

Módulo 1

Conceptos y herramientas básicas

INAEM – Cursos de formación



Índice de contenidos

1. Programar: Escribir + Traducir + Ejecutar
 1. Escribir
 2. Traducir
 3. Ejecutar
2. Arquitectura .NET Framework
3. IDE, SDK y Framework
4. Visual Studio 2019 Community Edition
5. Primer programa: Hello World!

01

Programar: Escribir+Traducir+Ejecutar

¿En qué consiste la programación?

- Usar una herramienta para ESCRIBIR instrucciones (**Código**) en un lenguaje de programación determinado que resulta fácil de entender para los seres humanos
- Usar otra herramienta para TRADUCIR ese código a las mismas instrucciones, pero escritas en otro lenguaje que entiende el procesador del ordenador
- Usar otra herramienta para EJECUTAR en el procesador del ordenador esas instrucciones generadas tras la traducción

1.1 - Escribir

- Cada lenguaje de programación tiene:
 - **Normas** que definen cómo se escriben correctamente las distintas instrucciones
 - **Vocabulario** disponible que delimita lo que se puede escribir (y por tanto ejecutar) con él

1.1 - Escribir

- 3 tipos de normas
 - Normas **léxicas**: cómo escribir palabras
 - Normas **sintácticas**: cómo escribir frases (en programación: instrucciones)
 - Normas **semánticas**: cómo se relaciona cada instrucción con lo que pasará durante la ejecución

1.1 - Escribir

- El vocabulario de un lenguaje
 - Se debe **CONOCER** para poder usarlo
 - Se puede **AMPLIAR** si lo que ya hay hecho no es suficiente

1.1 - Escribir

- Si se escribe mal la más mínima parte de un código de programa → ¡el programa entero dará error durante la traducción y no se podrá ejecutar!
- Por ello, en programación es vital el **control de errores** de código
- La herramienta que permite escribir código (**Editor**), a veces va acompañada de otra herramienta (**Corrector**) que va analizando lo escrito en busca de errores léxicos (palabras mal escritas) y errores sintácticos (instrucciones mal escritas), avisando a tiempo real, si encuentra alguno, de dónde está y por qué lo considera un error

1.2 - Traducir

- La herramienta que traduce el código (**Compilador**) REVISA todo ese código en busca de errores léxicos (palabras mal escritas) y errores sintácticos (instrucciones mal escritas)
 - si hay errores → avisa de dónde están y por qué se consideran errores
 - si no hay errores → se hace la traducción y se generan los archivos correspondientes, escritos en el lenguaje del procesador (archivos ejecutables)

1.3 - Ejecutar

- Además de los errores léxicos y sintácticos, que se pueden detectar durante la edición o la compilación, también pueden existir **errores semánticos**, que solo se pueden detectar durante la ejecución: en el momento en que se ejecuta una instrucción con algún valor no válido para esa instrucción
- Para clasificar ambos tipos de errores, a veces éstos reciben los siguientes nombres:
 - Léxicos/Sintácticos: errores de compilación o errores estáticos
 - Semánticos: errores de ejecución o errores dinámicos

1.3 - Ejecutar

- Para cualquier instrucción escrita en el lenguaje del procesador, la herramienta que permite su ejecución es el **Sistema Operativo**
 - Si una instrucción genera un error de ejecución → El procesador da como resultado una **excepción**
- Si una excepción **no está controlada** por el código → El Sistema Operativo genera un mensaje informando de la excepción devuelta por el procesador, y termina el programa
- Algunos lenguajes permiten escribir código para **controlar una excepción**
 - Controlar una excepción permite que el programa se siga ejecutando a pesar de haber tenido un error de ejecución

1.3 - Ejecutar

- En los lenguajes que no permiten controlar excepciones, la herramienta encargada de mandar cada instrucción compilada al procesador la manda directamente, sin revisar.
 - En estos casos, cualquier error de ejecución que produzca una excepción hará que termine el programa

1.3 - Ejecutar

- En los lenguajes que permiten controlar excepciones, la herramienta encargada de mandar cada instrucción compilada al procesador, primero la revisa, y solo la manda al procesador si comprueba que no va a dar error de ejecución.
- Si comprueba que va a dar error de ejecución, avisa al programa de ello, indicando la excepción que se habría producido.
- Si el programa tiene escrito un código que controla esa excepción para esa instrucción, en vez de ejecutar esa instrucción, se ejecutarán otras en su lugar y el programa seguirá adelante.

02

Arquitectura .NET Framework

2 - Características de .NET Framework

- .NET Framework es una arquitectura de programación de Microsoft con dos características principales:
 - Permite crear programas compuestos de trozos de código escritos en distintos lenguajes de programación (C#, F#, Visual Basic .NET, etc...), como si fueran un puzzle.
 - En cualquiera de los lenguajes de programación que se pueden usar para ejecutar programas con esta arquitectura, se pueden controlar excepciones, por lo que si se escriben adecuadamente, los programas ejecutables con .NET Framework nunca tendrán errores de ejecución.

2 - Características de .NET Framework

- En .NET Framework, la traducción (Compilación) del código se hace a otro lenguaje de programación distinto del lenguaje del procesador, llamado **Lenguaje Intermedio (IL)**, que es común a todos los lenguajes con los que se pueden crear trozos del programa.
- Los archivos con código en IL generados tras la compilación tienen extensión .dll o .exe
- Existe una herramienta (**Intérprete**), que procesa UNA A UNA las instrucciones en IL que se han generado en la compilación

2 - Características de .NET Framework

- Cada instrucción en Lenguaje Intermedio pasa por el siguiente proceso:
 - Una herramienta (**Debugger**) revisa la instrucción en busca de algún error que, en caso de que el procesador la ejecutase, haría que devolviese una excepción
 - **Si no encuentra errores**, es decir, si está seguro de que no va a fallar:
 - **Se traduce** de Lenguaje Intermedio **al lenguaje del procesador**
 - El Sistema Operativo **manda al procesador que ejecute la instrucción** traducida
 - **Si encuentra algún error** en la instrucción, se **genera un mensaje que avisa de dónde está** ubicada, dentro del código del programa que hemos escrito, **y por qué se considera un error**

2 - En resumen...

