





Profesores:

- ❖ Ferrando Matías
 - ❖ Ferrando Eduardo
- 

EQUIVALENCIAS

DIAGRAMAS DE FLUJO -
VISUAL BASIC.NET -
C# -
JAVASCRIPT -
PHP



Índice

Consideraciones generales.....	2
C# / Javascript / PHP:	2
Visual Basic .NET:	2
Declarar una variable, con y sin asignación del valor por defecto	3
Variables predefinidas en PHP.....	4
Ámbito (niveles de accesibilidad)	5
Ámbitos predeterminados (si no se indica).....	6
Operadores lógicos y aritméticos	8
Comparaciones, If, Else... ..	10
Comparaciones Case	12
Bucles For y For Each (foreach)	13
Bucles While, Do... Loop.....	15
Abandonar un bucle o procedimiento	16
Visual Basic .NET:	16
Procedimientos / Métodos (funciones, propiedades)	17
Procedimiento de tipo Sub	17
Procedimiento de tipo Function	17
Procedimiento de tipo Property	18

Consideraciones generales

C# / Javascript / PHP:

Una cosa que debes tener muy presente cuando quieras escribir código en los mencionados lenguajes, es que todas las líneas deben acabar en un punto y coma (;). Debido a esta peculiaridad, puedes "alargar" cada línea de código en varias líneas, ya que el fin de una "sentencia" viene indicada por el punto y coma.

En estos lenguajes, se pueden agrupar líneas de código en bloques, los bloques siempre estarán dentro de un par de llaves: { y }

En C# todas las variables deben estar declaradas con un tipo de datos específico, lo mismo ocurre cuando asignamos datos de distintos tipos, siempre debe indicarse el tipo al que se quiere convertir. Sin embargo, en Javascript o PHP no se especifica el tipo de dato de la variable como ocurre en C#.

Visual Basic .NET:

En Visual Basic .NET cada línea física es una sentencia, si necesitas que ocupe más de una línea, tendrás que usar el guión bajo al final de la línea y continuar en la siguiente. Esto no es aplicable a los comentarios.

En VB.NET no se obliga a que se especifique el tipo de datos, pero si quieres hacer las cosas bien, deberías tener siempre conectado Option Strict On, esto te obligará a declarar las variables con el tipo de datos adecuado y así cuando hagas una asignación entre tipos diferentes de datos, tendrás que especificar el tipo, para ello hay que hacer una conversión explícita de datos (casting). Esto último es algo que muchos programadores de VB no suelen hacer, pero te lo recomiendo encarecidamente que lo hagas.

Declarar una variable, con y sin asignación del valor por defecto

En Visual Basic .NET se utiliza de la siguiente forma:

```
<ámbito> <variable> As <tipo>
<ámbito> <variable> As <tipo> = <valor>
```

En C# se declara de la siguiente forma:

```
<ámbito> <tipo> <variable>;
<ámbito> <tipo> <variable> = <valor>;
```

Diagramas de Flujo	Visual Basic .NET	C#	JavaScript	PHP
Declaracion de variables:	Dim i As Integer	int i;	Var numero = 5;	\$numero= 5;
Variable = 0	Dim i1 As Integer = 10	int i1 = 10;	Var texto = "texto";	\$texto="texto";
Variable = "texto"	Dim d As Double	double d;	Var bool = true;	\$bool= true;
	Dim d1 As Double = 3.5#	double d1 = 3.5;		Const NOMBRE = valor;
	Dim f As Single	float f;		
	Dim s As String	string s;		
	Dim c As Char	char c;		
	Dim l As Long	long l;		
	Dim m As Decimal	decimal m;		
	Dim o As MiClase	MiClase o;		
	Dim o1 As New MiClase()	MiClase o1 = new MiClase();		
		MiClase o2 = new MiClase();		

	<pre>Dim o2 As MiClase = New MiClase() Public Sp As String Private s1 As String</pre>	<pre>public string Sp; private string s1;</pre>		

En Visual Basic .NET cuando se declaran variables dentro de un procedimiento (método o propiedad) sólo se puede indicar Dim ya que esas variables serán privadas al procedimiento. En estos casos, en C# no se indicará el ámbito, simplemente el tipo de la variable.

