Repaso C#.NET

Sistema de Aprendizaje Automático



|  |
| --- |
| **C#**: <https://learn.microsoft.com/es-es/dotnet/csharp/>  **Documentación de C#**:  <https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/>  **Paseo por el lenguaje C#**: <https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/overview> |

Índice

[**Creación de Proyecto en Visual Studio 1**](#_n9epmjo6lglg)

[Actividad 3](#_ww2qb54jctrs)

[**Ejecución del Proyecto en Visual Studio 4**](#_8ntuun2difvo)

[Actividad 4](#_3g2c5ol420ml)

[**Declaración de variables 5**](#_v7u5kad3g2f9)

[Actividad 5](#_tg8p1xuy7h4x)

[**Mostrar/Pedir información por pantalla 6**](#_6kcf3f4cryn8)

[Actividad 6](#_5d1zmw1zgoll)

[**Condicionales 7**](#_y6brwejd5j3n)

[Actividad 7](#_m3dil81uie21)

[**Bucles 8**](#_osktb4lrat71)

[Actividad 8](#_u7ete17e106a)

[**Arrays 10**](#_bs5l377dbyq1)

[Actividad 10](#_avw6tt4ovnri)

[**Clases y Structs 11**](#_k0goq0e2m36)

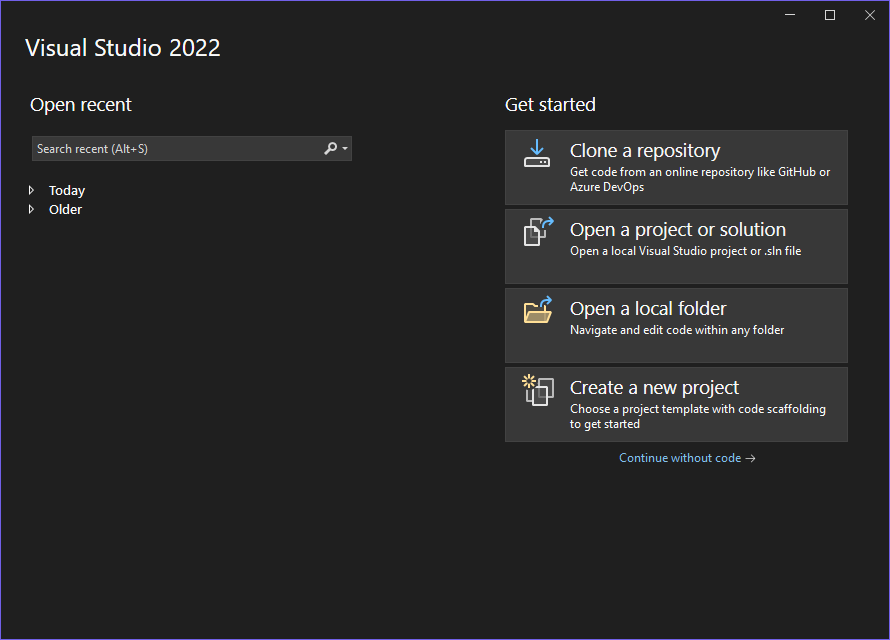
[Actividad 12](#_ps3qfpcdn5f2)

[**Leer de ficheros 13**](#_wvg3emvf1b8s)

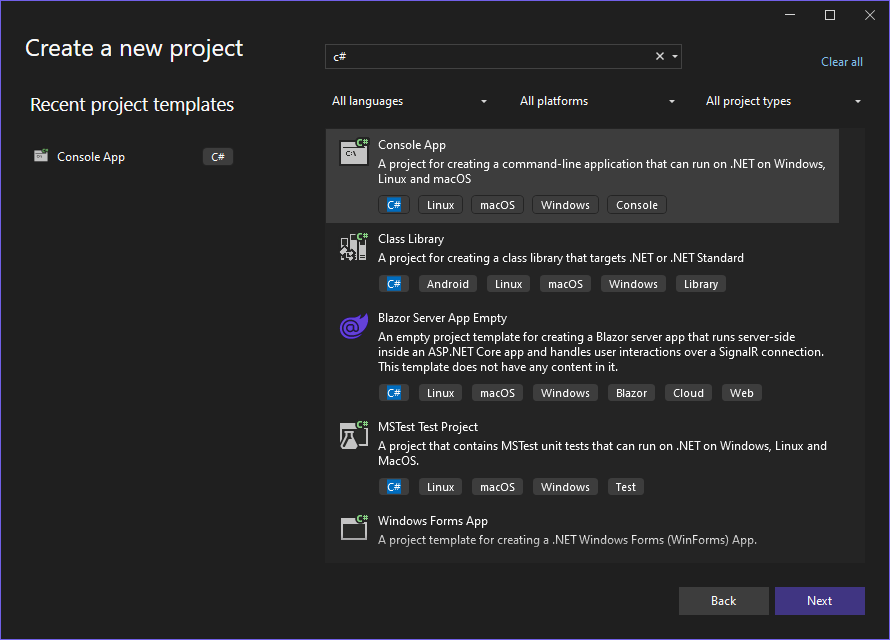
[Actividad 15](#_yy4ll7w8iznc)

