

Universidad Rafael Landívar
Pensamiento Computacional
Catedrático: Luis Aguilar

ACTIVIDAD 1 SEMANA 17

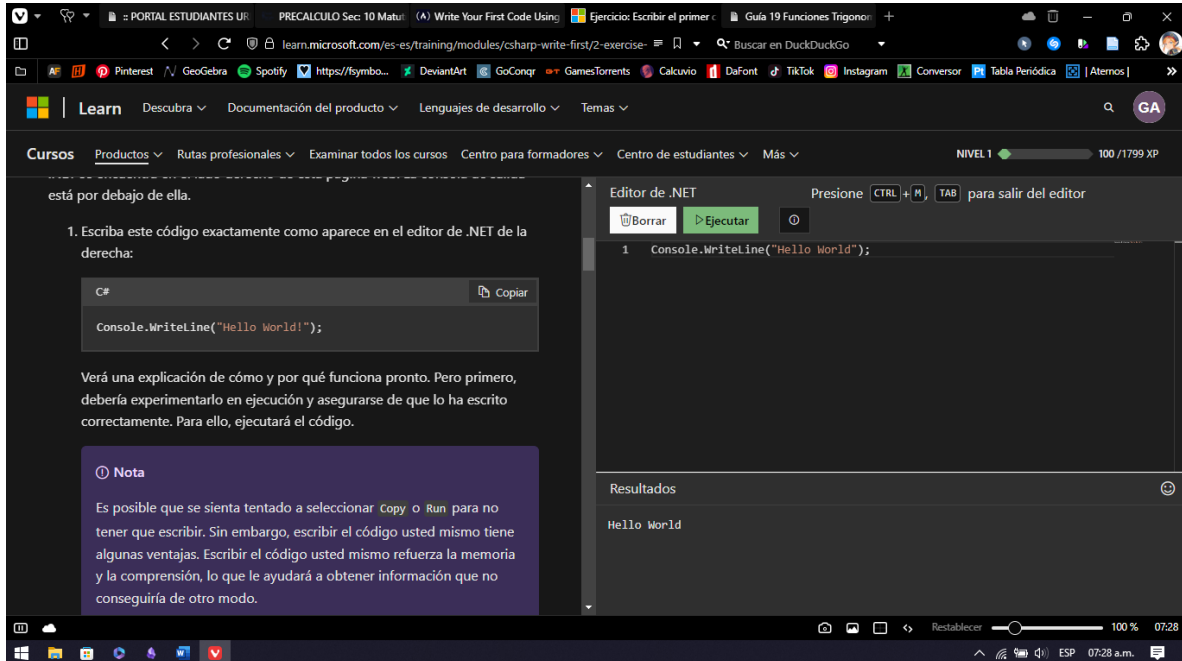
PENSAMIENTO COMPUTACIONAL

Gabriel Alejandro Ajin Izaguirre
Carné: 1184924

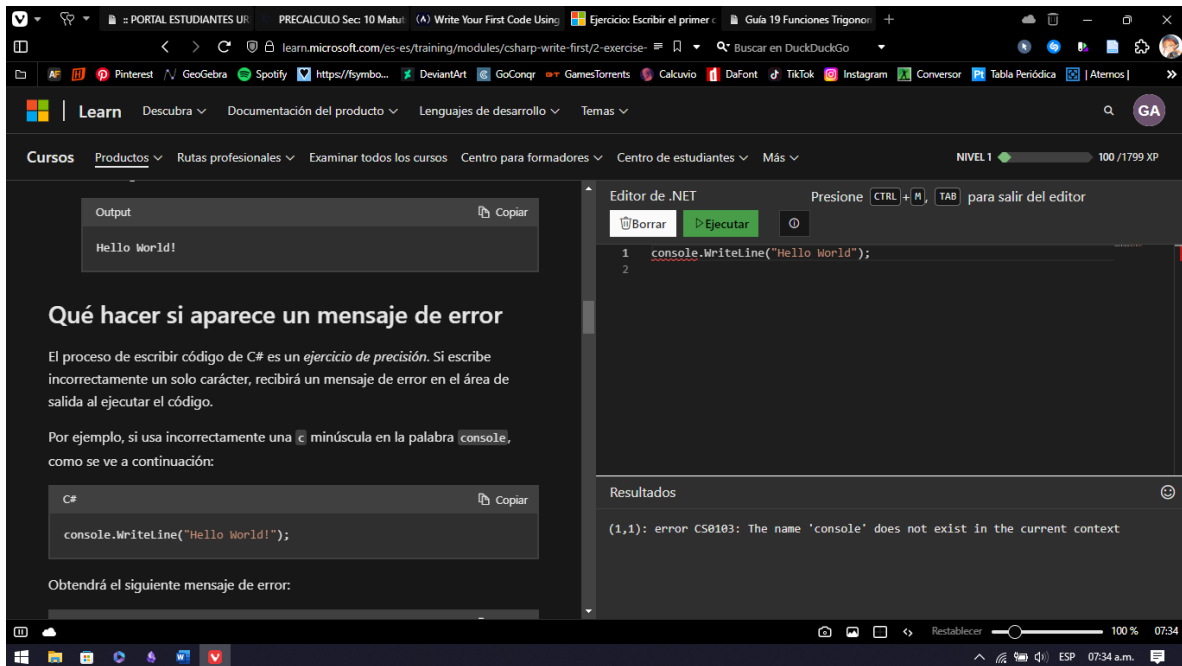
Guatemala 10 de may. de 24

MODULO 1

HELLO WORLD



ERROR DE ESCRITURA EN CONSOLE.WRITE LINE



CONSOLE.WRITE();

The screenshot shows the Microsoft Learn interface for the exercise "Escribir el primer programa en C#". The main content area on the left is titled "La diferencia entre Console.Write y Console.WriteLine". It explains that `Console.WriteLine()` prints a message and moves the cursor to the next line, while `Console.Write()` prints a message on the same line. A code editor on the right, titled "Editor de .NET", contains the following code:

```
1 Console.WriteLine("Hello World");
2 Console.Write("Congratulations!");
3 Console.Write(" ");
4 Console.Write("You wrote your first lines of code.");
```

Below the code editor, the "Resultados" (Results) section shows the output of the program:

```
Hello World
Congratulations! You wrote your first lines of code.
```

The top navigation bar includes links to "Cursos", "Productos", "Rutas profesionales", "Examinar todos los cursos", "Centro para formadores", "Centro de estudiantes", and "Más". The user's progress is shown as "NIVEL 1" with a progress bar at 100% and 1799 XP.

COMPROBACION DE CONOCIMIENTOS

The screenshot shows the Microsoft Learn interface for the exercise "Comprobación de conocimientos". The main content area on the left is titled "Comprobación de conocimientos" and contains a question: "1. ¿Cuál es la diferencia entre Console.Write y Console.WriteLine?". The options are:

- ☐ Console.Write imprime la salida en una nueva línea.
- ☐ Console.WriteLine imprime la salida en una nueva línea.
- ☒ Console.WriteLine anexa una nueva línea después de la salida.

The correct answer is indicated by a green checkmark and the text: "Correcto. Console.WriteLine imprime la salida en la línea existente y anexa una nueva línea después de ella."

Below the question, the "Siguiente unidad: Completar el desafío" (Next unit: Complete the challenge) section is visible, with a "Continuar" button.

The code editor on the right, titled "Editor de .NET", contains the following code:

```
1 Console.WriteLine("Hello World!");
```

Below the code editor, the "Resultados" (Results) section shows the output of the program:

```
Hello World!
```

The top navigation bar includes links to "Cursos", "Productos", "Rutas profesionales", "Examinar todos los cursos", "Centro para formadores", "Centro de estudiantes", and "Preguntas más frecuentes y ayuda". The user's progress is shown as "NIVEL 1" with a progress bar at 700% and 1799 XP.

PORTAL ESTUDIANTES UR | Write Your First Code Using | Introducción - Training | Mi | Completar el desafío - Train

learn.microsoft.com/es-es/training/modules/csharp-write-first/4-challenge

Learn Descubra Documentación del producto Lenguajes de desarrollo Temas

Cursos Productos Rutas profesionales Examinar todos los cursos Centro para formadores Centro de estudiantes Preguntas más frecuentes y ayuda NIVEL 1 800/1799 XP

5 minutos

Los desafíos de codificación en estos módulos reforzarán lo que ha aprendido y le ayudarán a tener más confianza antes de continuar.