Variables predefinidas en PHP

Php proporciona una gran cantidad de variables predefinidas. Estas son las principales de tipo array.

\$_SERVER – Información del entorno del servidor y de ejecución.

\$_GET – Variables recibidas por HTTP GET.

\$_POST – Variables recibidas por HTTP POST.

\$_FILES – Variables de subida de ficheros HTTP.

\$_REQUEST – Combina \$_GET, \$_POST y \$_FILES.

\$_COOKIE – Variables de las cookies HTTP.

\$_SESSION – Variables de sesión PHP.

Ámbito (niveles de accesibilidad)

Cualquiera de los miembros de una clase pueden tener un ámbito o nivel de accesibilidad que dependerá de cómo y desde dónde se podrá acceder a dicho miembro. Los ámbitos pueden ser: privados, públicos, a nivel de ensamblado, etc.

Veamos las equivalencias de los modificadores de ámbito entre Visual Basic y C#, así como una pequeña descripción de esos mismos ámbitos.

Visual Basic .NET	C#	Descripción del ámbito
Private	private	Accesible dentro del mismo módulo, clase o estructura.
Friend	internal	Accesible desde dentro del mismo proyecto, pero no desde fuera de él.
Protected	protected	Accesible desde dentro de la misma clase o desde una clase derivada de ella.
Protected Friend	protected internal	Accesible desde clases derivadas o desde dentro del mismo proyecto, o ambos.
Public	public	Accesible desde cualquier parte del mismo proyecto, desde otros proyectos que hagan referencia al proyecto, y desde un ensamblado generado a partir del proyecto.

Ámbitos predeterminados (si no se indica)

Cuando declaramos una variable, método, etc. y no indicamos el ámbito que tienen, el compilador le asigna uno de forma predeterminada, según el lenguaje que estemos usando o el tipo de elemento en el que estemos haciendo esa declaración, tendrá un ámbito o nivel de accesibilidad diferente.

En la siguiente tabla, podemos ver qué ámbito tendría, dependiendo de dónde se declare.







Miembros de	Accesibilidad predeterminada	Accesibilidades declaradas permitidas
enum	public	Ninguna
class	private	public protected internal private protected internal
interface	public	Ninguna
struct	private	public internal private

Equivalencias Diagramas de flujo - Visual Basic.NET - C# - JavaScript - PHP

VB: No he encontrado una lista con esta información... así que en algunos casos, simplemente lo he comprobado.

Miembros de	Accesibilidad predeterminada	Accesibilidades declaradas permitidas
Enum	Public	Ninguna
Class Module	Public	Public Protected Friend Private Protected Friend
Interface	Public	Ninguna
Structure	Siempre hay que indicar el ámbito	Public Friend Private

Operadores lógicos y aritméticos

Visual Basic .NET / Diagramas de Flujo	C#	JavaScript	PHP	Definición
+	+	+	+	Suma
-	-	-	-	Resta
\	/	/	/	División
*	*	*	*	Multiplicación
^	Math.pow(numero, exponente);	Math.pow(numero, exponente); **	**	Exponenciación
=	=	=	=	Asignación
+= / -=	+= / -=	+= / -=	+= / -=	Acumulación / Sustracción
	++a	++a	++\$a	Pre-Incremento
	a++	a++	\$a++	Post-Incremento
	--a	--a	--\$a	Pre-Decremento
	a--	a--	\$a--	Post-Decremento
Mod	%	%	%	Modulo
And	&	&&	&&	Y
AndAlso	&&			Y (Exclusivo)
Or				O

Equivalencias Diagramas de flujo - Visual Basic.NET - C# - JavaScript - PHP

OrElse				O (Exclusivo)
Not	!	!	!	No
=	==	=	==	Igual a
<>	!=	!=	!=	Diferente a
		===	===	Contenido y tipo igual a
		!==	!==	Contenido y tipo diferente a
&	+	Variable1 + ' ' + variable2	\$Variable1.\$Variable2	Concatenación de cadenas
Is Nothing	== null / IsNullOrEmpty	if (m === null) { }	is_null(\$var)	Comparación algo nulo
>	>	>	>	Mayor que
<	<	<	<	Menor que
>=	>=	>=	>=	Mayor o igual que
<=	<=	<=	<=	Menor o igual que

En C# sólo se utiliza el símbolo / para división tanto de números enteros como decimales

En VB la división de números enteros se realiza con \, la división de números decimales se hace con /.

