

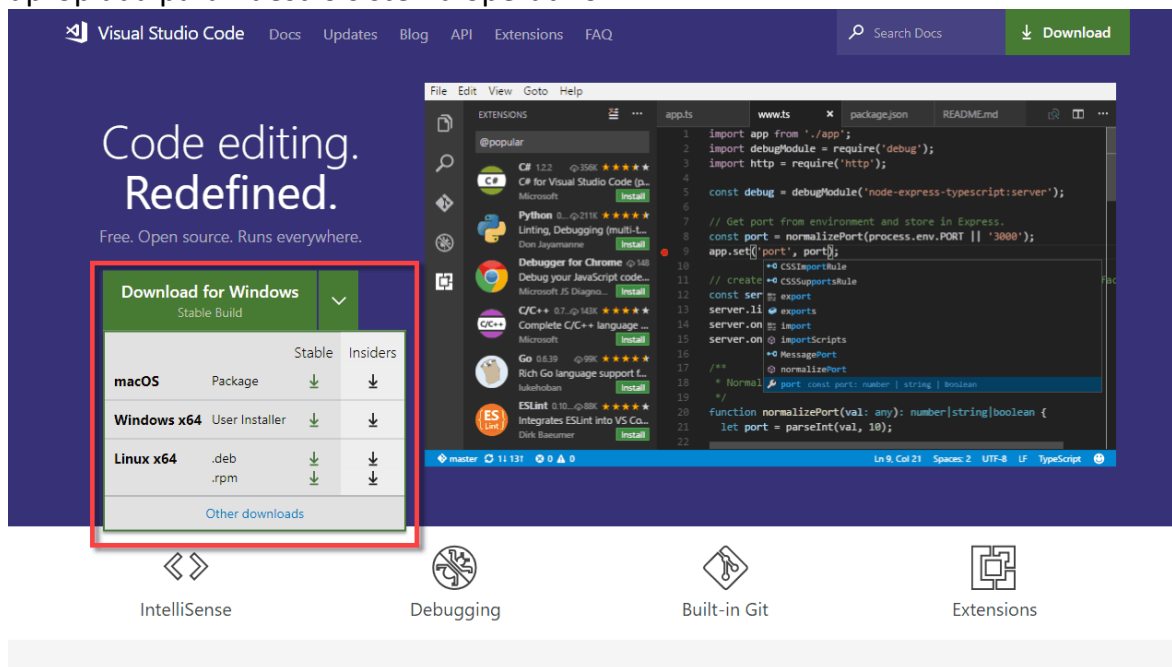
Visual Studio Code: cómo preparar un entorno de trabajo para .NET Core

Una de las grandes ventajas de .NET Core, es su ejecución multiplataforma, lo que nos permite trabajar en entornos que no sean Windows. Es por eso que Microsoft lanzó al mercado su IDE (Entorno de Desarrollo Integrado, en inglés: *Integrated Development Environment*) **gratuito y multiplataforma** [Visual Studio Code](#).

En principio, programar para .NET Core con Visual Studio Code puede parecer algo confuso, ya que todo funciona por comandos, hacen falta algunos ficheros JSON que no son necesarios en Visual Studio y aparentemente tiene las herramientas limitadas. Sin embargo, en realidad es muy fácil de configurar y no vas a notar grandes carencias respecto a su hermano mayor, por lo que puede convertirse en una gran opción, más ágil y que además podrás usar en Mac o Linux.

Instalando Visual Studio Code

Lo primero que debes hacer es instalar Visual Studio Code. Para ello basta con que entremos a su página de descargas y nos descarguemos e instalemos la versión apropiada para nuestro sistema operativo.



Instalando .NET Core

Visual Studio Code en sí mismo es un editor de ficheros "vitaminado". Esto quiere decir que es como el bloc de notas de toda la vida, pero con mejoras (muchas mejoras). Por eso, lo primero que tenemos que hacer para poder usar .NET Core con Visual Studio Code, es instalar .NET Core para nuestra plataforma. Para ello descargaremos desde su [página de descargas](#) los binarios del SDK (Kit de Desarrollo de Software, en inglés: *Software Development Kit*) para nuestro sistema operativo.