Desafío: Escriba código en el editor de .NET para mostrar dos mensajes

1. Seleccione todo el código en el editor de .NET y presione la tecla **Suprimir** o **Retroceso** para eliminarlo.
2. Escriba código que genere la siguiente salida:

Output

```
This is the first line.  
This is the second line.
```

En la unidad anterior, ha aprendido a mostrar un mensaje en una sola línea de código y a mostrar un mensaje con varias líneas de código. Use ambas técnicas en este desafío. No importa qué técnica aplique a cada línea ni de cuántas maneras divida uno de los mensajes en varias líneas de código; usted decide.

Independientemente de cómo lo haga, el código debe generar la salida especificada.

Editor de .NET Presione **CTRL + M**, **TAB** para salir del editor **Borrar** **Ejecutar**

```
1 Console.WriteLine ("This is the first line.");  
2 Console.WriteLine ("This is the second line.");
```

Resultados

```
This is the first line.  
This is the second line.
```

Restablecer 90% 07:59


PORTAL ESTUDIANTES UR | Write Your First Code Using | Introducción - Training | Mi | Completar el desafío - Train

www.freecodecamp.org/learn/foundational-c-sharp-with-microsoft/write-...

freeCodeCamp

Search 10,700+ tutorials

You've got the touch!



Write Your First Code Using C#

14% complete

Submit and go to next challenge (Ctrl + Enter)

Ask for Help

Restablecer 100% 07:57

MODULO 2

LITERALES ENTEROS

The screenshot shows a Microsoft Learn page titled 'LITERALES ENTEROS'. The page content includes a paragraph explaining integer literals in C#, followed by a numbered list with one step: '1. Agregue la siguiente línea de código en el editor de código:'. Below this is a code block containing `Console.WriteLine(123);`. A second step says '2. Presione el botón verde Ejecutar para ejecutar el código. Deberíamos ver el resultado siguiente en la consola de salida:'. Below this is an 'Output' box showing '123'. To the right is a C# code editor with the same line of code. Below the editor is a 'Resultados' box also showing '123'. The page title 'LITERALES ENTEROS' is at the top. The bottom of the page has a navigation bar with 'Cursos', 'Productos', 'Rutas profesionales', 'Examinar todos los cursos', 'Centro para formadores', 'Centro de estudiantes', and 'Preguntas más frecuentes y ayuda'. The user's profile is 'GA'.

Learn Descubra Documentación del producto Lenguajes de desarrollo Temas

Cursos Productos Rutas profesionales Examinar todos los cursos Centro para formadores Centro de estudiantes Preguntas más frecuentes y ayuda NIVEL 1 1700 / 1799 XP

Presione CTRL + M TAB para salir del editor Borrar Ejecutar

```
1 Console.WriteLine(123);
```

Resultados

123

Uso de literales de punto flotante

Un número de punto flotante es uno que contiene decimales, por ejemplo, 3.14159. C# admite tres tipos de datos para representar números decimales: `float`, `double` y `decimal`. Cada tipo admite distintos grados de precisión.

SUFIJOS LITERALES

The screenshot shows a Microsoft Learn page titled 'SUFIJOS LITERALES'. The page content includes a paragraph explaining floating-point literals in C#, followed by a numbered list with two steps. Step 1 says '1. Presione el botón verde Ejecutar para ejecutar el código. Deberíamos ver el resultado siguiente en la consola de salida:'. Below this is an 'Output' box showing '0.25'. Step 2 says '2. Agregue la siguiente línea de código en el editor de código:'. Below this is a code block containing `Console.WriteLine(0.25F);`. To the right is a C# code editor with the same line of code. Below the editor is a 'Resultados' box also showing '0.25'. The page title 'SUFIJOS LITERALES' is at the top. The bottom of the page has a navigation bar with 'Cursos', 'Productos', 'Rutas profesionales', 'Examinar todos los cursos', 'Centro para formadores', 'Centro de estudiantes', and 'Preguntas más frecuentes y ayuda'. The user's profile is 'GA'.

Learn Descubra Documentación del producto Lenguajes de desarrollo Temas

Cursos Productos Rutas profesionales Examinar todos los cursos Centro para formadores Centro de estudiantes Preguntas más frecuentes y ayuda NIVEL 1 1700 / 1799 XP

Presione CTRL + M TAB para salir del editor Borrar Ejecutar

```
1 Console.WriteLine(0.25F);
```

Resultados

0.25

Para crear un literal decimal `float`, anexe la letra `F` después del número. En este contexto, `F` se denomina *sufijo literal*. El sufijo literal le indica al compilador que queremos trabajar con un valor de tipo `float`. Puede usar `f` en minúsculas o `F` en mayúsculas como sufijo literal de `float`.