# Creación de Proyecto en Visual Studio

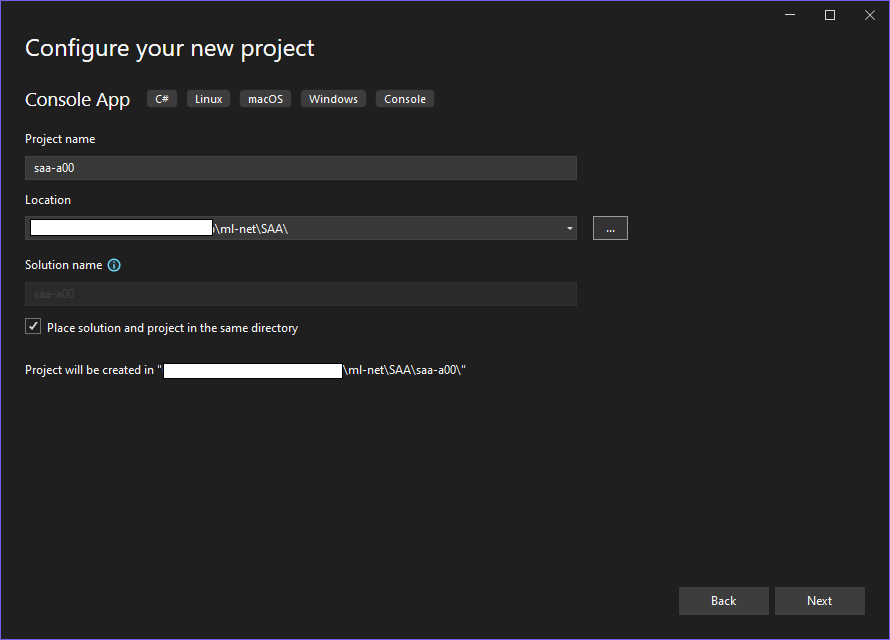
Una vez iniciada la aplicación, hacemos clic en “***Create a new project***”.



Vamos a usar una plantilla, en este caso vamos a utilizar “***Console App***” con el lenguaje C#. Para asegurarnos de que es C#, en el buscador escribimos C# y veremos en azul que efectivamente es la que necesitamos. Lo seleccionamos.