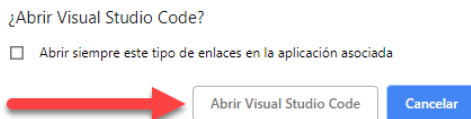
Instalando las extensiones necesarias

Para poder desarrollar todo su potencial, Visual Studio Code utiliza un sistema de extensiones que nos permiten ampliar su funcionalidad. Estas extensiones se pueden descargar desde el propio entorno o desde el Extensión Marketplace (aunque esto realmente, lo que va a hacer es abrir el IDE y llevarnos a la extensión).

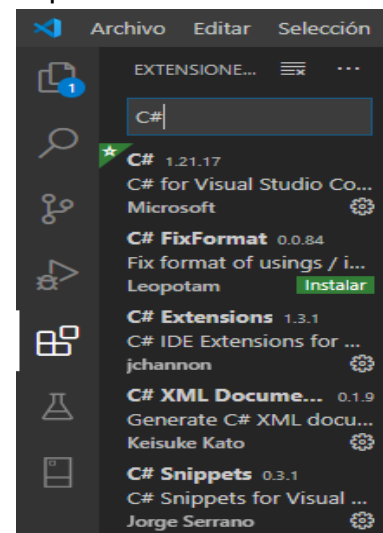
Para empezar a preparar nuestro entorno, vamos a utilizar la extensión para el lenguaje C#.

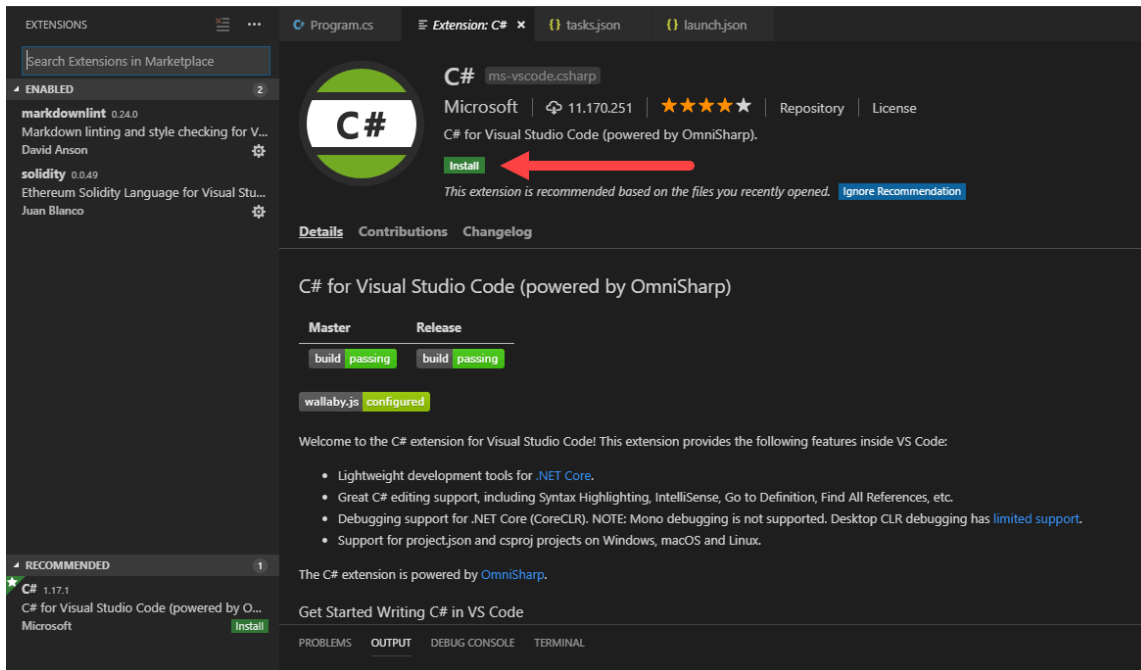
Para instalar esta extensión, desde la web, basta con que pulsemos sobre `Install`:

Nos va a pedir como requisito tener instalado Visual Studio Code, y como nosotros ya lo tenemos, en la ventana emergente pulsamos sobre el botón **Abrir Visual Studio Code**:

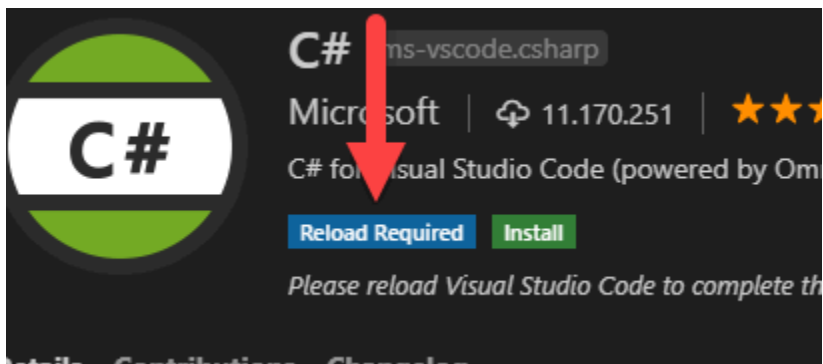


Esto nos abre Visual Studio Code, buscamos directamente la extensión, aquí basta con pulsar sobre el botón `Install` para que se inicie el proceso:





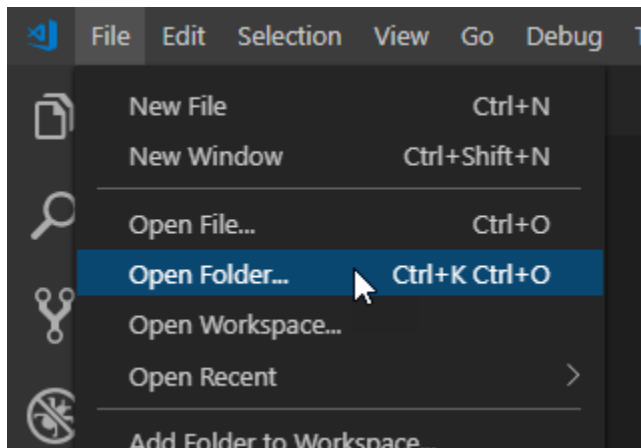
Una vez que termine tenemos que recargar el IDE. Para eso basta con pulsar en el botón de recargar:



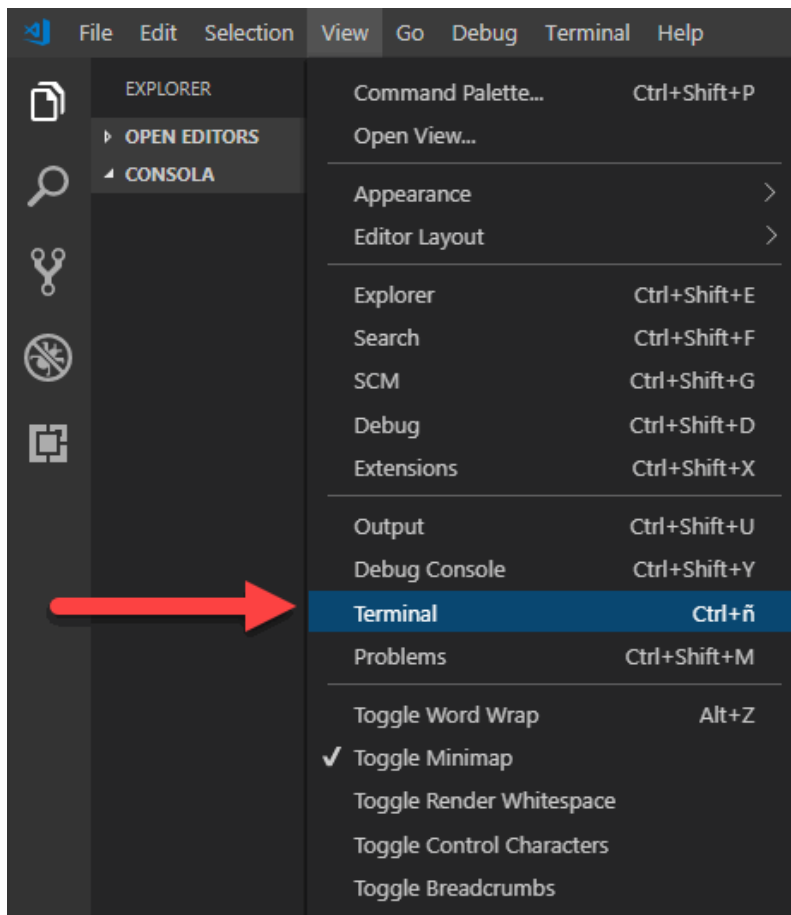
Con esto, ya tenemos nuestro entorno de trabajo Visual Studio Code para trabajar con el lenguaje C# y con .NET Core.

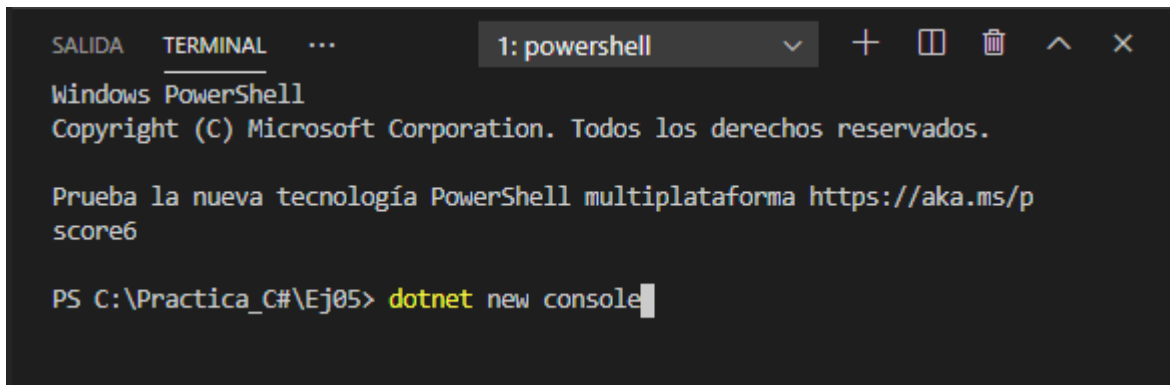
Creando una aplicación de consola .NET Core en Visual Studio Code

Lo primero que vamos a necesitar es crear una carpeta en la que queramos que esté el proyecto. Una vez que la tengamos vamos a abrir en Visual Studio Code la carpeta que acabamos de crear, utilizando el menú **File Open Folder** o con el acceso directo **Ctrl+K, Ctrl+O** (son dos combinaciones de teclas, una después de la otra):



Una vez que tengamos la carpeta, vamos a necesitar sacar una ventana de terminal para lanzar comandos de compilación y similares. Para eso, dentro del menú **View**, pulsamos sobre el botón **Terminal** o con el acceso directo **Ctrl+ñ**:



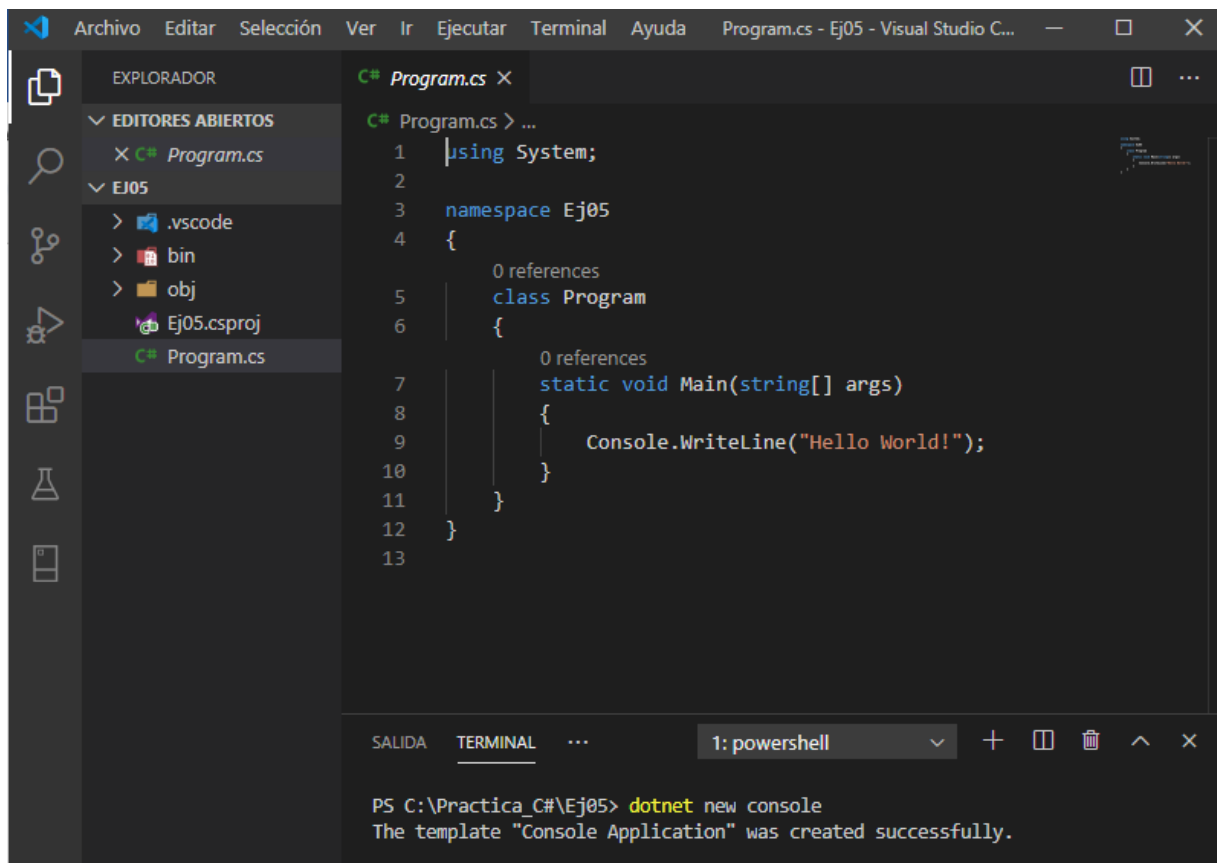


```
SALIDA  TERMINAL  ...  1: powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/p
score6

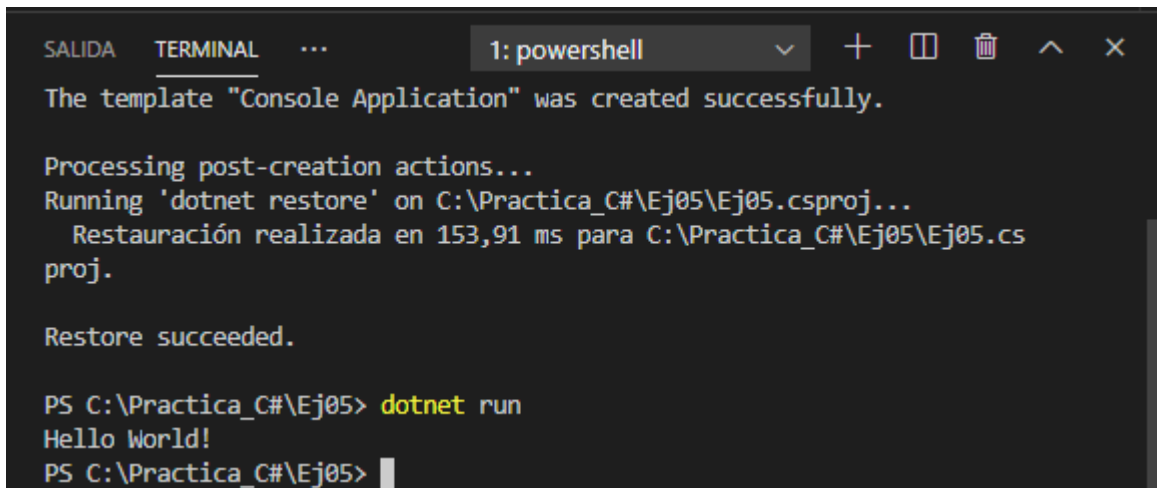
PS C:\Practica_C#\Ej05> dotnet new console
```