2. Presione el botón verde Ejecutar para ejecutar el código. Deberíamos ver el resultado siguiente en la consola de salida:

Output

0.25

Tenga en cuenta que el tipo de datos `float` es el menos preciso, por lo que es mejor usar este tipo de datos para los valores fraccionarios fijos y así evitar errores de cálculo imprevistos.

3. Agregue la siguiente línea de código en el editor de código:

```
C#
```

```
Console.WriteLine(0.25F);
```

12.39816

Uso de literales booleanos

Si queremos imprimir un valor que represente `true` o `false`, podemos usar un literal `bool`.

El término `bool` es la abreviatura de *booleano*. En C#, se conoce oficialmente como "bool", pero a menudo los desarrolladores usan el término "booleano".

1. Agregue las siguientes línea de código en el editor de código:

```
C#  
Console.WriteLine(true);  
Console.WriteLine(false);
```

2. Presione el botón verde Ejecutar para ejecutar el código. Deberíamos ver el resultado siguiente en la consola de salida:

```
Output  
True  
False
```

Editor de .NET Presione **CTRL + M, TAB** para salir del editor **Borrar** **Ejecutar**

```
1 Console.WriteLine(0.25f);  
2 Console.WriteLine(2.625);  
3 Console.WriteLine(12.39816m);
```

Resultados

```
0.25  
2.625  
12.39816
```

BOOLEANOS

12.39816

Uso de literales booleanos

Si queremos imprimir un valor que represente `true` o `false`, podemos usar un literal `bool`.

El término `bool` es la abreviatura de *booleano*. En C#, se conoce oficialmente como "bool", pero a menudo los desarrolladores usan el término "booleano".

1. Agregue las siguientes línea de código en el editor de código:

```
C#  
Console.WriteLine(true);  
Console.WriteLine(false);
```

2. Presione el botón verde Ejecutar para ejecutar el código. Deberíamos ver el resultado siguiente en la consola de salida:

```
Output  
True  
False
```

Editor de .NET Presione **CTRL + M, TAB** para salir del editor **Borrar** **Ejecutar**

```
1 Console.WriteLine(true);  
2 Console.WriteLine(false);
```

Resultados

```
True  
False
```

DIFERENCIA TIPOS DE DATOS

y códigos postales, es preferible usar un tipo de datos `string` al trabajar con ellos.

Lo mismo puede decirse de `bool`. Si necesita trabajar con las palabras `"true"` y `"false"` en la aplicación, debe usar un valor `string`. Pero si necesita trabajar con el concepto de `true` o `false` al realizar una evaluación, use un valor `bool`.

Es importante saber que estos valores pueden ser similares a sus equivalentes de literales de cadena. Es decir, puede pensar que estas instrucciones son iguales:

```
C#
Console.WriteLine("123");
Console.WriteLine(123);
Console.WriteLine("true");
Console.WriteLine(true);
```

Sin embargo, solo es similar la salida que se muestra. El hecho es que los tipos de cosas que puede hacer con el valor subyacente `int` o `bool` son diferentes a los de su equivalente `string`.

Resumen

Editor de .NET Presione **CTRL + M**, **TAB** para salir del editor **Borrar** **Ejecutar**

```
1 Console.WriteLine("123");
2 Console.WriteLine(123);
3
4 Console.WriteLine("true");
5 Console.WriteLine(true);
```

Resultados

```
123
123
true
True
```

DECLARACION DE VARIABLES

Anterior Unidad 3 de 9 Siguiendo >

Declaración de variables

6 minutos

Un literal es *literalmente* un valor codificado de forma rígida. Los valores codificados de forma rígida son valores que son constantes e inalterables durante la ejecución del programa. Sin embargo, la mayoría de las aplicaciones precisan que trabaje con valores de los que no se sabe mucho de antemano. Es decir, tiene que trabajar con datos que provienen de usuarios, de archivos o de la red.

Cuando necesite trabajar con datos que no están codificados de forma rígida, declarará una variable.

¿Qué es una variable?