- Un programa se escribe, se traduce y se ejecuta
- Pueden darse errores de compilación y errores de ejecución
- Con .NET Framework, SIEMPRE que haya un error, sea de compilación o de ejecución (gracias al Debugger), se nos va a informar de dónde está ese error y por qué es un error
 - Esto facilita MUCHO la creación y corrección (el desarrollo) de programas con .NET Framework
- Con .NET Framework, mediante control de excepciones, se puede evitar que el programa termine aunque se produzca algún error de ejecución

03

IDE, SDK y Framework

3 - IDE

- Integrated Development Environment (Entorno de Desarrollo Integrado)
 - Es un programa, habitualmente una **aplicación de escritorio**
 - Dispone en su interior de varias o todas las herramientas mencionadas hasta ahora
 - **Editor, Corrector, Compilador, Intérprete y Debugger**
 - **Dispone de otras muchas herramientas** para facilitar todo lo posible la escritura de código en un lenguaje de programación determinado
 - sugerencias de autocompletado, buscadores, referencias a invocadores, etc.

3 - SDK

- Software Development Kit (Kit de Desarrollo de Software)
 - Es un IDE, que además de permitir escribir, compilar y ejecutar programas en un lenguaje determinado, **incluye muchas bibliotecas con funciones ya hechas** que se pueden invocar desde los programas que escribamos en ese lenguaje
 - Además de las bibliotecas incluidas por defecto al instalar el SDK, existen muchas otras bibliotecas para cada lenguaje, que no se instalan por defecto pero están accesibles a través de Internet
 - Si el SDK la incluye, la herramienta que permite gestionar la instalación de esas bibliotecas externas se llama **Gestor de Paquetes**

3 - Framework

- Marco de trabajo (Frame= marco o contenedor; work= trabajo)
- Es básicamente un SDK donde el lenguaje con el que se programa cumple con determinados principios de la Programación Orientada a Objetos
 - Existe un conjunto de bibliotecas con funciones básicas, que no se pueden modificar
 - Como el lenguaje es Orientado a Objetos, existe la posibilidad de EXTENDER o personalizar esas funciones básicas si es necesario
 - Aparte de eso, se pueden crear nuevas funciones que no tengan que ver con ninguna de las funciones existentes

04

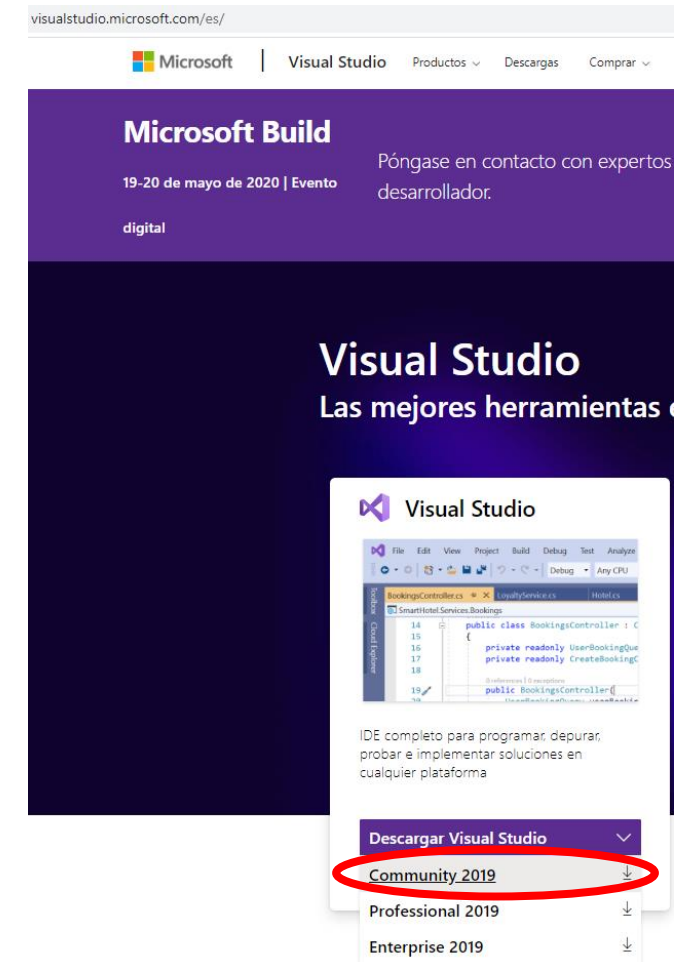
Visual Studio 2019 Community Edition

4 - Visual Studio 2019

- Es un IDE de Microsoft que permite seleccionar entre varios lenguajes para programar
- Varios de esos lenguajes incluyen bibliotecas adicionales a las mínimas necesarias para editar, compilar y ejecutar, por lo que al seleccionar uno de esos lenguajes, el IDE pasa a ser un SDK
- El lenguaje que vamos a usar en el curso, C#, es uno de esos lenguajes, y además es un lenguaje Orientado a Objetos, por lo que para nosotros Visual Studio se comportará como un Framework
















4 - Visual Studio 2019

- Instalación desde: <https://visualstudio.microsoft.com/es/>



4 - Visual Studio 2019

- Paquetes disponibles:

Cargas de trabajo	Componentes individuales	Paquetes de idioma	Ubicaciones de instalación	
Web y nube (4)				
 Desarrollo de ASP.NET y web Compila aplicaciones web con ASP.NET Core, ASP.NET, HTML/JavaScript y contenedores, e incluye compatibilidad...	<input type="checkbox"/>	 Desarrollo de Azure Proyectos, herramientas y SDK de Azure para desarrollar aplicaciones en la nube y crear recursos mediante .NET...	<input type="checkbox"/>	
 Desarrollo de Node.js Compile aplicaciones de red escalables con Node.js, un entorno de ejecución JavaScript controlado por eventos...	<input type="checkbox"/>		 Desarrollo de Python Edición, depuración, desarrollo interactivo y control de código fuente de Python.	<input type="checkbox"/>
Móviles y de escritorio (5)				
 Desarrollo de escritorio de .NET Compila WPF, Windows Forms y aplicaciones de consola mediante C#, Visual Basic y F# con .NET Core y .NET...	<input type="checkbox"/>	 Desarrollo para el escritorio con C++ Cree aplicaciones modernas de C++ para Windows con las herramientas que prefiera, como MSVC, Clang, CMake o...	<input type="checkbox"/>	
 Desarrollo para dispositivos móviles con .NET Compile aplicaciones multiplataforma para iOS, Android o Windows con Xamarin.	<input type="checkbox"/>	 Desarrollo móvil con C++ Compile aplicaciones multiplataforma para iOS, Android o Windows con C++.	<input type="checkbox"/>	
Juegos (2)				
 Desarrollo de juego con Unity Crear juegos 2D y 3D con Unity, un entorno de desarrollo eficaz de varias plataformas.	<input type="checkbox"/>	 Desarrollo de juegos con C++ Utilizar toda la potencia de C++ para crear juegos profesionales con tecnología DirectX, Unreal o Cocos2d.	<input type="checkbox"/>	
Otros conjuntos de herramientas (6)				
 Almacenamiento y procesamiento de datos Conecte, desarrolle y pruebe soluciones de datos con SQL Server, Azure Data Lake o Hadoop.	<input type="checkbox"/>	 Aplicaciones de ciencia de datos y de análisis Lenguajes y herramientas para crear aplicaciones de ciencia de datos, como Python y F#.	<input type="checkbox"/>	
 Desarrollo de Office y SharePoint Cree complementos de Office y SharePoint, soluciones de SharePoint y complementos VSTO mediante C#, VB y...	<input type="checkbox"/>	 Desarrollo de Linux con C++ Crear y depurar aplicaciones que se ejecutan en un entorno Linux.	<input type="checkbox"/>	
		 Desarrollo multiplataforma de .NET Core Compila aplicaciones multiplataforma con .NET Core, ASP.NET Core, HTML/JavaScript y Containers, además de...	<input type="checkbox"/>	

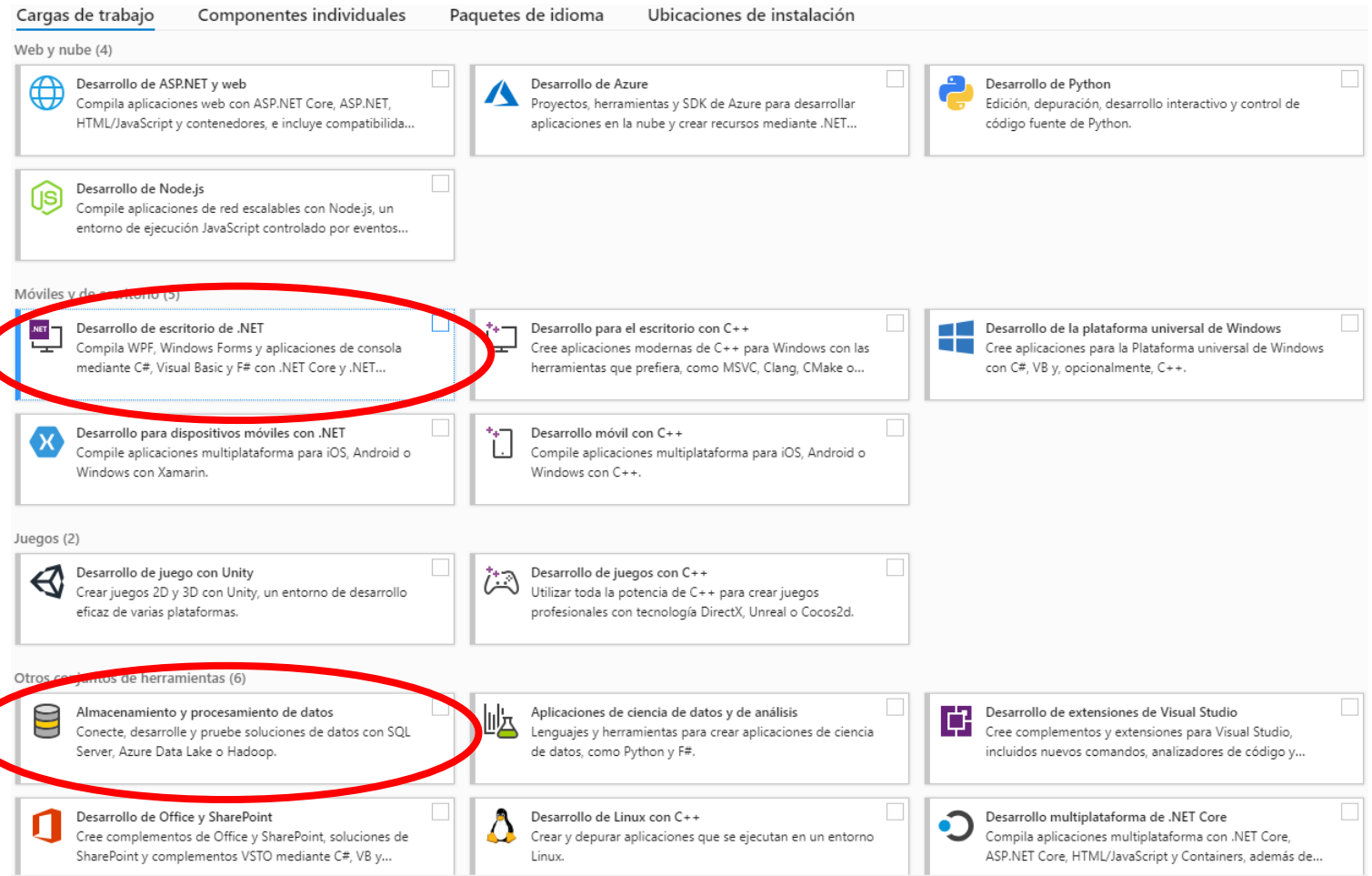
4 - Visual Studio 2019

- Paquetes a instalar:

- Para los módulos 1 y 2

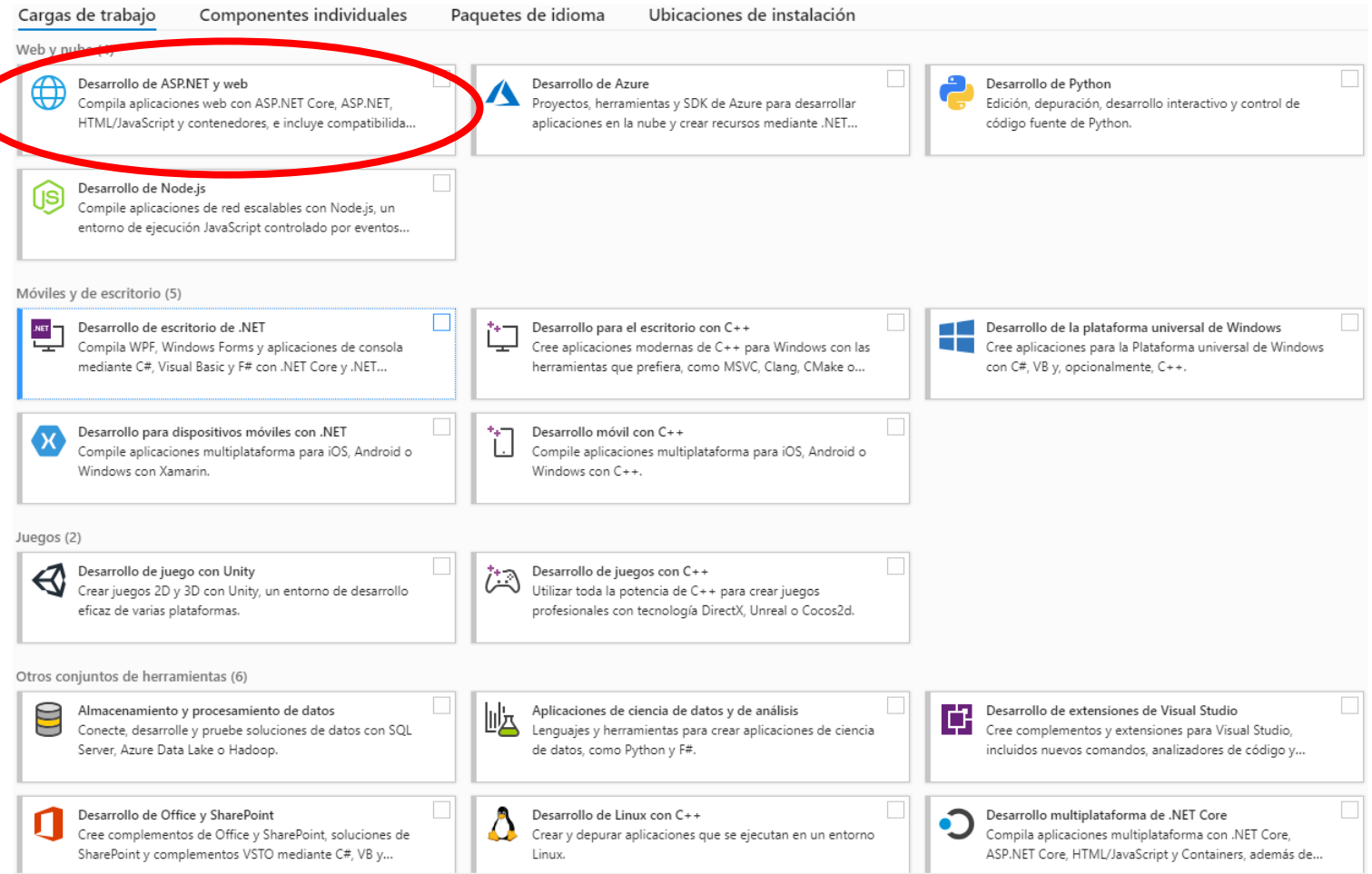


- Para el módulo 3



4 - Visual Studio 2019

- Paquetes a instalar:
 - Para el módulo 4



05

Primer programa: Hello World!

5 - Hello World!

- En este primer módulo del curso vamos a programar en **entorno de consola**
- Hello World! es un programa inicial clásico en el mundo de la programación, que lo único que hace es escribir una línea de texto por pantalla: Hello World!
- A veces es utilizado para comparar distintos lenguajes entre sí
 - Se compara la sencillez o complejidad léxica y sintáctica (nº de instrucciones, nº de líneas, nº de palabras, nº de caracteres, etc...) necesaria para escribir un programa muy sencillo (para muchos, es el programa más sencillo que se puede escribir)

5 - Creando programa Hello World!

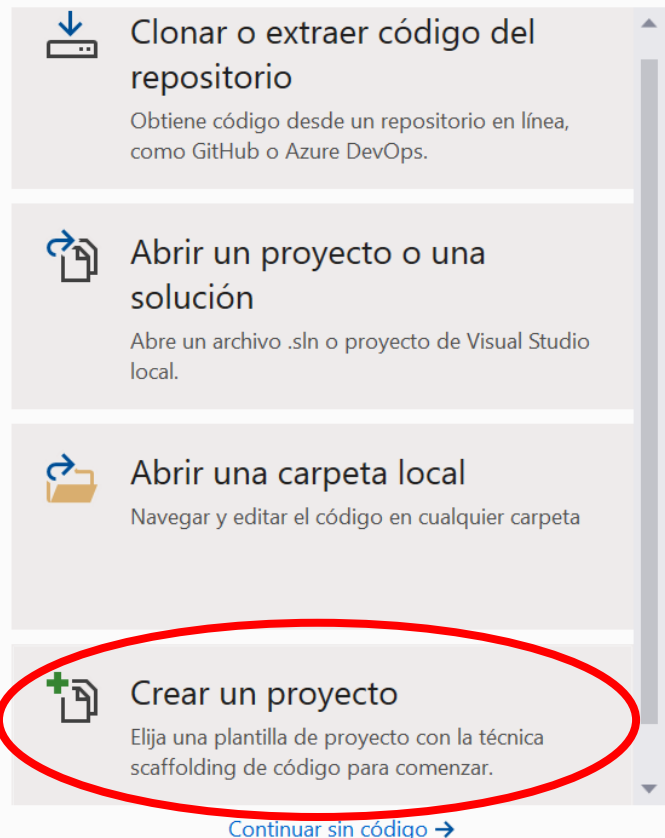
Visual Studio 2019


Abrir recientes


Cuando abra proyectos, carpetas o archivos en Visual Studio, se mostrarán aquí para obtener un acceso rápido.


Puede anclar cualquier elemento que abra con frecuencia para que aparezca siempre en la parte superior de la lista.