Comparaciones, If, Else...

	Visual Basic .NET	C# / JavaScript / PHP
<p>Condicional Simple:</p> <pre> graph TD Start(()) --> Cond{Condición} Cond -- "Si [Entonces]" --> Box1[Instrucción 1
Instrucción 2
...
Instrucción n] Cond -- "No (Sino)" --> Box2[Instrucción A
Instrucción B
...
Instrucción Z] Box1 --> Exit(()) Box2 --> Exit </pre>	<pre> If x = 10 Then '... End If </pre>	<pre> if(x == 10) { //... } </pre>
	<pre> If x = 10 Then '... Else '... End If </pre>	<pre> if(x == 10) { //... } else { //... } </pre>
	<pre> If x = 10 Then '... ElseIf x > 50 Then '... End If </pre>	<pre> if(x == 10) { } else if(x > 50) { } </pre>

Equivalencias Diagramas de flujo - Visual Basic.NET - C# - JavaScript - PHP

En C# no es necesario usar llaves para indicar la instrucción a ejecutar cuando se cumpla la condición del IF, pero si hay más de una instrucción, es necesario usar las llaves. Estos ejemplos lo aclararán mejor:

```
// con una instrucción, no es necesario usar llaves
if(x == 10)
    Console.WriteLine(x);

// con más de una instrucción, hay que usar las llaves
if(x == 10)
{
    Console.WriteLine(x);
    x += 10;
}
```

En Visual Basic no es necesario usar End If si el código a ejecutar se incluye en una línea, pero esa instrucción (o instrucciones separadas por dos puntos (:)) deberán estar en la misma línea.

```
' con una instrucción
If x = 10 Then Console.WriteLine(x)

' con más de una instrucción, pero en una sola línea
' pero separando cada instrucción con dos puntos
If x = 10 Then Console.WriteLine(x) : x += 10
```

Comparaciones Case

Diagramas de Flujo	Visual Basic .NET	C# / JavaScript / PHP
<pre>graph TD INICIO([INICIO]) --> Input[/A, B/] Input --> Menu[/MENU
1. Sumar
2. Restar
3. Multiplicar
4. Dividir
Elija opción:/] Menu --> Option[/OPCION/] Option --> Decision{OPCION} Decision -- 1 --> R1[R=A+B] Decision -- 2 --> R2[R=A-B] Decision -- 3 --> R3[R=A*B] Decision -- 4 --> R4[R=A/B] R1 --> Output(()) R2 --> Output R3 --> Output R4 --> Output</pre>	<pre>Dim number As Integer = 8 Select Case number Case 1 To 5 Case 6, 7, 8 Case 9 Case Else End Select</pre>	<pre>Int/var caseSwitch = 1; switch (caseSwitch) { case 1: break; case 2: break; default: break; }</pre>

Bucles For y For Each (foreach)

En C#, cuando se usa un bucle **foreach**, siempre hay que declarar la variable a usar con el bucle.

Diagramas de Flujo	Visual Basic .NET	C# / JavaScript / PHP
<p>Ciclo de repeticion:</p> <pre> graph TD Inicio([Inicio]) --> Decision{Desde i = 1 a 35} Decision -- "i > 35" --> Exit(()) Decision -- "i <= 35" --> LeerNota[/Leer Nota(i)/] LeerNota --> Sumatorio[Sumatorio = Sumatorio + Nota(i)] Sumatorio --> Decision Sumatorio --> Media[Media = Sumatorio / 35] Media --> Mostrar[/Mostrar "La media es", Media/] </pre>	<pre> Dim i As Integer For i = 1 To 10 ' ... Next </pre>	<pre> Int/var i; for(i = 1; i<= 10; i++) { // ... } </pre>
	<p>' Sólo en Visual Studio .NET 2003</p> <pre> For i As Integer = 1 To 10 </pre>	<pre> for(int/var i = 1; i<=10; i++) </pre>
	<pre> Dim objeto As <Tipo> </pre>	<p>C#:</p> <pre> foreach(<Tipo> objeto in colección) </pre>