Una **variable** es un contenedor para almacenar un tipo de valor. Las variables son importantes porque sus valores pueden cambiar o variar durante la ejecución de un programa. Las variables se pueden asignar, leer y cambiar. Las variables se usan para

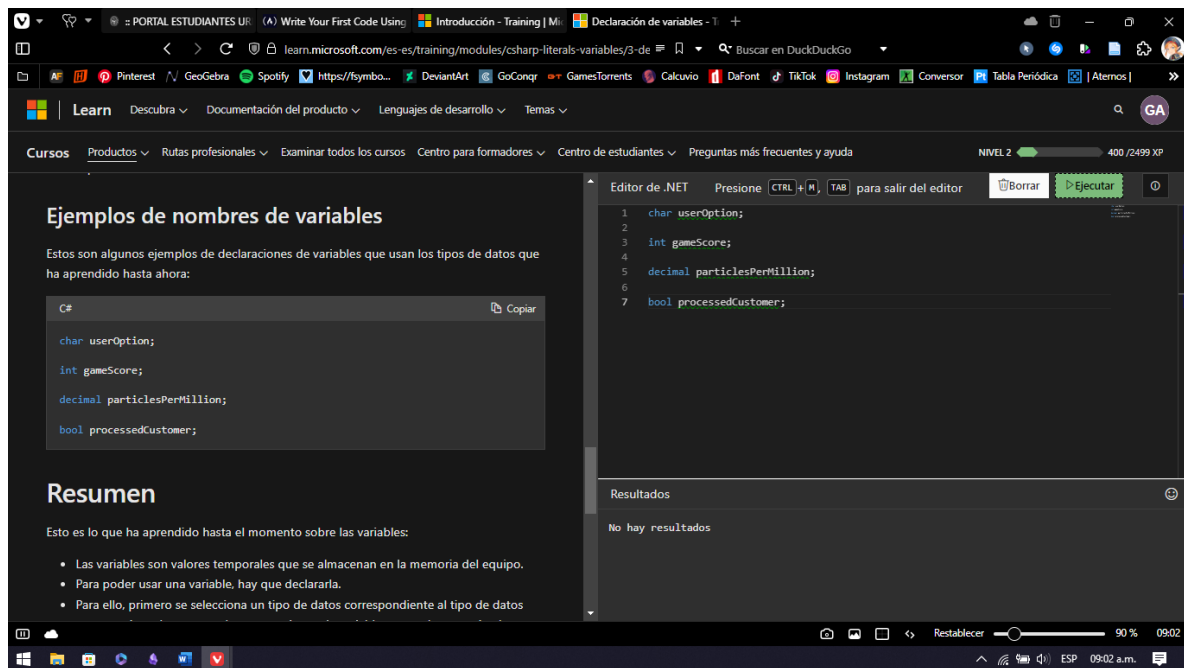
Editor de .NET Presione **CTRL + M**, **TAB** para salir del editor **Borrar** **Ejecutar**

```
1 string firstName;
```

Resultados

No hay resultados

EJEMPLOS DE VARIABLES



Ejemplos de nombres de variables

Estos son algunos ejemplos de declaraciones de variables que usan los tipos de datos que ha aprendido hasta ahora:

```
C#  
char userOption;  
int gameScore;  
decimal particlesPerMillion;  
bool processedCustomer;
```

Resumen

Esto es lo que ha aprendido hasta el momento sobre las variables:

- Las variables son valores temporales que se almacenan en la memoria del equipo.
- Para poder usar una variable, hay que declararla.
- Para ello, primero se selecciona un tipo de datos correspondiente al tipo de datos

