

Visual Studio .Net y C#



Temas a tratar:

4.1- Variables y Arreglos.

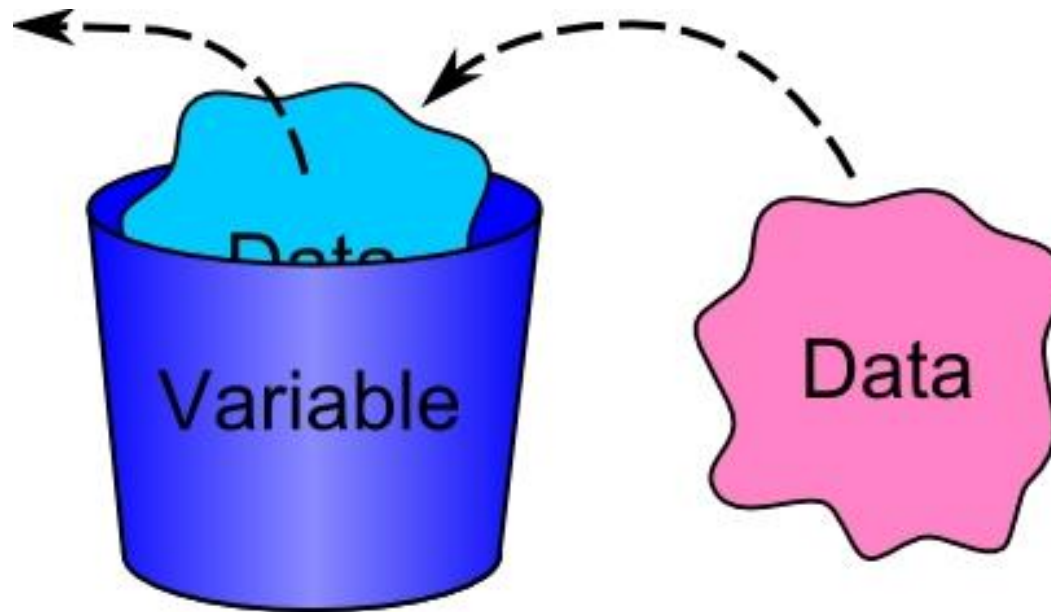
4.2- Ámbito y accesibilidad

4.3- Estructuras condicionales.

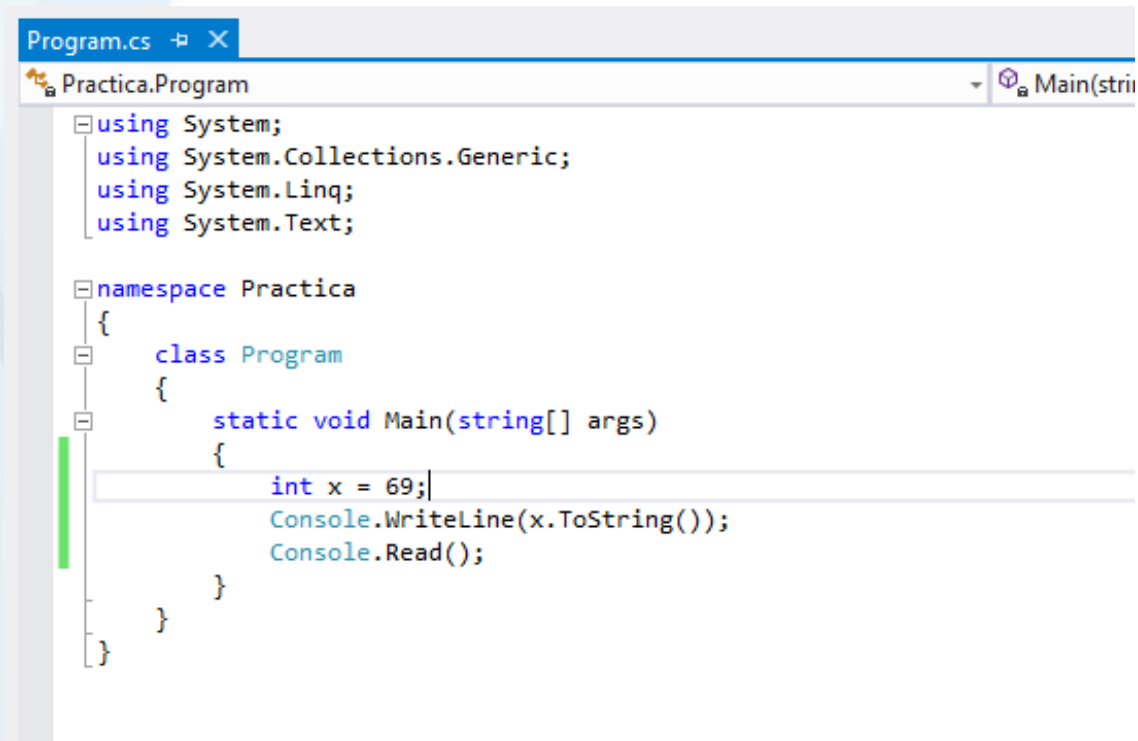
4.4- Procedimientos y funciones.