	For Each objeto In colección	
	' Sólo en Visual Studio .NET 2003 For Each objeto As <Tipo> In colección	<u>PHP:</u> Foreach(\$array as \$valor){ ... } \$computadora=["HP", "Windows"]; Foreach(\$computadora as \$pc){ Echo \$pc; }

Bucles While, Do... Loop

Diagramas de Flujo	Visual Basic .NET	C# / JavaScript / PHP
<p>Bucle Hacer Mientras:</p> <pre> graph TD Entry(()) --> I1[Instrucción 1] I1 --> I2[Instrucción 2] I2 --> In[Instrucción n] In --> D{Repetir Mientras [Condición]} D -- Sí --> Entry D -- No --> Exit(()) </pre>	<pre> Do '...' Loop While <expresión> </pre>	<pre> do { //... }while(<expresión>); </pre>
<p>Bucle Mientras que:</p>	<pre> While <expresión> '...' End While </pre>	<pre> while(<expresión>) { //... } </pre>
	<pre> Do While <expresión> '...' Loop </pre>	<pre> while(<expresión>) { //... } </pre>

<pre> graph TD Inicio([Inicio]) --> Init[Existencias = 1000] Init --> Decision{Mientras Existencias >= 200} Decision -- Sí --> Input[/Pedir Entregadas/] Input --> Calc[Existencias = Existencias - Entregadas] Calc --> Decision Decision -- No --> Output[/Mostrar "El inventario ha bajado de 200 unidades. Debe comunicarlo"/] Output --> Fin([Fin]) </pre>	Do	
	'...'	
	Loop	
	Do Until <expresión>	
	'...'	
	Loop	
	Do	
	'...'	
	Loop Until <expresión>	

Abandonar un bucle o procedimiento

Visual Basic .NET:

- Para abandonar un bucle Do... Loop, se utiliza Exit Do.
- Para abandonar un bucle While... End While, se utiliza Exit While.
- Para abandonar un bucle For o For Each, se utiliza Exit For.
- Para abandonar un procedimiento Function (función), se utilizará Exit Function.
- Para abandonar un procedimiento Sub, se utilizará Exit Sub.

- Para abandonar un procedimiento Property, se utilizará Exit Property.

C# / JavaScript / PHP:

- Para abandonar cualquier tipo de bucle, se utilizará break.
- Para abandonar cualquier tipo de procedimiento, se utilizará return.

Procedimientos / Métodos (funciones, propiedades)

En Visual Basic existen tres tipos de procedimientos: Sub, Function y Property. En C# los procedimientos pueden ser funciones o propiedades. Las funciones pueden o no devolver algún valor, en caso de que no devuelvan un valor se comportan como los Subs de Visual Basic.

Procedimiento de tipo Sub

Visual Basic .NET	C#
<code><ámbito> Sub <nombre>() End Sub</code>	<code><ámbito> void <nombre>() { }</code>

Procedimiento de tipo Function

Visual Basic .NET	C# / JavaScript / PHP
<code><ámbito> Function <nombre>() As <tipo></code>	<code><ámbito> <tipo> <nombre>() {</code>

End Function	}

Procedimiento de tipo Property

Visual Basic .NET	C#	JavaScript
<pre> <ámbito> Property <nombre>() As <tipo> Get '... End Get Set '... End Set End Property </pre>	<pre> <ámbito> <tipo> <nombre> { get{ //... } set{ //.. } } </pre>	<pre> // Definiendo todo explicitamente Object.defineProperty(obj, 'key', { enumerable: false, configurable: false, writable: false, value: 'static' }); </pre>
		<pre> var bValue = 38; Object.defineProperty(o, 'b', { get: function() { return bValue; }, set: function(newValue) { bValue = newValue; }, enumerable: true, configurable: true }); </pre>

Equivalencias Diagramas de flujo - Visual Basic.NET - C# - JavaScript - PHP

<pre><ámbito> ReadOnly Property <nombre> As <tipo> Get ' ... End Get End Property</pre>	<pre><ámbito> <tipo> <nombre> { get{ //... } }</pre>	
<pre><ámbito> WriteOnly Property <nombre> As <tipo> Set ' ... End Set End Property</pre>	<pre><ámbito> <tipo> <nombre> { set{ //... } }</pre>	