

Universidad Rafael Landivar.
Facultad de Ingeniería.
Ingeniería en Informática y Sistemas.
Docente: Ing. Luis Aguilar

BOOTCAMP

Estudiante: Choi Galindo, Ho Sung
Carné: 2139224

Guatemala, 10 de mayo del año 2024

Foundational C# with Microsoft en FreeCodeCamp

1) Write you first C# code

This screenshot shows the first step of a C# tutorial on the Microsoft Learn platform. The browser address bar shows the URL: `learn.microsoft.com/en-us/training/modules/csharp-write-first/2-exercise-hello-world`. The page content includes:

- A code editor with the following C# code:

```
// Console.WriteLine("Hello World!");
```
- Text explaining that `//` is a comment prefix and that code comments are helpful for documentation.
- Step 2: "Add new lines of code to match the following code snippet:" followed by a code block:

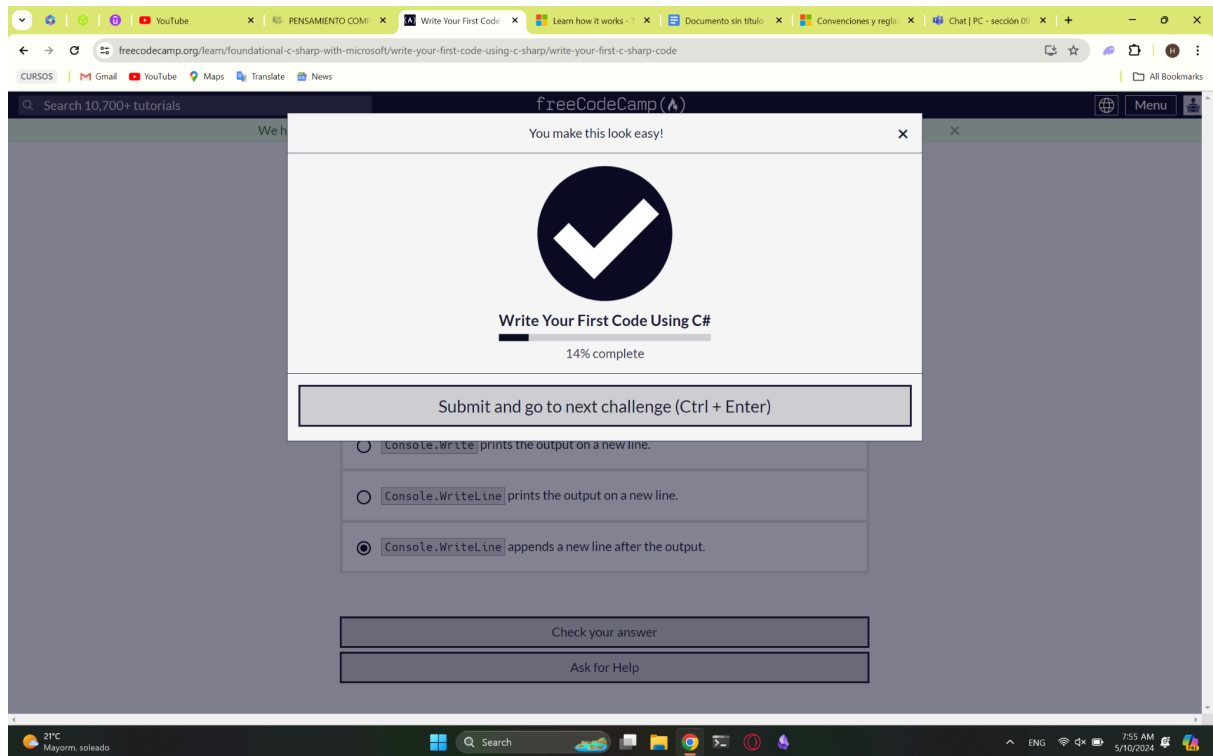
```
C#
Console.WriteLine("Congratulations!");
Console.Write(" ");
Console.WriteLine("You wrote your first lines of code.");
```
- Step 3: "Press the green Run button again. This time, you should get the following output." followed by an output box showing: "Congratulations! You wrote your first lines of code."
- A section titled "The difference between Console.Write and Console.WriteLine" explaining that `Console.WriteLine()` adds a new line.

The right side of the screen shows a ".NET Editor" with the same code and an "Output" window displaying "Hello World!".

This screenshot shows the second step of the C# tutorial, titled "Learn how it works". The browser address bar shows the URL: `learn.microsoft.com/en-us/training/modules/csharp-write-first/3-how-it-works`. The page content includes:

- A breadcrumb trail: "Learn / Training / Browse / Write your first C# code /".
- A section titled "Learn how it works" with a duration of "6 minutes".
- Text explaining that understanding how code works involves stepping back and thinking about the programming language.
- A section titled "What is a programming language?" explaining that programming languages like C# let you write instructions for the computer to carry out. It notes that each language has its own syntax but shares many concepts.
- Text explaining that a developer can update and change the code, but the computer can't understand the code until it is compiled into a format the computer can understand.
- A section titled "What is compilation?" explaining that a compiler converts source code into a format the CPU can execute.
- Text explaining why code needs to be compiled: "Although most programming languages seem cryptic at first, they".

The right side of the screen shows the ".NET Editor" with the same code as the previous step and an "Output" window displaying "Hello World!" and "Congratulations! You wrote your first lines of code."



2) Store and retrieve values using literal and variable values in C#

Output

```
123
```

Use floating-point literals

A floating-point number is a number that contains a decimal, for example 3.14159. C# supports three data types to represent decimal numbers: `float`, `double`, and `decimal`. Each type supports varying degrees of precision.