4.1.- Variables y Arreglos

Una variable representa un valor numérico o de cadena o un objeto de una clase. El valor que la variable almacena puede cambiar, pero el nombre sigue siendo el mismo. Una variable es un tipo de campo



- En C#, las variables se declaran con un tipo de datos y una etiqueta concretos.
- Se tiene que especificar si la variable es de tipo int, float, byte, short u otro cualquiera entre más de 20 tipos de datos diferentes.



The screenshot shows a C# program in a file named Program.cs. The code is as follows:

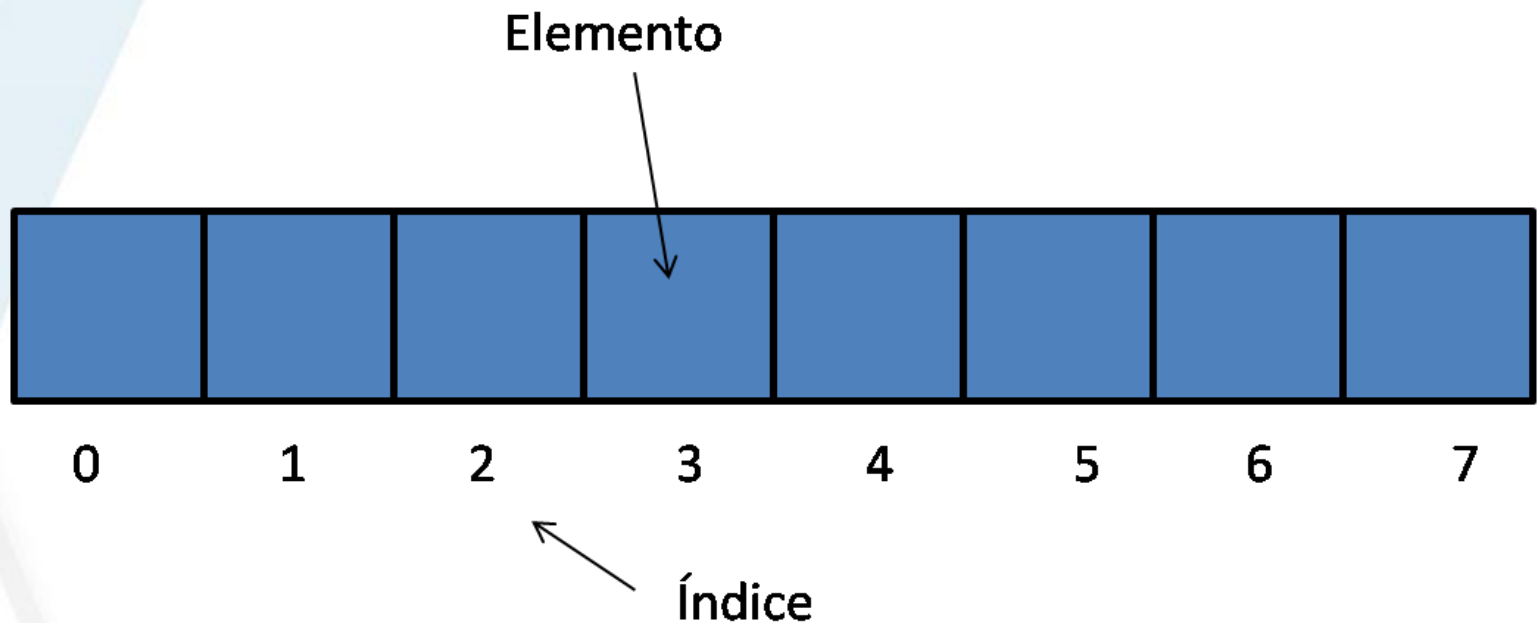
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Practica
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 69;
            Console.WriteLine(x.ToString());
            Console.Read();
        }
    }
}
```

The code demonstrates the declaration of an integer variable `x` with the value 69, followed by its conversion to a string and output to the console. The program also includes standard C# boilerplate code for namespaces, classes, and the `Main` method.

Arreglos

Es una colección de datos del mismo tipo. Sirve para manejar un número “n” de elementos en común, ya sea de tipos definidos por el Lenguaje, (int, float, String, etc...) así como aquellos definidos por el programador.



