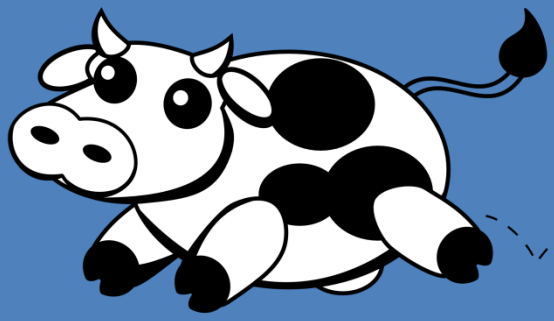
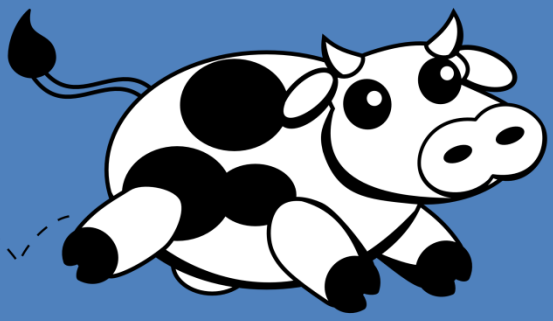
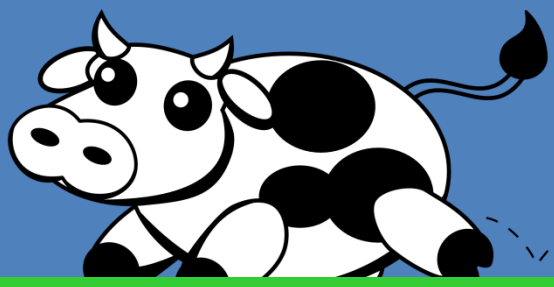






# Programación de Aplicaciones con C#

Para 3° año de EMT de Informática de UTU/CETP

**Prof. Tco. Luis Sebastián de los Angeles**

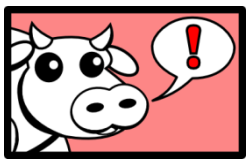
# Programación de Aplicaciones con C#

## 1. Introducción a la Programación de Aplicaciones

**Bienvenidos** al curso de Programación de 3° año del Bachillerato de informática. En este curso el enfoque se centra en el desarrollo de **aplicaciones completas**, y no tanto en el estudio de estructuras algorítmicas y de programación como fue el caso de años anteriores.

Esto implica que se espera que el alumno **maneje de forma fluida los conceptos básicos de programación** estudiados en años anteriores. Sin perjuicio de ello, siempre que toquemos un tema o concepto de años anteriores por primera vez en este texto, se hará una introducción y **se recordará su definición general**.

Se espera, sin embargo, que el alumno **tenga la iniciativa de buscar** en los materiales de años anteriores y en otras fuentes confiables **siempre que tenga dudas**.



**IMPORTANTE:** Un desarrollador siempre debe ir a las **fuentes de la documentación oficial** y consultarlas para acceder a la información más actualizada, evitando, de esta manera, posibles errores de seguridad al confiar en información de su memoria, la cual puede basarse en datos desactualizados.

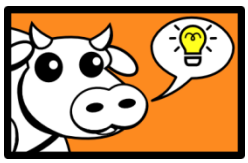
### Áreas de Estudio del Curso

Este curso se desarrollará estudiando tres grandes núcleos conceptuales:

1. **Programación en Capas:** Un paradigma de programación que establecerá la forma **en que organizaremos los componentes** de nuestras aplicaciones de acuerdo a su **comportamiento**, su relación con las **acciones del usuario**, o su **funcionalidad general**.

2. **Programación Orientada a Eventos:** Un paradigma de programación que define la forma en que plantearemos el **flujo de los algoritmos** de nuestra aplicación con respecto a su **entorno informático** (comunicación de red, bases de datos, sistemas de archivos, etc.) y con **respecto al usuario** (detección de acciones concretas como ser presionar un botón, abrir o cerrar ventanas, ingresar datos, acceder a recursos, etc.).

3. **Lenguaje de Programación C#:** El **lenguaje de programación** que utilizaremos en este curso.