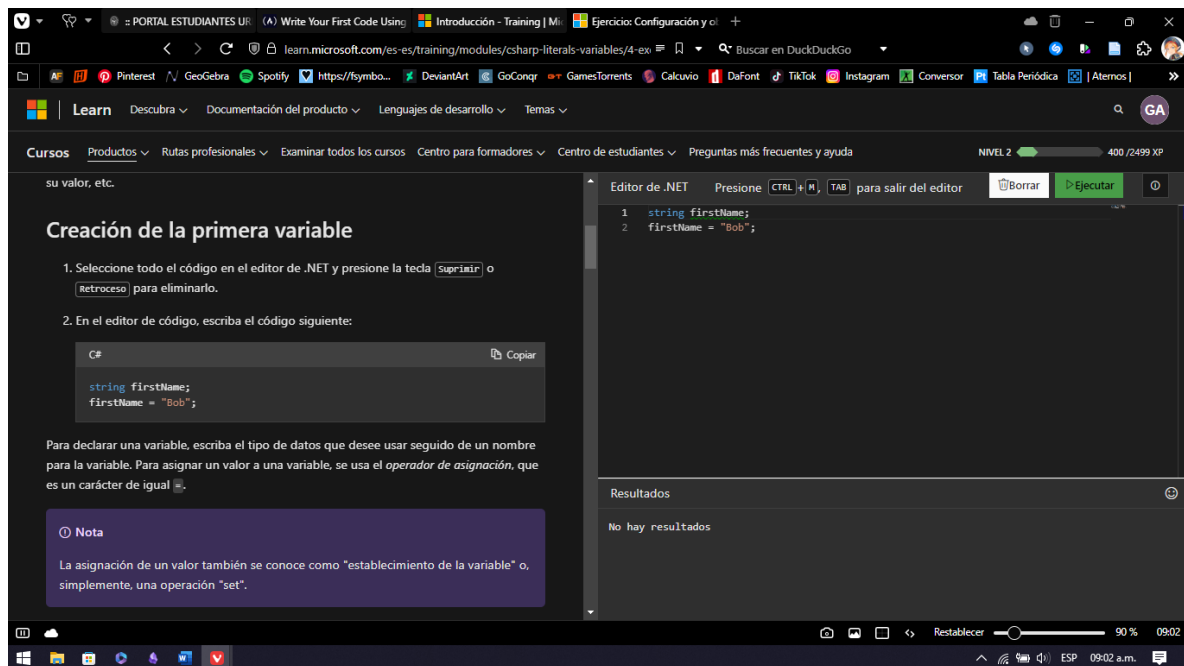
Editor de .NET

```
1 char userOption;  
2  
3 int gameScore;  
4  
5 decimal particlesPerMillion;  
6  
7 bool processedCustomer;
```

Resultados

No hay resultados

CREACION DE LA PRIMERA VARIABLE



su valor, etc.

Creación de la primera variable

- Seleccione todo el código en el editor de .NET y presione la tecla **Suprimir** o **Retroceso** para eliminarlo.
- En el editor de código, escriba el código siguiente:

```
C#  
string firstName;  
firstName = "Bob";
```

Para declarar una variable, escriba el tipo de datos que desee usar seguido de un nombre para la variable. Para asignar un valor a una variable, se usa el *operador de asignación*, que es un carácter de igual `=`.

Nota

La asignación de un valor también se conoce como "establecimiento de la variable" o, simplemente, una operación "set".

Editor de .NET

```
1 string firstName;  
2 firstName = "Bob";
```

Resultados

No hay resultados

ASIGNACION INCORRECTA DE LA VARIABLE

CursosProductosRutas profesionalesExaminar todos los cursosCentro para formadoresCentro de estudiantesPreguntas más frecuentes y ayuda

NIVEL 2400 / 2499 XP

lado izquierdo del operador de asignación. Si el orden se revierte, se confunde al compilador de C#.

1. Modifique el código escrito para que coincida con el siguiente:

```
C# Copiarstring firstName;
"Bob" = firstName;
```

2. Ahora, ejecute el código. Se ve el error siguiente en la consola de salida:

```
Output Copiar(2,1): error CS0131: The left-hand side of an assignment must be a variable,
```

Asignación de un valor del tipo de datos incorrecto a la variable

Hemos aprendido que C# se ha diseñado para aplicar tipos. Cuando trabajamos con variables, *aplicar tipos* significa que no se puede asignar un valor de un tipo de datos a una variable declarada para contener otro distinto.

1. Modifique el código escrito para que coincida con el siguiente:

```
C# Copiarint firstName;
firstName = "Bob";
```

RECUPERACIÓN DE UN VALOR ALMACENADO EN LA VARIABLE

Learn

Descubra

Documentación del producto

Lenguajes de desarrollo

Temas

Cursos

Productos

Rutas profesionales

Examinar todos los cursos

Centro para formadores

Centro de estudiantes

Preguntas más frecuentes y ayuda

NIVEL 2

400 / 2499 XP

Más adelante encontraremos más información sobre la conversión de tipos implícita y explícita. Por ahora, recuerde que una variable solo puede contener valores que coincidan con su tipo de datos especificado.

Recuperación de un valor almacenado en la variable

Para recuperar un valor de una variable, simplemente se usa el nombre de la variable. En este ejemplo se establece el valor de una variable; después se recupera ese valor y se muestra en la consola.