Ahora dentro de esa terminal vamos a ejecutar el comando **dotnet new console** para crear nuestro proyecto (éste tomará el nombre de la carpeta en la que estamos). Cuando acabe la ejecución, podemos ver que han aparecido los ficheros del proyecto en el explorador:



```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  Program.cs - Ej05 - Visual Studio C...
EXPLORADOR
EDITORES ABIERTOS
C# Program.cs
EJ05
  > .vscode
  > bin
  > obj
  > Ej05.csproj
  C# Program.cs
C# Program.cs
1  using System;
2
3  namespace Ej05
4  {
5      0 references
6      class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             Console.WriteLine("Hello World!");
12         }
13     }
SALIDA  TERMINAL  ...  1: powershell
PS C:\Practica_C#\Ej05> dotnet new console
The template "Console Application" was created successfully.
```

Con esto, si ejecutamos desde el terminal el comando **dotnet run**, podemos ver como la aplicación se ejecuta sin ningún problema:



```
SALIDA  TERMINAL  ...  1: powershell  +  [ ]  [X]  ^  X

The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on C:\Practica_C#\Ej05\Ej05.csproj...
  Restauración realizada en 153,91 ms para C:\Practica_C#\Ej05\Ej05.csproj.

Restore succeeded.

PS C:\Practica_C#\Ej05> dotnet run
Hello World!
PS C:\Practica_C#\Ej05> 
```

En resumen

Con los pasos anteriores hemos conseguido un entorno de trabajo básico, pero totalmente funcional para .NET Core con Visual Studio Code. Estaremos en condiciones de trabajar con .NET Core sin problemas y sacarle partido a todas sus características, apoyados por Visual Studio Code y todas sus potentes funcionalidades:

- **IntelliSense** con sugerencias mientras escribimos código
- **"Snippets"** o fragmentos de código ya hecho para acelerar la escritura
- Localización y navegación rápida por el código
- **CodeLens** para obtener información sobre referencias y relaciones entre el código
- Refactorización

ALGORITMOS Y CÓDIGO

Analice e indique qué hace el siguiente algoritmo

```
Algoritmo ejemplo()  
entero d1, d2, d3  
Lea d1, d2, d3  
si(d1=d2 ∧ d2=d3) entonces  
    muestre d1, d2, d3  
sino  
    si (d1 >= d2 ∧ d2 >= d3) entonces  
        muestre d1, d2, d3  
    sino  
        si (d1 >= d3 ∧ d3 >= d2) entonces  
            muestre d1, d3, d2  
        sino  
            si (d2 >= d1) ∧ (d1 >= d3) entonces  
                muestre d2, d1, d3  
            sino  
                si (d2 >= d3 ∧ d3 >= d1) entonces  
                    muestre d2, d3, d1  
                sino  
                    si (d3 >= d1 ∧ d1 >= d2) entonces  
                        muestre d3, d1, d2  
                    sino  
                        muestre d3, d2, d1  
                fin si  
            fin si  
        fin si  
    fin si  
fin si  
fin algoritmo ejemplo
```