Float Type	Precision
<code>float</code>	~6-9 digits
<code>double</code>	~15-17 digits
<code>decimal</code>	28-29 digits

Here, precision reflects the number of digits past the decimal that are accurate.

1. Add the following line of code in the code editor:

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

To create a `float` literal, append the letter `F` after the number. In this context, the `F` is called a *literal suffix*. The literal suffix tells the compiler you wish to work with a value of `float` type. You can use either a lower-case `f` or upper-case `F` as the literal suffix for a `float`.

2. Press the green Run button to run your code. You should see the following result in the output console:

Output

```
123123132132.132
```

Learn

Discover

Product documentation

Development languages

Topics

Training

Products

Career Paths

Browse all training

Educator Center

Student Hub

FAQ & Help

LEVEL 1

300 / 1799 XP

Output

Copy

2.625

5. Add the following line of code in the code editor:

C#

Copy

Console.WriteLine(12.39816m);

To create a decimal literal, append the letter `m` after the number. In this context, the `m` is called a *literal suffix*. The literal suffix tells the compiler you wish to work with a value of `decimal` type. You can use either a lower-case `m` or upper-case `M` as the literal suffix for a `decimal`.

6. Press the green Run button to run your code. You should see the following result in the output console:

Output

Copy

12.39816

Use Boolean literals

If you wanted to print a value representing either `true` or `false`, you could use a `bool` literal.

The term `bool` is short for *Boolean*. In C#, they're officially referred to as "bool", but often developers use the term "Boolean".

1. Add the following lines of code in the code editor:

.NET Editor

Press `CTRL + M`, `TAB` to exit the editor

Clear

Run

1 Console.WriteLine(0.25f);

2 Console.WriteLine(2.625);

3 Console.WriteLine(12.39816m);

Output

Copy

0.25

2.625

12.39816

21°C

Viento

Search

ENG

8:16 AM

5/10/2024

Learn

Discover

Product documentation

Development languages

Topics

Training

Products

Career Paths

Browse all training

Educator Center

Student Hub

FAQ & Help

LEVEL 1

300 / 1799 XP

When you need to perform mathematical calculations on phone numbers and postal codes, you should prefer to use a `string` data type when working with them.

The same can be said of `bool`. If you need to work with the words `"true"` and `"false"` in your application, you would use a `string`. However, if you need to work with the concept of `true` or `false` when performing an evaluation, you use a `bool`.

It's important to know that these values may look like their string literal equivalents. In other words, you may think these statements are the same:

C#

Copy

Console.WriteLine("123");

Console.WriteLine(123);

Console.WriteLine("true");

Console.WriteLine(true);

However, it's only the displayed output that appears to be similar. The fact is that the kinds of things you can do with the underlying `int` or `bool` will be different than their `string` equivalent.

Recap

The main takeaway is that there are many data types, but you'll focus on just a few for now:

- `string` for words, phrases, or any alphanumeric data for presentation, not calculation
- `char` for a single alphanumeric character
- `int` for a whole number
- `decimal` for a number with a fractional component
- `bool` for a `true/false` value

.NET Editor

Press `CTRL + M`, `TAB` to exit the editor

Clear

Run

1 Console.WriteLine("123");

2 Console.WriteLine(123);

3

4 Console.WriteLine("true");

5 Console.WriteLine(true);

Output

Copy

True

False

21°C

Soleado

Search

ENG

8:21 AM

5/10/2024

learn.microsoft.com/en-us/training/modules/csharp-literals-variables/3-declaring-variables

CURSOS | Gmail | YouTube | Maps | Translate | News | All Bookmarks

Learn Discover Product documentation Development languages Topics

Training Products Career Paths Browse all training Educator Center Student Hub FAQ & Help

LEVEL 1 400 / 1799 XP

- Variable names are case-sensitive, meaning that `string Value;` and `string value;` are two different variables.
- Variable names must not be a C# keyword. For example, you cannot use the following variable declarations: `decimal decimal;` or `string string;`.

There are coding conventions that help keep variables readable and easy to identify. As you develop larger applications, these coding conventions can help you keep track of variables among other text.

Here are some coding conventions for variables:

- Variable names should use camel case, which is a style of writing that uses a lower-case letter at the beginning of the first word and an upper-case letter at the beginning of each subsequent word. For example, `string thisIsCamelCase;`.
- Variable names should begin with an alphabetical letter. Developers use the underscore for a special purpose, so try to not use that for now.
- Variable names should be descriptive and meaningful in your app. Choose a name for your variable that represents the kind of data it will hold.
- Variable names should be one or more entire words appended together. Don't use contractions or abbreviations because the name of the variable (and therefore, its purpose) may be unclear to others who are reading your code.
- Variable names shouldn't include the data type of the variable. You might see some advice to use a style like `string strValue;`. That advice is no longer current.

The example `string firstName;` follows all of these rules and conventions, assuming you want to use this variable to store data that represents someone's first name.

Variable name examples

Here's a few examples of variable declarations using the data types you learned about thus far:

```
C#
```

Copy

.NET Editor Press CTRL + R, TAB to exit the editor Clear Run

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

Output

Choi

21°C Soleado


freecodecamp.org/learn/foundational-c-sharp-with-microsoft/write-your-first-code-using-c-sharp/store-and-retrieve-data-using-literal-and-variable-values-in-c-sharp

CURSOS | Gmail | YouTube | Maps | Translate | News | All Bookmarks

Search 10,700+ tutorials

freeCodeCamp (A)

You're an all star!



Write Your First Code Using C#

29% complete

Submit and go to next challenge (Ctrl + Enter)

☐ `int x = 12.3m;`

☒ `decimal x = 12.3m;`

☐ `bool x = 'False';`

Check your answer

Ask for Help

21°C Soleado