```
class Arreglos
{
    static void Main()
    {
        // Declara un arreglo uni-dimensional
        int[] array1 = new int[5];

        // Declara y asigna valores a un arreglo
        int[] array2 = new int[] { 1, 3, 5, 7, 9 };

        // Sintaxis alternativa
        int[] array3 = { 1, 3, 5, 7, 9 };

        // Declare un arreglo de 2 dimensiones
        int[,] multiDimensionalArray1 = new int[2, 3];

        // Declara y asigna valores a los elementos del arreglo bi-dimensional
        int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 } };

        // Declara un arreglo escalonado
        int[][] escalonadoArray = new int[6][];
        // Asigna valores al primer arreglo en la estructura escalonada
        escalonadoArray[0] = new int[4] { 1, 2, 3, 4 };
    }
}
```

4.2.- Ámbito y accesibilidad

Los **Setters & Getters** son métodos de acceso lo que indica que son siempre declarados públicos, y nos sirven para la accesibilidad de datos dentro de un objeto.

- **Setters:**

Permiten asignar un valor inicial a un atributo, pero de forma explícita, además el Setter nunca retorna nada (Siempre es void), y solo nos permite dar acceso público a ciertos atributos que deseemos el usuario pueda modificar.

- **Getters:**

Permite obtener (recuperar o acceder) el valor ya asignado a un atributo y utilizarlo para cierto método.

4.3.- Estructuras condicionales

La instrucción if - else permite controlar qué procesos tienen lugar, típicamente en función del valor de una o varias variables, de un valor de cálculo o booleano, o de las decisiones del usuario.

```
bool condition = true;
```

```
if (condition) {  
    Console.WriteLine("La variable es true.");  
} else {  
    Console.WriteLine("La variable es false.");  
}
```


Otro Ejemplo...

```
private void btnEquals_Click(object sender, EventArgs e)
{
    if (plusButtonClicked == true)
    {
        total2 = total1 + double.Parse(txtDisplay.Text);
    }
    else if (minusButtonClicked == true)
    {
        total2 = total1 - double.Parse(txtDisplay.Text);
    }

    txtDisplay.Text = total2.ToString();
    total1 = 0;
}
```

4.4.- Procedimientos y funciones

Una función es un conjunto de líneas de código que realizan una tarea específica y puede retornar un valor. Las funciones pueden tomar parámetros que modifiquen su funcionamiento.

Las funciones son utilizadas para descomponer grandes problemas en tareas simples y para implementar operaciones que son comúnmente utilizadas durante un programa y de esta manera reducir la cantidad de código.

```
//Función que devuelve un string (cadena)
static string devuelveTexto()
{
    return "Hola otra vez";
}

static void Main(string[] args)
{
    //Escribo el texto que devuelve la función
    Console.WriteLine(devuelveTexto());

    Console.ReadKey();
}
```