**Resumiendo:** En este curso estudiaremos cómo **desarrollar aplicaciones** que permitan una **interacción fluida con el usuario** y otros **componentes informáticos** (bases de datos, archivos, etc.) siguiendo los patrones establecidos por la **programación orientada a eventos**, organizando nuestro código de acuerdo a lo que establece la **programación en capas**, utilizando el **lenguaje de programación C#.**

## 2. El Lenguaje de Programación C#

Comencemos estudiando un poco el lenguaje con el que trabajaremos durante todo el año.

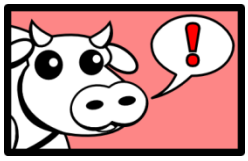
El lenguaje de programación C# o C *Sharp* (en este libro se usará C# para referirse al lenguaje), es un lenguaje de programación de **propósito general y multiparadigma** desarrollado por **Microsoft** y cuya primera versión fue publicada en el año 2000. Su última versión estable al momento de la redacción de este texto es la versión **8.0**[1], publicada el 23 de septiembre de 2019.

### Entornos de Desarrollo para C#

Como es de esperar en cualquier lenguaje de programación, es posible desarrollar aplicaciones en C# utilizando solamente **un editor de texto plano** y el conjunto de **herramientas del kit de desarrollo** del lenguaje. Sin embargo, esto resulta, en gran medida, incómodo y hasta contraproducente hoy en día. Lo recomendado para desarrollar aplicaciones en C# es el uso de un **entorno de desarrollo integrado** (o ***IDE***, por sus siglas en inglés) que facilite nuestra tarea. En los ejemplos de este texto utilizaremos dos IDE's diferentes, dependiendo del sistema operativo en el que estemos trabajando:

- **Visual Studio Community**, para Microsoft Windows.
  Se obtiene de la página oficial (https://visualstudio.microsoft.com/es/downloads/) bajo la opción "**Comunidad**". El instalador nos dará una serie de opciones para instalar, las recomendadas para este curso son "**Desarrollo de escritorio de .NET**" y "**Desarrollo de la plataforma universal de Windows**".

- **MonoDevelop**, para GNU/Linux o Mac OS.
  Se obtiene de la página oficial (https://www.monodevelop.com/download/), donde, además se listan los requerimientos y el procedimiento de instalación para diferentes distribuciones.



NOTA IMPORTANTE:
En cualquiera de los dos casos, la instalación involucra la descarga e instalación de una cantidad considerable de paquetes de software, por lo que es recomendable realizar la instalación contando con un buen ancho de banda (y conexión eléctrica, de estar usando una notebook).

### Características del Lenguaje

Sintácticamente hablando, pertenece a la familia de lenguajes que descienden de **C**, por lo que una primera vista a una porción de código escrito en **C#** nos mostrará una serie de elementos comunes a otros lenguajes de esta familia, como son **Java** o **PHP.**

1. Todas las sentencias terminan con **punto y coma.**
```
public static Image img;
```

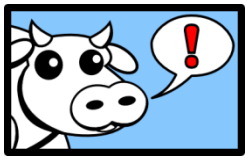
2. Los bloques de código se delimitan **mediante llaves.**
```
public class Ejemplo{
}
```

3. Los arrays, listas o vectores hacen uso de **corchetes** (también conocidos como paréntesis rectos) en su sintaxis.
```
int[] arrayNumerico = new int[10];
arrayNumerico[0] = 25;
```

[1] Según información obtenida de https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-8

En general podemos hablar de un lenguaje que permite desarrollar código de una forma muy similar a la experiencia que se tiene al trabajar con Java.

Las similitudes son muchas:



**Semejanzas entre C# y JAVA:**

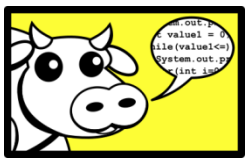
- Fuertemente tipados.
- Orientados a objetos basados en clases.
- Basados en compilación JIT (parte del código puede compilarse justo antes de la ejecución).
- Pasaje automático entre tipos de datos primitivos y referenciados.
- Etc.

Pero también tiene particularidades que lo diferencian de otros lenguajes de esta familia, como veremos en este texto.

## 1. Declaración de variables

Como ya se dijo, C# es un lenguaje **fuertemente tipado**, lo que implica que **variables, objetos y métodos deben tener un tipo declarado** y ese tipo **no cambia durante la ejecución.**

Las variables se declaran indicando su **tipo de datos**, su **nombre** y su **valor inicial**. En general también es recomendable declarar alguno de los **modificadores de acceso**, los cuales estudiaremos en profundidad más adelante.



