# Consideraciones importantes

C# es “Case Sensitive” es decir distingue entre mayúsculas y minúsculas entre cada instrucción.

Todas las instrucciones deben terminar con “;”.

Al igual que en todos los lenguajes cada variable, método, constante, clase y similares deben tener un nombre único.

Cabe recalcar que para que se ejecute un método este tiene que ser llamado desde le Main, de lo contrario no ocurrirá nada.

Por último cada aplicación se debe guiar por la siguiente estructura básica.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

# Tipos de datos

Están los números enteros, los Reales(decimales), string y booleanos.

Los enteros se comprenden en dos grandes grupos:

sin signo:

* Byte
* Ushort
* Unit
* Ulong

Con signo:

* Sbyte
* Short
* Int
* Long

la capacidad de almacenamiento de cada tipo de dato es mayor a medida que se va bajando.

Los reales(decimales) solamente son 3 float, double y decimal, al igual que los enteros su capacidad de almacenamiento crece.

Los booleanos solo pueden tomar como únicos valores True o False, estos son comúnmente utilizados para estructuras de control de flujo.

# Instrucciones generales

## Comentarios

En C# existen dos formas de realizar comentarios:

“**//**” Que sirve solamente para comentar en una sola línea

“**/\***” Para abrir un comentario de una o más líneas y para cerrar “**\*/**”.

## Declaración de variables

Se pueden declarar 4 grandes tipos de variable sal igual que se menciono anteriormente en la sección de tipos de datos estas son enteros, decimales y booleanos.

Declaración de cada tipo de variable:

Imagen que contiene captura de pantalla

Descripción generada automáticamente

En la imagen se ven dos tipos que se se mencionaron antes y estos son las de tipo “const” y “var”, la variable de tipo “const” es solo una constante y como indica su valor no puede ser cambiado, y la variable de tipo “var” almacena cualquier tipo de dato pero este después no puede ser cambiado.

En C# para poder usar una variable esta debe estar iniciada, porque de lo contrario arroja error, existe dos formas de poder realizar esto declarar la variable y posteriormente más abajo en el programa usarla o declarar e iniciar la variable en la misma línea.

Las variables se pueden declarar tanto dentro de la clase como en el Main, es mas practico declarar las variables en la clase ya que existen ocasiones en las que son necesarias para varios métodos u otras situaciones.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

## Operadores aritméticos

Ya que vimos la declaración de los distintos tipos de variables y los tipos de datos veamos los operadores:

|  |  |  |
| --- | --- | --- |
| Operación | Símbolo | Uso |
| Suma | + | Números y string |
| Resta | - | Números y string |
| Multiplicación | \* | Números |
| División | / | Números |
| Modulo | % | Números |
| Incremento | ++ | Números |
| Decremento | -- | Números |

***Nota al realizar divisiones en C# entre números enteros esta siempre será entero independiente que el resultado sea decimal.***

***Imagen que contiene captura de pantalla

Descripción generada automáticamente***

***Con este ejemplo se muestra que la división entre 2 enteros con parte decimal esta última es truncada. (entero/entero = entero; entero/decimal = decimal)***

## Mostrar y pedir datos por pantalla al usuario

Para mostrar datos al usuario usamos la instrucción:

Console.ReadLine();

Esta no lleva ningún tipo de contenido entre los paréntesis.

Este método cada valor recibido por el usuario es almacenado como string, pero hay ocasiones en la que se le pide un numero al usuario y es necesario transformarlo. Esto se hace con le instrucción Parse(), pero esta necesita que se le especifique el tipo de numero y la sintaxis es esta

tipoDeNumero variable = tipoDeNumero.Parse(Console.ReadLine());

Console.WriteLine(“lo que queremos mostrar”);

Para pedir valores al usuario es con la instrucción:

### Formas de concatenación

Imagen que contiene captura de pantalla

Descripción generada automáticamente

# Métodos

En C# no diferencia entre métodos y funciones en este lenguaje ambas realizan la misma tarea o sentencias, estos no se ejecutan hasta que son llamados desde el Main del programa

Sintaxis de un método

Static tipoDeDato nombreDelMetodo (parámetros)

{

Cuerpo del metodo

}

Imagen que contiene captura de pantalla, texto

Descripción generada automáticamente

Todos los métodos deben llevar la palabra reservada “static” no ocurrirá nada más bien ocurrirá un error de tipo referencia a objeto, esto se explica de mejor manera en POO(Programación Orientada a Objetos), aunque de momento no sepamos que significa debemos colocarla en los métodos.