Después de analizarlo pasamos a realizar la codificación en una aplicación de consola en los siguientes lenguajes VB.NET y C#. También agregamos al código la condición cuando los números que se digiten sean iguales y cuando se digiten solo ceros (0), se debe visualizar el mensaje correspondiente

CODIGO VB.NET

```
Module Module1  
    Sub Main()  
        Dim d1, d2, d3 As Integer  
        Dim linea As String  
        Console.WriteLine("Digite 1er N° entero.....: ")  
        linea = Console.ReadLine()  
        d1 = Integer.Parse(linea)  
        Console.WriteLine("Digite el 2do entero.....: ")  
        linea = Console.ReadLine()
```

```

d2 = Integer.Parse(linea)
Console.Write("Digite el 3er entero.....: ")
linea = Console.ReadLine()
d3 = Integer.Parse(linea)

If (d1 > d2) And (d2 > d3) Then
    Console.Write(d1 & d2 & d3)
    Console.ReadKey()
Else
    If (d1 >= d2) And (d2 >= d3) Then
        Console.WriteLine(d1 & d2 & d3)
        Console.ReadKey()
    Else
        If (d1 >= d3) And (d3 >= d2) Then
            Console.WriteLine(d1 & d3 & d2)
            Console.ReadKey()
        Else
            If (d2 >= d1) And (d1 >= d3) Then
                Console.WriteLine(d2 & d1 & d3)
                Console.ReadKey()
            Else
                If (d2 >= d3) And (d3 >= d1) Then
                    Console.WriteLine(d2 & d3 & d1)
                    Console.ReadKey()
                Else
                    If (d3 >= d1) And (d1 >= d2) Then
                        Console.WriteLine(d3 & d1 & d2)
                        Console.ReadKey()
                    Else
                        Console.WriteLine(d3 & d2 & d1)
                    End If
                End If
            End If
        End If
    End If
End If
End Sub
End Module

*****

```

CODIGO C#

Codificamos el mismo algoritmo en lenguaje C#, con esta actividad vamos analizando las diferencias de sintaxis entre estos 2 lenguajes de la familia Microsoft.

```

using System;

namespace Orden_Numeros
{
    static class Module1
    {
        public static void Main()
        {
            int d1, d2, d3;
            string linea;
            Console.Write("Digite 1er N° entero.....: ");

```



```

linea = Console.ReadLine();
d1 = int.Parse(linea);
Console.Write("Digite el 2do entero....: ");
linea = Console.ReadLine();
d2 = int.Parse(linea);
Console.Write("Digite el 3er entero....: ");
linea = Console.ReadLine();
d3 = int.Parse(linea);
if (d1 > d2 & d2 > d3)
{
    Console.Write(d1.ToString() + d2 + d3);
    Console.ReadKey();
}
else if (d1 >= d2 & d2 >= d3)
{
    Console.WriteLine(d1.ToString() + d2 + d3);
    Console.ReadKey();
}
else if (d1 >= d3 & d3 >= d2)
{
    Console.WriteLine(d1.ToString() + d3 + d2);
    Console.ReadKey();
}
else if (d2 >= d1 & d1 >= d3)
{
    Console.WriteLine(d2.ToString() + d1 + d3);
    Console.ReadKey();
}
else if (d2 >= d3 & d3 >= d1)
{
    Console.WriteLine(d2.ToString() + d3 + d1);
    Console.ReadKey();
}
else
{
    if (d3 >= d1 & d1 >= d2)
    {
        Console.WriteLine(d3.ToString() + d1 + d2);
        Console.ReadKey();
    }
    else
    {
        Console.WriteLine(d3.ToString() + d2 + d1);
    }

    Console.ReadKey();
}
}
}
}

```