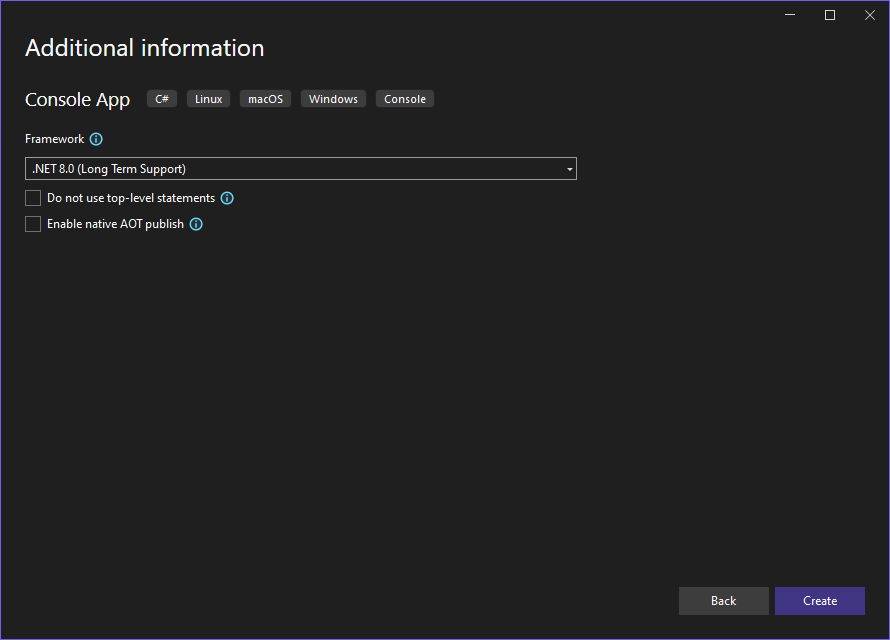


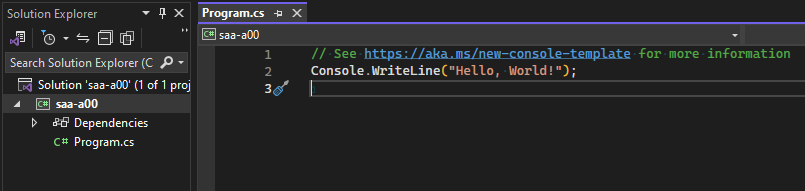
Una vez hayamos hecho clic en “***Next***”, establecemos un nombre para el proyecto. En este caso será “saa-a00” (sistemas de aprendizaje automático - actividad 00). Indicamos la ruta donde se va a crear (os recomiendo que creéis una ruta para las actividad de SAA). Y debemos marcar el check “***Place solution and project in the same directory***”.



Una vez hecho clic en “***Next***”, debemos seleccionar la versión del framework. En este caso vamos a usar .NET 8 (LTS).

|  |
| --- |
| ⚠️  La versión **.NET 9** está a punto de salir.  <https://learn.microsoft.com/es-es/dotnet/core/whats-new/dotnet-9/overview> |



Una vez hayamos hecho clic en “***Create***”, tendremos el proyecto listo para ejecutarse.

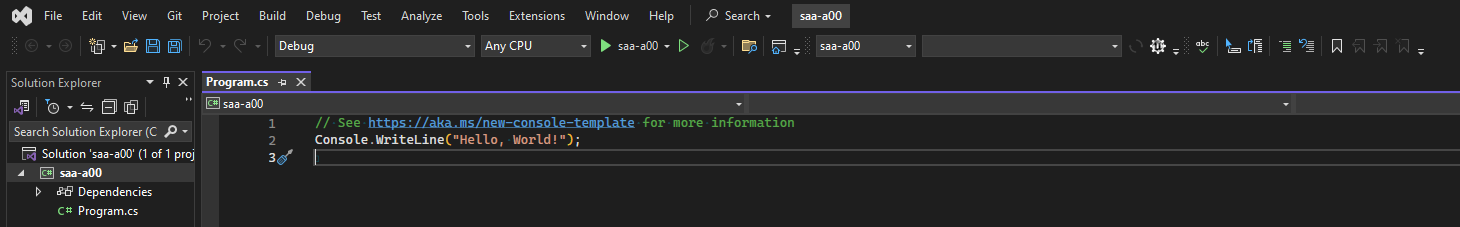
## Actividad

Crear un proyecto y mostrar el contenido de “**Program.cs**”:

|  |
| --- |
|  |

# Ejecución del Proyecto en Visual Studio

Una vez creado el proyecto, si hacemos clic en el botón verde (***triángulo***), también podemos hacer uso de la tecla **F5**, se ejecutará el programa.



En mi caso se ejecuta en una terminal:



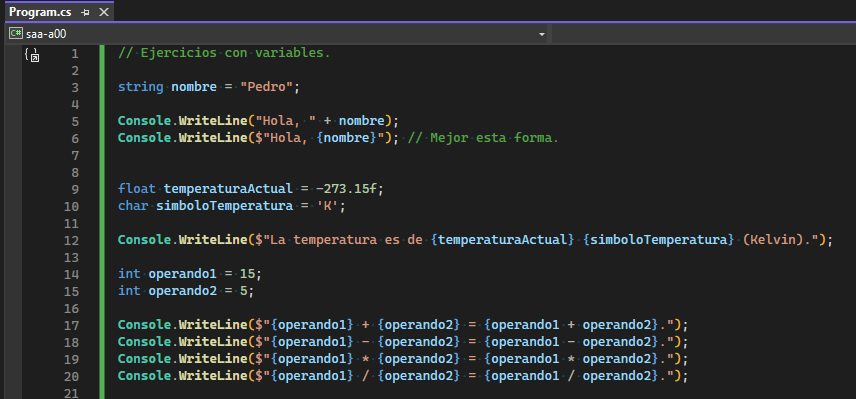
## Actividad

Ejecutar la aplicación y mostrar la consola entera:

|  |
| --- |
|  |

# Declaración de variables

En el siguiente código podemos ver diferentes tipos de variables y operaciones:



## Actividad

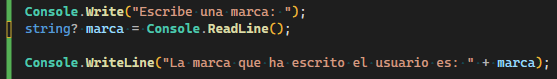
1. Crear un programa que muestre la tabla de multiplicar del 5.
2. Concatenar vuestro nombre, apellidos y edad en la misma línea (cada valor es una variable).

|  |
| --- |
|  |

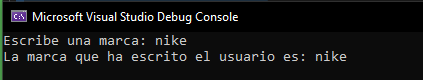
# Mostrar/Pedir información por pantalla

Podemos hacer uso de **Console.Write** o **Console.WriteLine**. El primero no hace nueva línea (el equivalente a la tecla enter) y el segundo crea una nueva línea.

Para leer datos de la consola, debemos hacer uso de **Console.ReadLine**. Este devuelve el valor en forma de **string**, concretamente **string?**, es decir, un string que puede ser **null**.



Si lo ejecutamos tenemos que:



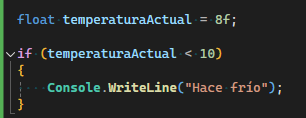
|  |
| --- |
| ⚠️  C# tiene el concepto de **tipo nullable**, para ello se hace uso del operador **?**.  Esto permite escribir código más seguro, de tal modo que el programador en base a esta característica se verá forzado a hacer las comprobaciones evitando crasheos del programa por **NullReferenceException**.  <https://learn.microsoft.com/es-es/dotnet/csharp/language-reference/builtin-types/nullable-value-types> |

## Actividad

1. Crear un programa que muestre la tabla de multiplicar de un número que pida por pantalla.
2. Concatenar vuestro nombre, apellidos y edad en la misma línea, cada valor es una variable que pide al usuario.

|  |
| --- |
|  |

# Condicionales



Si lo ejecutamos:



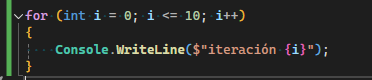
## Actividad

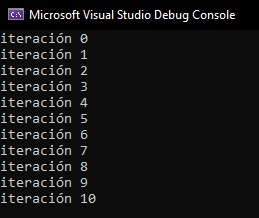
Crear un programa que:

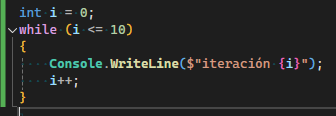
1. Pida al usuario un número
2. Pida al usuario un segundo número
3. Muestre si son iguales
4. Muestre si el primero es mayor que el segundo
5. Muestre si el segundo es menor o igual que el primero
6. Muestre si son distintos

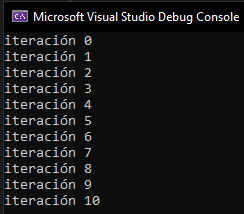
|  |
| --- |
| *mostrar resultados* |

# Bucles









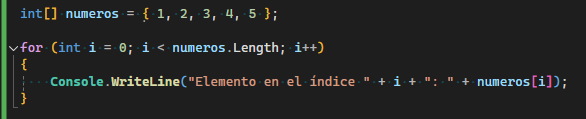
## Actividad

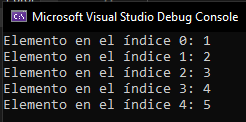
Crear un programa que pida por consola un número y calcule su tabla de multiplicar:

|  |
| --- |
| *mostrar resultados* |

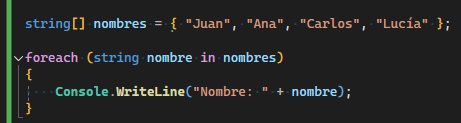
# Arrays

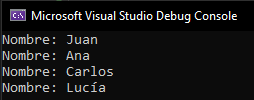
Creación de array de enteros (int) y recorrer con for i.





Creación de array de cadenas de caracteres (string) y recorrer con for each.



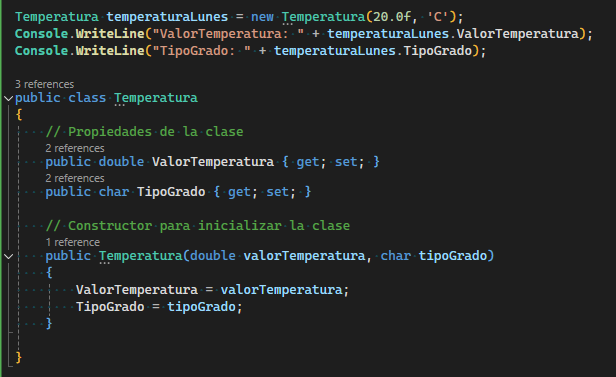


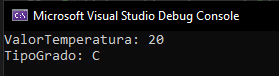
## Actividad

Crear un programa que recorre un array vacío de tamaño 5, de tipo string, y lo va rellenando, pidiendo por consola un nombre.

