VB.net

Variables, constantes, comentarios, asignación, conversión de tipos, mostrar datos por pantalla, operadores, condiciones y bucles

Declaración de variables I

```
Dim nombre As tipo
Dim nombreAbreviatura
```

```
Dim edad As Integer
Dim edad%
```

- No puede empezar por número
- No puede tener espacios
- No puede estar formada por operadores (+, -, !...)
- No puede haber dos variables que se llamen igual aunque sean de diferente tipo
- Se recomienda escribir el nombre en minúsculas y si tienen más de una palabra separarlas con _ o colocar la primera letra de la segunda palabra en mayúsculas

- No es necesario hacer la declaración al principio
- No diferencia entre mayúsculas y minúsculas

Declaración de variables II

Tipo	Abreviatura	Asignación de almacenamiento	Valor por defecto
Byte		1 byte	0
Short		2 byte	0
Integer	%	4 byte	0
Long	&	8 byte	0
Single	!	4 byte	0.00
Double	#	8 byte	0.00
Decimal		16 byte	0.00
Char		2 byte	Nothing
String	$	En función de la plataforma de implementación	Nothing
Boolean	No tiene	2 byte	False
Object	No tiene	4 byte	Nothing
Date		8 byte	01/01/0001 0:00:00
Definido por el usuario		En función de la plataforma de implementación	

Declaración de variables III

- Se puede declarar más de una variable en la misma línea, pero se recomienda que estas sean del mismo tipo y tengan algún tipo de relación. Por ejemplo, variables que se van a usar para contadores.

```
Dim i As Integer, j As Integer, k As Integer

Dim i%, j%, k%

Dim i, j, k As Integer
```

- Se recomienda hacerlo como en el último ejemplo. Esta más abreviado que en el primero y más claro que en el segundo.

Option Explicit On

- Visual Basic permite no declarar una variable antes de ser usada. Pero es recomendable declararlas para generar buenos programas y evitar errores y confusiones.
- Para ello Visual Basic dispone de una instrucción que nos obliga a declarar todas las variables: Option Explicit On
- Para usarla hay dos opciones:
  - Escribirlo en la ventana de código
  - Configurarlo: Herramientas, Opciones, Proyectos y soluciones, Valores predeterminados de VB, Option Explicit, On

Declaración de constantes

- Es una variable que no cambia durante toda la ejecución.

Const nombre As tipo

- Siguen las mismas reglas que las variables

Asignación de valores

- Se puede asignar un valor al declarar la variable

  Dim nombre As tipo = valor

- O durante la ejecución del programa

  variable = valor

- Para asignar un valor leído por pantalla a una variable debemos usar el método Console.ReadLine :

  variable= Console.ReadLine()

Conversión de tipos I

- Tenemos que convertir variables para evitar errores a la hora de hacer asignaciones. Para ello usaremos los métodos de la clase System.Convert:

```
Dim x As Single = 123.5
Dim y As Integer
y=Convert.ToInt32(x)
```

- O el método Parse del que disponen cada una de las clases que representan los tipos de datos:

```
y=Integer.Parse(x)
```

Conversión de tipos II: Convert

ToBoolean	Convierte un valor especificado en un valor booleano equivalente.
ToByte	Convierte un valor especificado en un entero de 8 bits sin signo.
ToChar	Convierte un valor especificado en un carácter Unicode.
ToDateTime	Convierte un valor especificado en un tipo DateTime.
ToDecimal	Convierte un valor especificado en un número Decimal.
ToDouble	Convierte un valor especificado en un número de punto flotante de precisión doble.
ToInt16	Convierte un valor especificado en un entero de 16 bits con signo.
ToInt32	Convierte un valor especificado en un entero de 32 bits con signo.
ToInt64	Convierte un valor especificado en un entero de 64 bits con signo.
ToSByte	Convierte un valor especificado en un entero de 8 bits con signo.
ToSingle	Convierte un valor especificado en un número de punto flotante de precisión simple.
ToString	Convierte el valor especificado en la representación de tipo String equivalente.
ToUInt16	Convierte un valor especificado en un entero de 16 bits sin signo.
ToUInt32	Convierte un valor especificado en un entero de 32 bits sin signo.
ToUInt64	Convierte un valor especificado en un entero de 64 bits sin signo.

Comentarios

- Podemos usar la palabra reservada Rem. El comentario debe empezar en una nueva línea o estar separado del enunciado anterior por dos puntos.