1. Modifique el código escrito para que coincida con el siguiente:

C#

```
string firstName;  
firstName = "Bob";  
Console.WriteLine(firstName);
```

2. Ahora, ejecute el código. Se ve el resultado siguiente en la consola de salida:

Output

Bob

La recuperación de un valor de una variable también se conoce como "obtención de la variable" o, simplemente, una operación "get".

A medida que escribe líneas de código, verá que el compilador comprueba el código y

Editor de .NET

Presione **CTRL + M** para salir del editor

Borrar

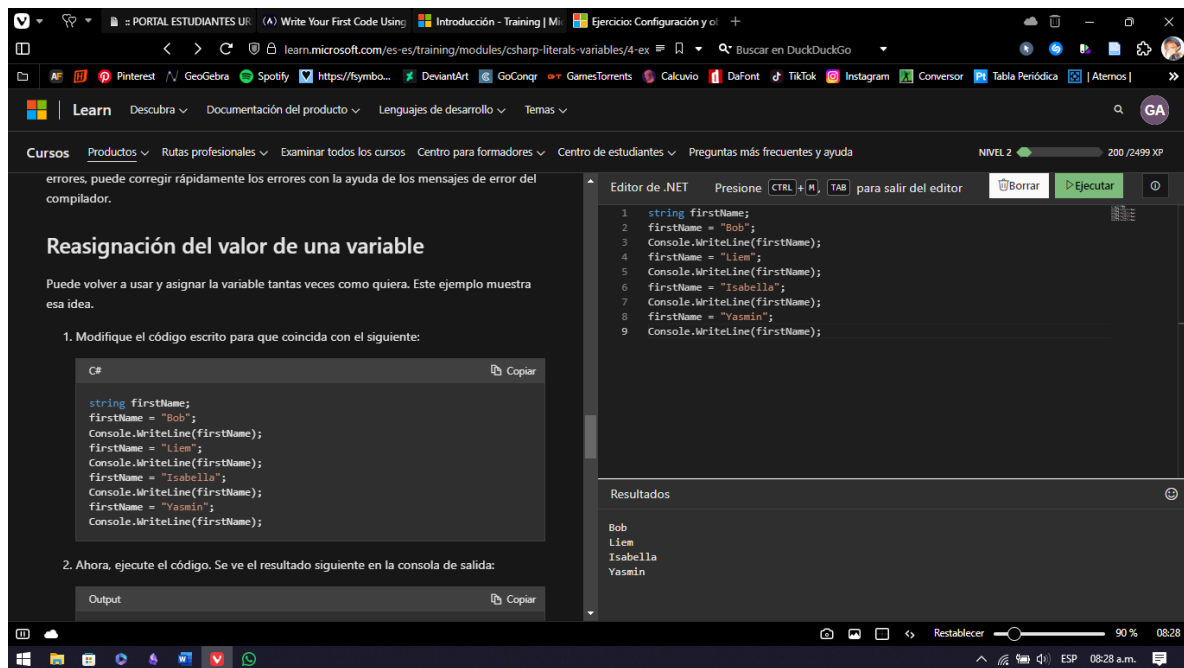
Ejecutar

```
1 string firstName;  
2 firstName = "Bob";  
3 Console.WriteLine(firstName);
```

Resultados

Bob

REASIGNACIÓN DEL VALOR DE UNA VARIABLE



errores, puede corregir rápidamente los errores con la ayuda de los mensajes de error del compilador.

Reasignación del valor de una variable

Puede volver a usar y asignar la variable tantas veces como quiera. Este ejemplo muestra esa idea.