Tareas iniciales



 **Clonar o extraer código del repositorio**
Obtiene código desde un repositorio en línea, como GitHub o Azure DevOps.

 **Abrir un proyecto o una solución**
Abre un archivo .sln o proyecto de Visual Studio local.

 **Abrir una carpeta local**
Navegar y editar el código en cualquier carpeta

 **Crear un proyecto**
Elija una plantilla de proyecto con la técnica scaffolding de código para comenzar.

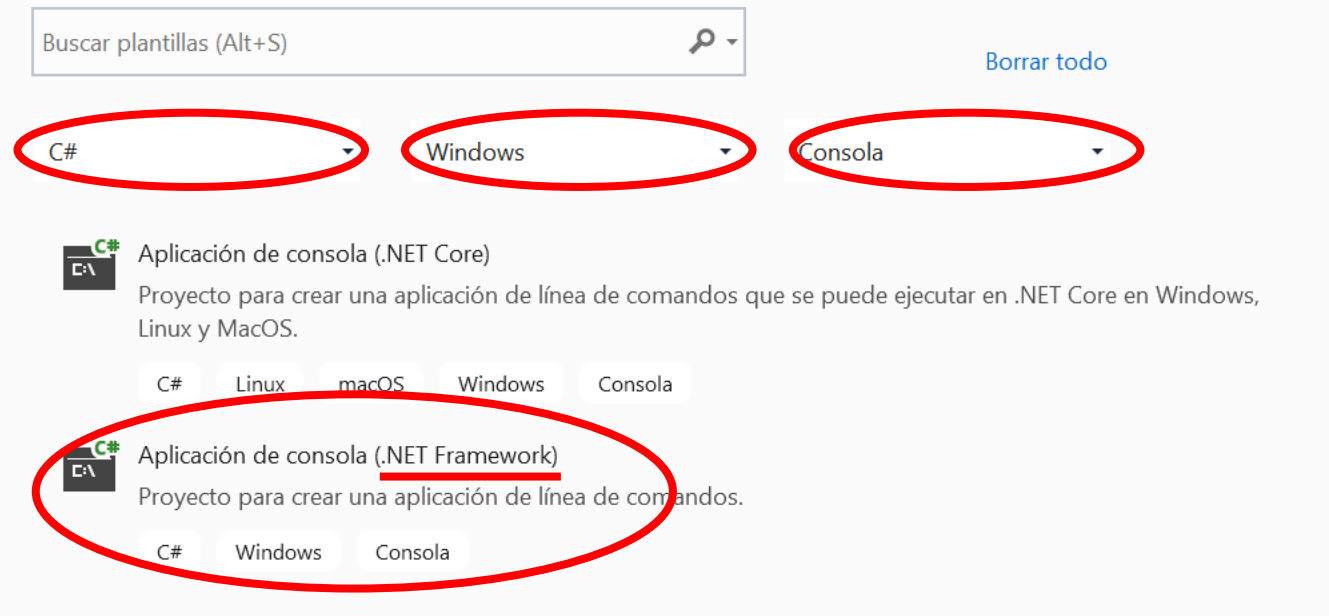
[Continuar sin código ->](#)

5 - Creando programa Hello World!

Crear un proyecto

Plantillas de proyecto recientes

Aquí se mostrará una lista de las plantillas usadas recientemente.



Seleccionar lenguaje **C#**, plataforma **Windows**, Interfaz **Consola**, proyecto **Aplicación de consola (.NET Framework)**

5 - Creando programa Hello World!

- Introducir un nombre para el proyecto
Por ejemplo: MiPrimerProyectoDeConsola
(el nombre de la solución cambia automáticamente para ser igual al nombre del proyecto. Dejarlo así)
- La ubicación es la carpeta del sistema donde se guardarán los archivos necesarios para escribir, traducir y ejecutar los programas que escribamos en el proyecto o los proyectos contenidos en esta solución
- La versión del Framework indicará qué funciones ya hechas nos ofrecerá .NET Framework para poder utilizar en nuestros programas. Cuanto más grande el número de versión, más funciones nuevas

Configure su nuevo proyecto

Aplicación de consola (.NET Framework) C# Windows Consola

Nombre del proyecto

ConsoleApp3

Ubicación

C:\Users\smilla\source\repos

Nombre de la solución ⓘ

ConsoleApp3

☐ Colocar la solución y el proyecto en el mismo directorio

Framework

.NET Framework 4.7.2

5 - Escribiendo programa Hello World!