  Rem esto es un comentario

  x=20 :Rem esto es un comentario

- O podemos usar una comilla simple antes del comentario

  ‘esto es un comentario

  x=20 ‘esto es un comentario

Mostrar datos por pantalla I

- Se usa el método System.Console.WriteLine. Se puede usar de varias maneras:

```
Console.WriteLine(“El valor es igual a “ & a)
Console.WriteLine(“El valor es igual a {0}”,a)
```

- Si lo hacemos de la segunda manera podemos incluir especificaciones de formato. Estas están compuestas por:

{posición[,ancho][:formato]}

```
Console.WriteLine(“El valor es igual a {0,4:d3}”,a)
```

Mostrar datos por pantalla II: Ancho y formato

- El ancho:
  - Es el mínimo número de posiciones que va a ocupar el dato en la salida.
  - Si es negativo se justifica a la izquierda y si no a la derecha
- El formato:
  - Como se va a visualizar el dato.
  - Hay formatos predefinidos que pueden ir seguidos de un número que representaría los decimales.

Carácter	Descripción
C o c	Moneda
D o d	Enteros
E o e	Científico
F o f	Coma fija
G o g	General
N o n	Numérico
P o p	%
X o x	Hexadecimal

  - O puedes crear los tuyos propios
    - https://docs.microsoft.com/es-es/dotnet/api/microsoft.visualbasic.strings.format?view=netframework-4.8

Mostrar datos por pantalla III

```
Dim a As Double = 0.001256
Console.WriteLine(“El valor es igual a {0,8:P3}”, a)
Console.WriteLine(“El valor es igual a {0,-10:e2}”, a)
Console.WriteLine("El valor es igual a {0:#,##0.00000""euro""}", a)
```

```
El valor es igual a  0,126 %
El valor es igual a 1,26e-003
El valor es igual a 0,00126euro
```

Operadores aritméticas

+	Suma. Operandos enteros o reales	+=
-	Resta. Operandos enteros o reales	-=
*	Multiplicación. Operandos enteros o reales	*=
/	División. Operandos enteros o reales. Resultado siempre Double	/=
\	División entera. Operandos enteros. Si alguno es real, será convertido a entero. Resultado siempre entero	\=
^	Exponencial. Operandos enteros o reales	^=
mod	Resto de una división. Operandos enteros o reales. Si ambos operandos son enteros el resto será entero, si no real.	mod=

Cuando en una operación aritmética los operandos son de distinto tipo, ambos operandos son convertidos al tipo de operando de precisión más alta. El resultado es convertido al tipo de la variable que lo almacena.

Operadores condicionales

<	Menor
>	Mayor
<=	Menor o igual
>=	Mayor o igual
<>	Diferente
=	Igual

Operadores lógicos

And	Da true si ambos son true, si no false
AndAlso	Igual que el and, pero si la primera condición es false no sigue evaluando
Or	Da false si ambos son false, si no true
OrElse	Igual que el or, pero si la primera condición es true no sigue evaluando
Not	Da false si es true, y true si es false
Xor	Da true cuando una condición es false y la otra true

Expresiones condicionales I: If

If condición then instrucción [else instrucción]

- Si hay más de una instrucción podrían escribirse en una línea separando las instrucciones por dos puntos o escribirlo en más de una línea.

```
If condición then
  instrucción1
  instrucción2
[else
  instrucción1
  instrucción2]
End if
```

Expresiones condicionales II: ElseIf

```
If condición then
  instrucciones
ElseIf condición then
  instrucciones
ElseIf condición then
  instrucciones
Else
  instrucciones
End If
```

- Podrían ser sustituidos por if anidados

Expresiones condicionales III: Select I

```
Select Case (expresión)
  Case expresión1
    instrucciones
  Case expresión2
    instrucciones
 …
   …
  Case Else
    instrucciones
End Select
```

Expresiones condicionales III: Select II

- Las expresiones de los Case pueden ser valores, intervalos, is condición o todas estas separadas por comas.
- Aquí podéis ver algunos ejemplos:

```
Case 3

Case 11, 13, 15

Case Is >=10

Case “si”

Case 1 To 20

Case -1, 2 To 5, Is <-3
```

Bucles I: While

```
While (condicion)
  instrucciones
End While
```

Bucles II: Do….Loop While/Until

```
Do
  instrucciones
Loop While/Until (condicion)
```

- Se diferencia del while en que evalúa la condición al final, por lo que al menos el bucle se hará una vez.

Bucles III: For

```
For variable=valor1 To valor2 [Step valor3]
  instrucciones
Next
```

- Si no ponemos el Step se entiende que la variable se va incrementando en 1. Si lo ponemos tendremos que ponerle un número positivo o negativo dependiendo de si queremos que en cada paso incremente o decremente.