**Declaración de variables:**

```
acceso tipo nombre = valorInicial;
```

**Ejemplos:**

```
public int numero = 0;
private string texto = "";
```

**Tipos de Datos Comunes:**

| Tipo | Variedad | Características | Tamaño en memoria |
|---|---|---|---|
| `bool` | Lógico | `true` y `false` son los únicos valores admitidos | 4 bytes |
| `sbyte` | Numérico | Enteros de -128 a 127 | 8 bits |
| `byte` | Numérico | Enteros positivos (sin signo) de 0 a 255 | 8 bits |
| `short` | Numérico | Enteros de -32.768 a 32.767 | 16 bits |
| `ushort` | Numérico | Enteros positivos (sin signo) de 0 a 65.535 | 16 bits |
| `int` | Numérico | Enteros de -2.147.483.648 a 2.147.483.647 | 32 bits |
| `uint` | Numérico | Enteros positivos (sin signo) de 0 a 4.294.967.295 | 32 bits |
| `long` | Numérico | Enteros de -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 | 64 bits |

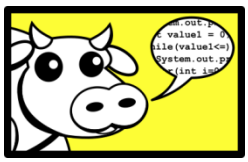
| | | | |
|---|---|---|---|
| **ulong** | Numérico | Enteros positivos (sin signo) de 0 a 18.446.744.073.709.551.615 | 64 bits |
| **float** | Numérico | Números decimales de ±1.5 x $10^{-45}$ a ±3.4 x $10^{38}$ con 6 a 9 dígitos después de la coma | 4 bytes |
| **double** | Numérico | Números decimales de ±5.0 × $10^{-324}$ a ±1.7 × $10^{308}$ con 15 o 16 dígitos después de la coma | 8 bytes |
| **decimal** | Numérico | Números decimales de ±1.0 x $10^{-28}$ a ±7.9228 x $10^{28}$ con 28 o 29 dígitos después de la coma | 8 bytes |
| **char** | Texto | Un (1) carácter Unicode | Variable |
| **string** | Texto | Secuencias de caracteres | Variable |
| **void** | - | Usado en funciones. Determina una función que **no devuelve** un valor u objeto tras su ejecución. | - |

## 2. Estructuras de control

Las estructuras de control en C# se utilizan **exactamente igual que en Java**. También existe una estructura de control extra llamada **`foreach`** que veremos más en profundidad cuando estudiemos el **manejo de colecciones**.

### a) IF

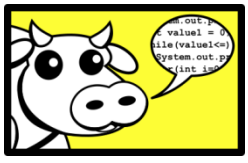
La estructura IF también tiene, en C#, una forma "continua" en la que añadimos una nueva estructura **`if`** inmediatamente después del **`else`**, la cual estudiaremos en profundidad más adelante.



**Declaración de IF:**

```
if (condición){
 }
else{
 }
```

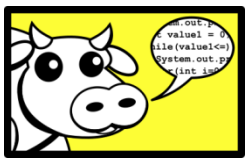
### b) WHILE



**Declaración de WHILE:**

```
while (condición){
 }
```

### c) FOR



**Declaración de FOR:**

```
for (int contador = 0; contador < valorTope; contador++){
}
```

## 3. Métodos o Funciones

Tal y cómo vimos en Java, la declaración de métodos o funciones es fundamental para el desarrollo de la aplicación, ya que es en esas funciones que estableceremos los algoritmos que formarán las funcionalidades de nuestra aplicación. Igual que en Java, existe el tipo de datos **`void`**, que usaremos cuando nuestro método o función **no requiera** la **devolución de una respuesta** mediante el comando **`return`**.

Estudiaremos la declaración de métodos en profundidad en los próximos capítulos.

**Declaración de Métodos:**

```
acceso tipoDelResultado nombre(tipoDelParametro parámetro){

}
```

**Ejemplos:**

```
public int sumar(int numeroA, int numeroB){
      int numeroResultado = 0;
      numeroResultado = numeroA + numeroB;
      return numeroResultado;
}

private void mostrar(string mensaje){
      Console.WriteLine(mensaje);
}
```