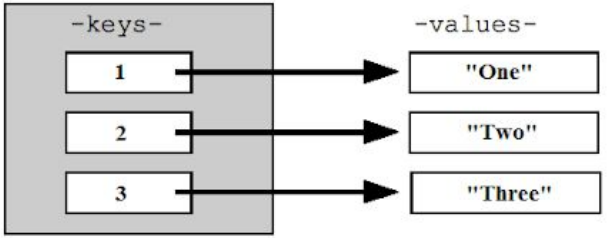
Visual Studio 2019 - C#

✓ Excepciones *muy parecido a Java*

Por ejemplo con: **System.IO.File.OpenRead (),** podría pasar alguna de las siguientes excepciones:

- ArgumentException
- ArgumentNullException
- PathTooLongException
- DirectoryNotFoundException
- UnauthorizedAccessException
- FileNotFoundException
- NotSupportedException
- SecurityException

Visual Studio 2019 - C:



✓ Diccionary

```
Dictionary<string, string> openWith =new Dictionary<string, string>();

//Add some elements to the dictionary.There are no duplicate keys
openWith.Add("txt", "notepad.exe");
openWith.Add("bmp", "paint.exe");
openWith.Add("dib", "paint.exe");
Console.WriteLine("For key = rtf, value = {0}.",openWith["rtf"]);
// If a key does not exist, adds a new key/value pair.
openWith["doc"] = "winword.exe";
openWith.remove("doc");
foreach( KeyValuePair<string, string> kvp in openWith )
{
    Console.WriteLine("Key= {0}, Val= {1}",kvp.Key, kvp.Value);
}
```

https://docs.microsoft.com/es-es/dotnet/api/system.collections.generic.dictionary-2?view=net-5.0

Visual Studio 2019 - C#

✓ Hash Table

```
Hashtable numberNames = new Hashtable();
numberNames.Add(1,"One"); //adding a key/value using the Add()
numberNames.Add(2,"Two");
numberNames.Add(3,"Three");
numberNames.Remove(3);


//The following throws run-time exception: key already added.
//numberNames.Add(3, "Three");

foreach(DictionaryEntry de in numberNames)
    Console.WriteLine("Key: {0}, Value: {1}", de.Key, de.Value);
```