Para llamar un método solo hay escribir su numbre en el Main y si el método recibe parámetros entregarle los parámetros en el paréntesis.

También existe la posibilidad de tener una ***sobrecarga de métodos***, para que puedan existir estos deben tener distinta cantidad de parámetros o devolver distintos tipos de datos.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Adicionalmente se le pueden definir parámetros opcionales a los métodos, estos siempre van al final de los obligatorios.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

# Control de flujos

El controlador de flujos su utilidad radica en que le sirve al programa para tomar decisiones por su cuenta.

## operadores lógicos y de comparación

|  |  |  |
| --- | --- | --- |
| Nombre | Símbolo | Uso |
| Negación | **!** | Invierte los valores booleanos |
| Igual que | **==** | Pega pal pablo |
| Diferente que | **!=** | Pega pal pablo |
| Menor que | **<** | Pega pal pablo |
| Menor igual | **<=** | Pega pal pablo |
| Mayor que | **>** | Pega pal pablo |
| Mayor igual | **>=** | Pega pal pablo |
| Y lógico | **&&** | Pega pal pablo |
| O lógico | **||** | Pega pal pablo |

## Condicional if

Sintaxis

If (condición)

{

Código a ejecutar si la condición es True

}

Else if (condición)

{

Código a ejecutarse en caso de que la condición anterior sea False y esta True

Else

{

Código a ejecutar si las condiciones anteriores son False

}

Ejemplo

Imagen que contiene captura de pantalla, mapa

Descripción generada automáticamente

## Bucles

Los bucles son muy convenientes a la hora de programar, ya que en ocasiones se presentan situaciones en las que hay que repetir ciertas líneas de código mas de una vez y por esto están los bucles por que de lo contrario habría que repetir n veces las mismas líneas de código y esto es altamente ineficiente.

While (condición)

{

Código a repetirse

}

do{

código a repetir

} while (condición);

While

Siempre ejecutara las líneas que están dentro de el siempre y cuando la condición sea verdadera.

Su funcionamiento es igual que en python no presenta ninguna diferencia.

do while

asegura que siempre por lo menos una vez entrara al ciclo aunque la condición la primera vez que entre sea falsa

for (iniciación; condición; incremento/decremento)

{

Código a repetir

}

for (int i = 0; i <=8;i++)

{

Código a reptir

}

Foreach (iterador in nombreArray)

{

Código a repetir

}

For

Es un bucle que nos sirve cuando sabemos cuántos ciclos se realizaran

Sintaxis

Foreach

Tiene la misma estructura que el for en Python cundo se trabaja con listas o diccionarios por dar algunos ejemplos

Clases .NET

Math

<https://docs.microsoft.com/en-us/dotnet/api/system.mathf?view=netcore-3.0>

# array (arreglos/matrices)

¿Qué es?

Una estructura que contiene una colección de valores del mismo tipo

¿para qué?

Para almacenar valores que normalmente tienen relación entre si

Sintaxis

Al igual que las variables, las array se pueden declarar y después iniciar mas adelante en el código o declarase e iniciarse en la misma línea

tipoDeDato [] nombre;

nombre = new tipoDeDato[cantidad de elementos que almacena];

tipoDeDato [] nombre = {valor1, valor2, valor3 …};

Agregar valores

Nombre[0] = “algún valor tanto entero como string”;

Existen varios tipos de array:

Array implícito Imagen que contiene captura de pantalla

Descripción generada automáticamente

Se puede decir que es la forma de declarar e iniciar una array

Array anonimo

Es el mas recomendado para trabajar con el bloque foraech

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Se define de esta forma y se recomienda trabajar con el bucle foreach porque se trabaja igual que una lista en Python

# Listas y tuplas

Listas

List<tipoDato> nombre = new list<tipoDato> ();

Agregar elementos a la listas

nombre.add(“contenido”);

nombre.insert(posición, elemento);

Tuplas

tipoDato nombre = (valor1 , valor2);

con tantos caompos sea necesaria la tupla