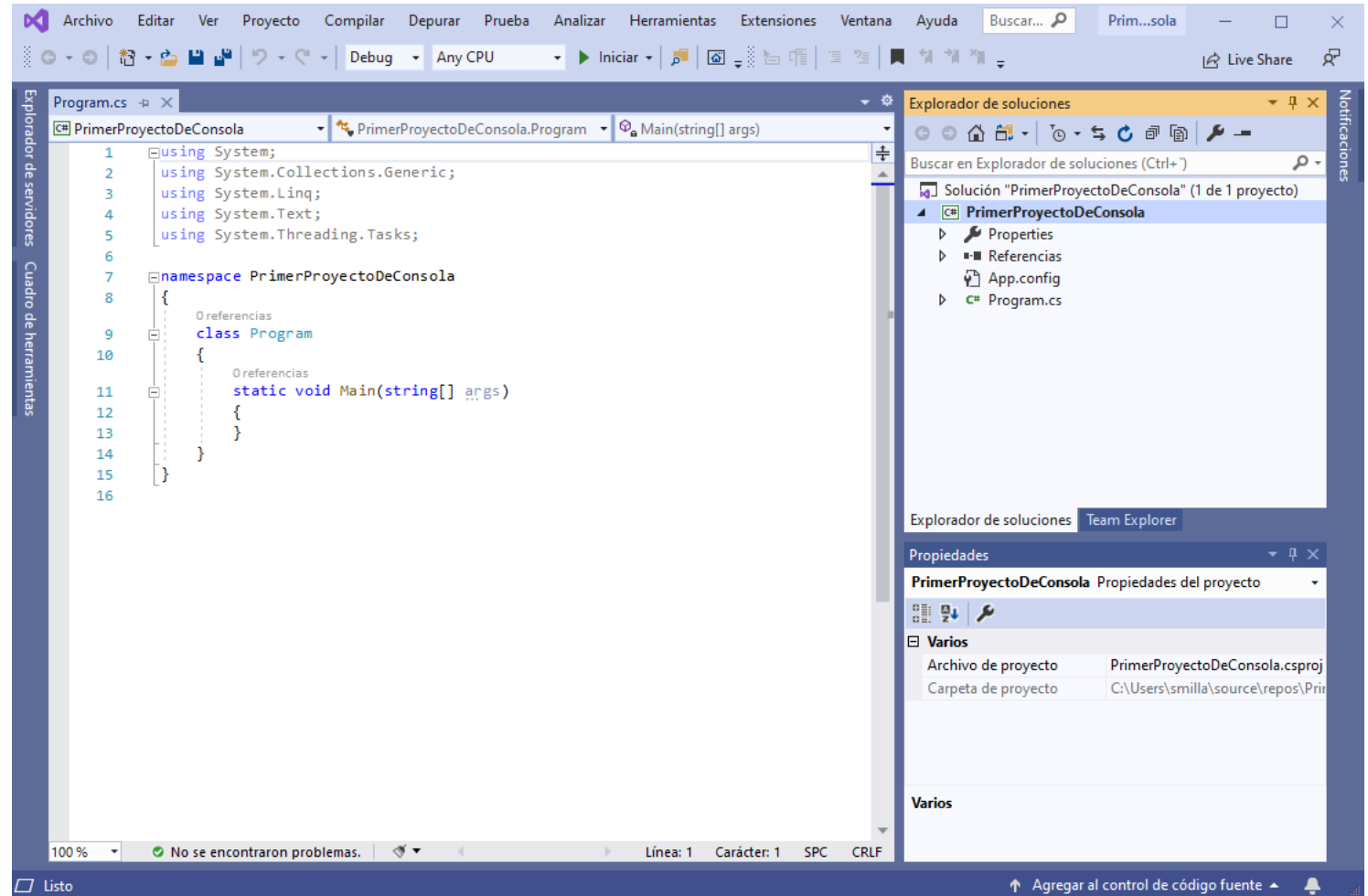
Pantalla inicial de Visual Studio para un nuevo proyecto de Consola de .NET Framework

Ahora, ¡a escribir nuestro programa!

Para eso:

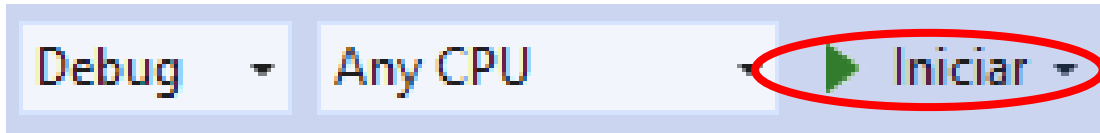
- haremos click en la línea 12, después del carácter '{', y pulsaremos la tecla Intro del teclado
- Desde la línea en blanco añadida, escribiremos dos líneas:

```
Console.WriteLine("Hello, World!");  
Console.ReadKey();
```



5 - Ejecutando programa Hello World!

Una vez escrito, para ejecutarlo pulsar botón iniciar (Play verde) de la parte superior:



