Ejemplo 1.

```
class HolaMundo
 {
    static void Main()
    {
       string var="Mundo";
       System.Console.WriteLine ("Hola {0}!", var);
    }
 }
```

Ejemplo 2.

```
namespace Programa1
 {
  class HolaMundo
  {
    static void Main()
    {
      string var="Mundo";
      System.Console.WriteLine ("Hola {0}!", var);
      int num1 = 1;
      int num2 = 3;
      int resultado = Suma (num1, num2);
      System.Console.WriteLine ("{0}+{1} = {2}", num1, num2, resultado);
    }

    static int Suma(int valor1, int valor2)
    {
      return (valor1+valor2);
    }
  }
 }
```

Ejemplo 3.

```
class Enteros{
  public static void Main()
  {
     int minuto = 60;    //segundos por minuto
     int hora = minuto*60;
     int dia = hora*24;
     long anio = dia*365;
     Console.WriteLine("Segundos en un dia: {0}", dia);
     Console.WriteLine("Segundos en un año: {0}", anio);
  }
}
```

Ejemplo 4.

```
class Flotante{
   public static void Main()
   {
      int a = 2;
      double log2 = Math.Log(a);
      double raiz2 = Math.Sqrt(a);
      Console.WriteLine("El logaritmo de dos es {0}", log2 );
      Console.WriteLine("La raiz de dos es {0}", raiz2 );
   }
 }
```

Crear nuevos métodos o funciones (ejemplo 1)
Para declarar un método o función utilizamos el siguiente formato:
[entorno] *tipo_a_retornar* **Nombre_de_la_Función** ([tipo Argumento1, tipo Argumento2,...])
{
//lineas de código
}
las palabras dentro de corchetes [] son partes opcionales de acuerdo con el ejemplo, la función Main() cumple con el formato establecido:
static *void* Main()
{
//lineas de código
}

- **entorno:** static
- **tipo_a_retornar:** void
- **Nombre_de_la_Función:** Main
- **Argumentos:** ninguno

En nuestro ejemplo la función Main() tiene como entorno la palabra clave **static** y como tipo_a_retornar la palabra clave **void**. Cuando la función no retorna ningún tipo, utilizamos la palabra **void**. Podemos añadir a nuestro componente o clase un ilimitado número de funciones.

Aplicaciones de consola
Las aplicaciones de consola no poseen una interfaz gráfica, no tienen botones o ventanas, poseen una interfaz basada simplemente en texto.
Para poder hacer uso de funciones **estáticas** que se encuentran en otros componentes, en C# como en la mayoría de lenguajes de programación orientados a objetos, debemos especificar el nombre del componente en primer lugar seguido por un punto y a continuación en el nombre de la función. Es por esto que utilizamos la frase Console.WriteLine. El lenguaje C# esta orientado con el paradigma de objetos y hereda muchos elementos de C++.

Namespaces (ejemplo 2)
Los espacios de nombres (namespaces) se crearon principalmente para dar más organización a los componentes. La plataforma .NET tiene incorporados muchísimos componentes y sería una tarea imposible tratar de memorizar todos los nombres de ellos para no repetirlos.
Supongamos que deseamos crear varios componentes para una institución educativa que se compone de educación primaria y educación secundaria. ¿Cómo podríamos crear un componente que se llame Presidente si existen 2 presidentes que tienen funciones distintas uno para la sección primaria y otro para la sección secundaria? En este caso podríamos aplicar un espacio de nombres para poder crear los componentes Presidente que cumplen distintas funciones. Podríamos crear el espacio de nombres Primaria y dentro de éste el componente Presidente. De igual forma el espacio de nombres Secundaria y dentro de éste el componente Presidente cada uno componentes podrá tener definiciones distintas. Para acceder a las funciones de los componentes Presidente podríamos usar:

Primaria.Presidente.nombre_de_la_función(); y Secundaria.Presidente.otra_función();

Cada una de las dos definiciones de Presidente son independientes, no tienen relación entre sí ya que pertenecen a dos espacios de nombre distintos.
De ésta forma podremos crear componentes y funciones con el nombre que deseemos siempre y cuando especifiquemos a qué espacio de nombres pertenecen.

Algo importante que debemos notar es que los espacios de nombres pueden tener sub-espacios de nombres y estos a su vez sub-espacios de nombres. El objetivo de esto, como lo hemos dicho, es mantener una organización de los componentes. Los espacios de nombres, componentes y métodos se accederán de la misma forma como lo hemos visto a través de un punto.

La palabra clave *using*
En ciertos proyectos tendremos que usar cierto espacio de nombres muchas veces. Supongamos que estamos implementando un programa de consola y tenemos que usar el componente Console repetidamente. Una forma de ahorrarnos escribir System varias veces es especificar el espacio de nombres que vamos a usar al inicio de nuestro programa con la palabra clave using. Si añadimos la línea de código using System; al inicio de nuestro programa, podemos llamar al componente Console sin escribir System al inicio.
Algo importante para tener en cuenta es que la palabra clave using no puede ser utilizada para ahorrarse el escribir el nombre de la clase. Por ejemplo la línea de código using System.Console es inválida y producirá errores de compilación.

Caracteres sensibles
C# como todos los lenguajes de programación derivados de C hace diferencia entre caracteres en mayúscula y caracteres en minúscula. Esto quiere decir que las palabras Using y using son distintas y por lo tanto no cumplen la misma función. Debido a esto debemos tener mucho cuidado cuando escribimos nuestros programas.