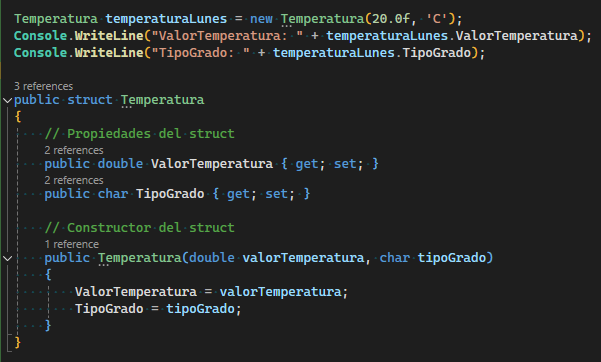
|  |
| --- |
| *mostrar resultados*  ***Hint****: string[] numeros = new string[5];* |

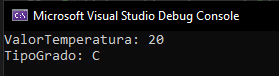
# Clases y Structs





Structs





## Actividad

Crear un programa que tenga la estructura Mascota, que tiene un nombre y edad:

* Crear 2 mascotas y mostrar su datos.

Crear un programa que tenga la clase Persona, que tiene un nombre, apellido1, apellido2 y edad y una Mascota:

* Crear 2 personas y mostrar su datos.
* Asignar a la propiedad mascota a las creadas al principio

|  |
| --- |
| *mostrar resultados* |

# Leer de ficheros

Leer línea por línea:

|  |
| --- |
| using System;  using System.IO;  class Program  {  static void Main()  {  // Especifica la ruta del archivo  string rutaArchivo = "ruta/del/archivo.txt";    try  {  // Usar StreamReader para leer el archivo línea por línea  using (StreamReader lector = new StreamReader(rutaArchivo))  {  string linea;  // Leer línea por línea hasta el final del archivo  while ((linea = lector.ReadLine()) != null)  {  Console.WriteLine(linea);  }  }  }  catch (Exception e)  {  // Capturar y mostrar cualquier excepción  Console.WriteLine("Ocurrió un error al leer el archivo:");  Console.WriteLine(e.Message);  }  }  } |

Leer todo el fichero:

|  |
| --- |
| using System;  using System.IO;  class Program  {  static void Main()  {  // Especifica la ruta del archivo  string rutaArchivo = "ruta/del/archivo.txt";    try  {  // Leer todo el contenido del archivo en una sola operación  string contenido = File.ReadAllText(rutaArchivo);  Console.WriteLine(contenido);  }  catch (Exception e)  {  // Capturar y mostrar cualquier excepción  Console.WriteLine("Ocurrió un error al leer el archivo:");  Console.WriteLine(e.Message);  }  }  } |

Leer todo el fichero y meterlo en un array:

|  |
| --- |
| using System;  using System.IO;  class Program  {  static void Main()  {  // Especifica la ruta del archivo  string rutaArchivo = "ruta/del/archivo.txt";    try  {  // Leer todas las líneas del archivo y almacenarlas en un array  string[] lineas = File.ReadAllLines(rutaArchivo);    // Recorrer el array y mostrar cada línea  foreach (string linea in lineas)  {  Console.WriteLine(linea);  }  }  catch (Exception e)  {  // Capturar y mostrar cualquier excepción  Console.WriteLine("Ocurrió un error al leer el archivo:");  Console.WriteLine(e.Message);  }  }  } |

**Importante**:

|  |
| --- |
| ⚠️   * **StreamReader** es útil cuando estás leyendo archivos grandes y quieres procesarlos línea por línea sin cargar todo el contenido en la memoria de una vez. * **File.ReadAllText()** y **File.ReadAllLines()** son útiles para archivos pequeños o cuando quieres todo el contenido de una sola vez. |

## Actividad

Crear un archivo txt que contenga, por cada línea, la temperatura de los días de la semana, cargarlo en un array, y mostrar el contenido.

|  |
| --- |
| *mostrar resultados*  Ejemplo de lo que tiene que mostrar:  Lunes: 20  Martes 23  Miércoles: 22  Jueves: 18  Viernes: 17  Sábado: 18  Domingo: 20 |