1. Modifique el código escrito para que coincida con el siguiente:

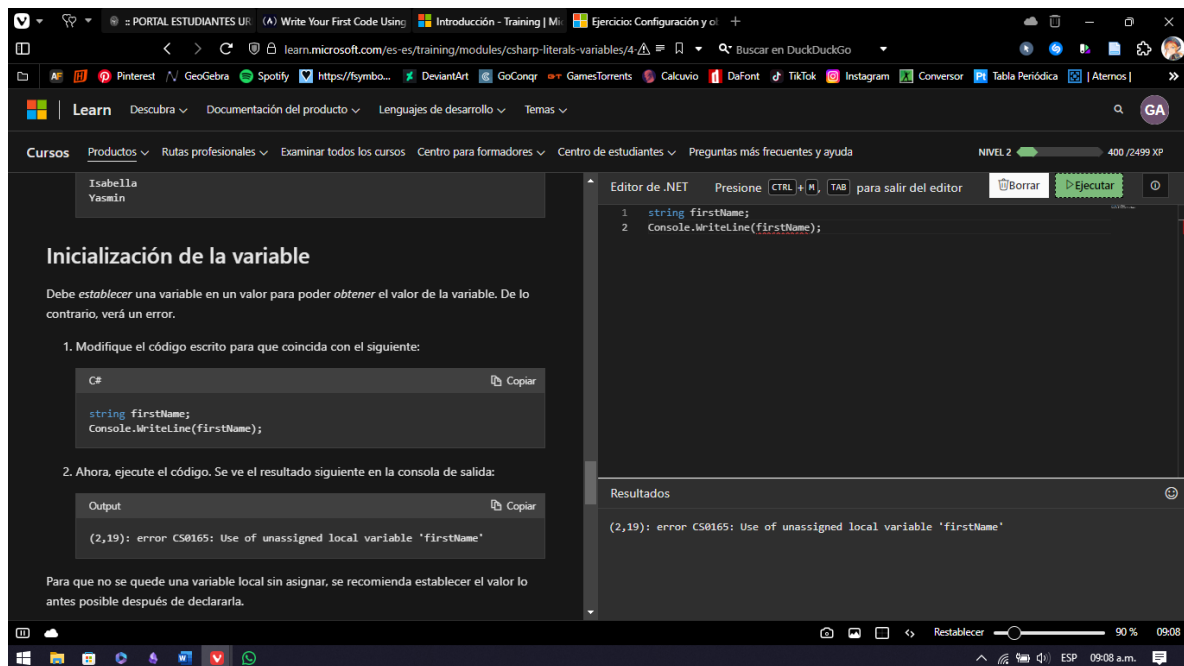
```
C#  
string firstName;  
firstName = "Bob";  
Console.WriteLine(firstName);  
firstName = "Liem";  
Console.WriteLine(firstName);  
firstName = "Isabella";  
Console.WriteLine(firstName);  
firstName = "Yasmin";  
Console.WriteLine(firstName);
```

2. Ahora, ejecute el código. Se ve el resultado siguiente en la consola de salida:

Output

```
Bob  
Liem  
Isabella  
Yasmin
```

INICIALIZACION DE LA VARIABLE



Inicialización de la variable

Debe *establecer* una variable en un valor para poder *obtener* el valor de la variable. De lo contrario, verá un error.

1. Modifique el código escrito para que coincida con el siguiente:

```
C#  
string firstName;  
Console.WriteLine(firstName);
```

2. Ahora, ejecute el código. Se ve el resultado siguiente en la consola de salida:

Output

```
(2,19): error CS0165: Use of unassigned local variable 'firstName'
```

Para que no se quede una variable local sin asignar, se recomienda establecer el valor lo antes posible después de declararla.

COMPROBACION

The screenshot shows a web browser window displaying a Microsoft Learn challenge. The page title is "Completar el desafío - Train". The challenge is titled "2. Almacene los siguientes valores en variables:" and lists three items: Bob, 3, and 34.4. It provides instructions on variable naming and data types, and asks the user to write C# code in the Visual Studio .NET editor. The code editor shows the following code:

```
1 string nombre= "Bob";
2 int numero= 3;
3 double numero2= 34.4;
4
5 Console.WriteLine($"Hello, {nombre}! You have {numero} messages in your inbox. The temperature is {numero2} celsius.");
6
```

The output window shows the result: "Hello, Bob! You have 3 messages in your inbox. The temperature is 34.4 celsius." The challenge is part of a course titled "Introducción - Training | Mi" and is worth 400 XP.

CONOCIMIENTOS PREVIOS

The screenshot shows a web browser window displaying a Microsoft Learn knowledge check. The page title is "Prueba de conocimientos -". The challenge is titled "Prueba de conocimientos" and is worth 200 XP. It is part of a course titled "Introducción - Training | Mi". The challenge is titled "Comprobación de conocimientos" and contains two questions:

- ¿Cuáles de los siguientes nombres de variable se deben evitar? *
- Cuál es el problema de esta línea de código: `var message; *`

The first question has three options: `$DATA`, `registrationComplete`, and `flag`. The second question has one option: `var` no es un tipo de datos.