Si todo ha ido bien, se abrirá una ventana nueva con el siguiente contenido:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PrimerProyectoDeConsola
{
    Oreferencias
    class Program
    {
        Oreferencias
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
            Console.ReadKey();
        }
    }
}
```


5 - Ejecutando programa Hello World!

¡Felicidades!

- Hemos escrito, compilado y ejecutado nuestro primer programa de .NET Framework con C#
- Posiblemente no tengamos ni idea de lo que hemos escrito, ni de lo que hemos hecho para ejecutarlo
- Pero al menos ya hemos tenido nuestra primera experiencia como programadores (ya no nos lo cuentan)
- En los próximos temas entenderemos:
 - Qué nos pueden pedir que programemos, y cómo nos lo van a pedir: Enunciado del problema
 - Qué podemos hacer con el enunciado para sacar de él información que nos permita escribir nuestro programa
 - Cómo usar la información extraída del enunciado para organizar las instrucciones de nuestro programa

5 - Ejecutando programa Hello World!

... pero antes de empezar esos próximos temas... ¡A JUGAR con este programa!

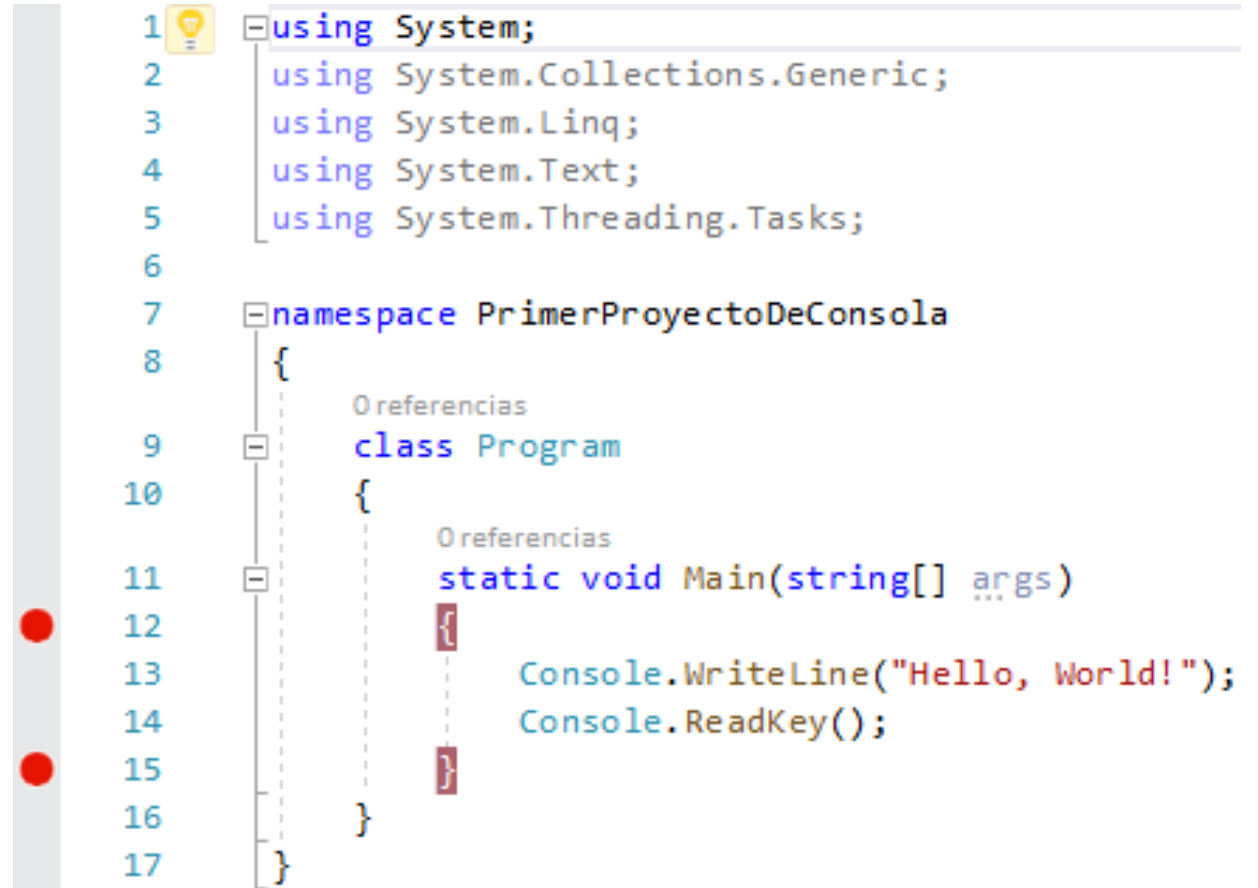
- Cambiad el texto que se escribe por pantalla
- Probad a escribir un número en vez de un texto, o incluso una operación numérica
- Probad a escribir algo que mezcle letras y números
- Probad a cambiar algo del código que produzca un error durante la compilación
- Probad a cambiar algo del código que produzca un error durante la ejecución
- Probad a buscar por Internet información de qué hacen las líneas que habéis escrito, tanto las del programa original como las que habéis escrito vosotr@s durante este juego de exploración
- Pensad en algo nuevo que os gustaría que hiciera este programa, y buscad información en Internet que os ayude a escribir el código necesario para conseguir el objetivo que os habéis propuesto...

5 - Ejecutando programa Hello World!

Ejecución paso a paso de instrucciones

Breakpoints: puntos de paro

- Al marcar un punto de paro, el texto de la línea correspondiente se muestra en rojo
- Para marcar un punto de paro, 2 opciones:
 - Click (botón izquierdo) en sombreado a la izquierda del nº de línea
 - Click (botón derecho) sobre la línea-> Punto de interrupción-> Marcar punto de interrupción
- En la imagen se han marcado dos puntos de paro: uno antes de la primera línea del programa, y otro después de la última línea del programa



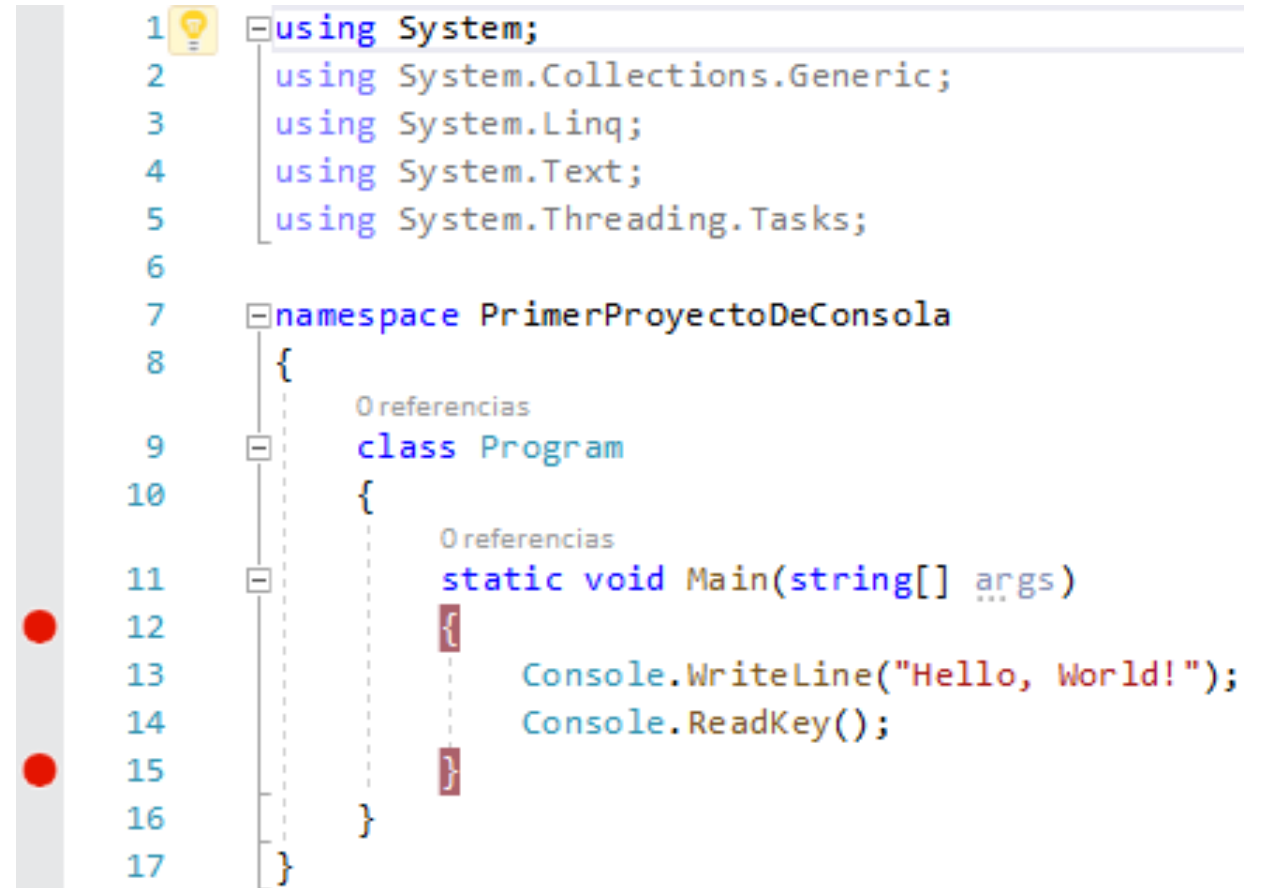
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace PrimerProyectoDeConsola
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13             Console.WriteLine("Hello, World!");
14             Console.ReadKey();
15         }
16     }
17 }
```

5 - Ejecutando programa Hello World!

Ejecución paso a paso de instrucciones

Breakpoints: puntos de paro

- Al ejecutar este programa, se pausará su ejecución cuando esté preparado para ejecutar su primera línea del programa.
- Para seguir ejecutando hasta el próximo punto de paro (o, si no hay más breakpoints, hasta el final del programa), pulsar el play verde (botón continuar) de la parte superior, o pulsar tecla F5



```
1  [?] using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace PrimerProyectoDeConsola
8  {
9      [?]
10     {
11         [?]
12         {
13             [?]
14             Console.WriteLine("Hello, World!");
15             Console.ReadKey();
16         }
17     }
18 }
```

The image shows a code editor window with a C# program. The code is as follows:

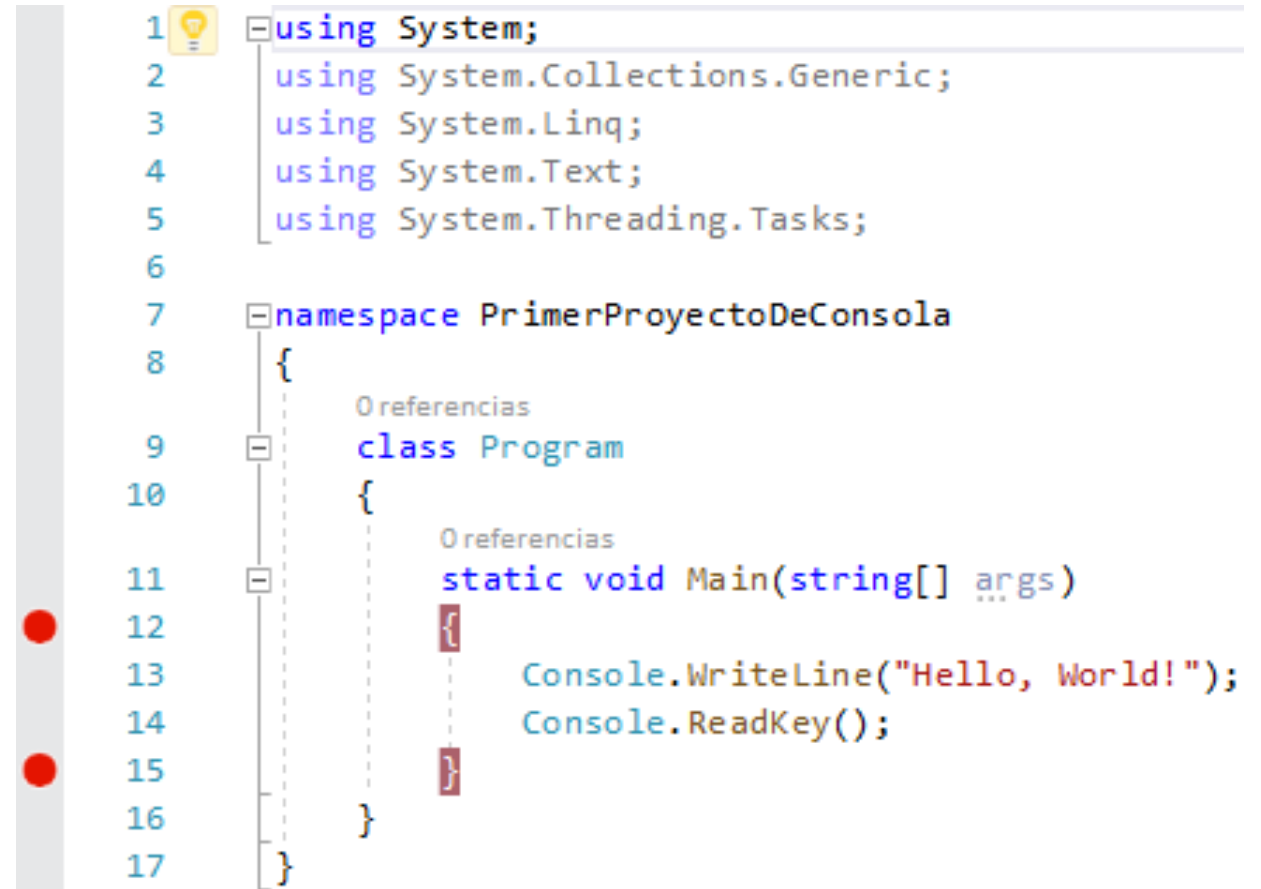
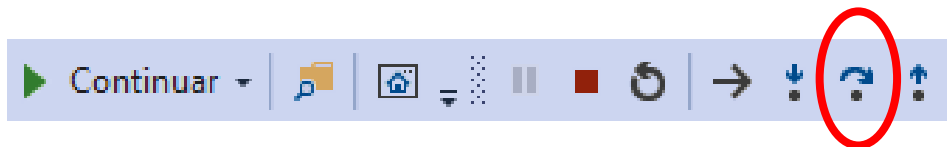
```
1  [?] using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace PrimerProyectoDeConsola
8  {
9      [?]
10     {
11         [?]
12         {
13             [?]
14             Console.WriteLine("Hello, World!");
15             Console.ReadKey();
16         }
17     }
18 }
```

Line numbers 1 through 17 are visible on the left. A yellow lightbulb icon is next to line 1. A vertical line with two red circular breakpoints is positioned to the left of the code. The first breakpoint is at line 9, and the second is at line 11. The code is color-coded: keywords are blue, namespace and class names are dark blue, and string literals are red.

5 - Ejecutando programa Hello World!

Ejecución paso a paso de instrucciones
Breakpoints: puntos de paro

- Para ejecutar solo la instrucción de la línea en la que está parado el programa (que en ese momento se muestra en amarillo) y pasar a la siguiente, pulsar F11 o buscar botón marcado en rojo en la siguiente imagen:



MUCHAS GRACIAS



Pasión por la innovación

www.integratecnologia.es