1. Tipos de datos

Las variables, como su nombre lo indica, se utilizan para almacenar valores que tienen la propiedad de variar el contenido. Cuando hablamos de contenido nos referimos a cualquier tipo de datos, por ejemplo un nombre, una fecha, un color, un número etc... .

A las variables se les asigna **un nombre** para poder utilizarlas. Por ejemplo puedo crear una variable llamada fecha y esta almacenará una fecha. A los nombres de las variables se los denomina **identificadores**. Cuando creamos variables, tenemos que tratar de asignarles un nombre que se relacione con el tipo de dato que queremos almacenar.

En **visual basic a las variables conviene declararlas**, o sea, avisarle a vb que vamos a utilizar dichas variables. A estas se las declara en el comienzo del código y se les antepone la **palabra reservada Dim**, luego el nombre que nosotros queramos y seguido el tipo de dato que almacenará, por ejemplo si quiero almacenar en una variable llamada **Numero.**

```
Dim numero As Integer
```

Los tipos de datos, indican el **tipo de valor que puede almacenar una variable**. Los principales tipos de datos:

Integer	Valor Entero	2 Bytes
Long	Valor Entero Largo	4 Bytes
Single	Valor Real	4 Bytes
Double	Valor Real Doble	8 Bytes
String	Carácter (texto)	1 Byte por carácter
Byte	Byte	1 Byte
Boolean	Valor Booleano (1/0)	2 Bytes
Currency	Monedas y Punto Fijo	8 Bytes
Date	Fecha	8 Bytes
Object	Referencias a objetos	4 Bytes
Variant	Cualquiera	16-22 Bytes

Podemos declarar las variables con las siguientes formas abreviadas:

Integer %
Long &
Single !
Double #
String $
Currency @

```
Dim num1 As Integer
Dim num2 As Integer

num1 = 10
num2 = 20

'se mostrará un mensaje con la
suma de las variables con el
resultado 30
MsgBox num1 + num2
```

```
Dim nombre As String
Dim apellido As String
'le establecemos valores
nombre = "Carlos"
apellido = "Peres"

'mostramos un mensaje con el
valor de las variables
MsgBox nombre
MsgBox apellido
```

Tambie´podemos declarar las variables de la siguiente forma:

Dim A% → Dim A As Integer

Dim nombre$ → Dim nombre As String

Las variables se pueden declarar con los siguientes modificadores, que afecta a la visibilidad de la misma:

Public Class classForEverybody

Protected Class classForMyHeirs

Friend stringForThisProject As String

Private numberForMeOnly As Integer

En ausencia de ellos se determinan como privadas.

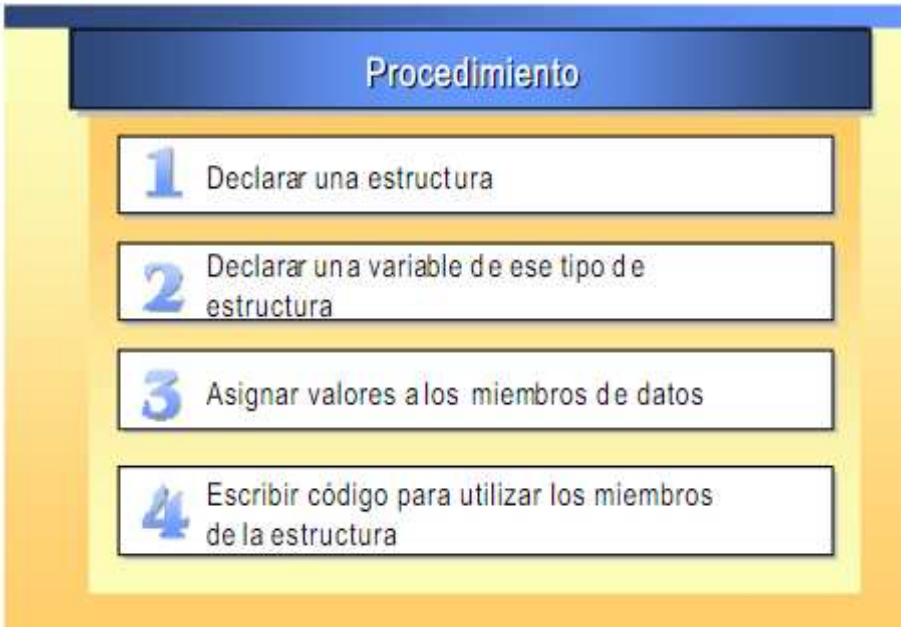
Tipos creados por el usuario:

```
Structure Monitor
  Dim Marca As String
  Dim Pulgadas As Integer
  Dim Resolución As String
  Dim Velocidad As String
End Structure
```

Para definir una variable de tipo estructura, una vez declarada la estructura haremos:

```
Dim M As Monitor
Dim M As new Monitor
```

Cómo utilizar las estructuras:



1. Declarar una estructura.

   Por ejemplo, podemos crear una estructura que registre información sobre el sistema de un ordenador, como sigue:

```
Private Structure computerInfo
   Public processor As String
   Public memory As Long
   Public purchaseDate As Date
End Structure
```

2. Declarar una variable del tipo de estructura declarado.

   Una vez declarada una estructura, podemos crear una variable de ese tipo de estructura, como sigue:

```
Dim mySystem As computerInfo
```

3. Asignar valores a los miembros de datos.

   Para asignar y recuperar valores desde los elementos de una variable estructura, utilizamos el nombre de la variable que hemos creado junto con el elemento en el bloque de estructura. Separamos el nombre del elemento con un punto. Por ejemplo, podemos acceder e inicializar las variables declaradas en la estructura *computerInfo* como sigue:

```
mySystem.processor = "x86"
mySystem.purchaseDate = #1/1/2003#
mySystem.memory = TextBox1.Text
```

4. Escribir código para utilizar los miembros de la estructura.

   Después de asignar valores a los miembros de datos, podemos escribir código para utilizarlos desde cualquier lugar de nuestro código que se encuentre dentro de su ámbito. En este ejemplo, los miembros de datos únicamente son accesibles desde dentro del archivo donde la estructura se ha declarado. Por ejemplo, puede verificar si el ordenador tiene suficiente memoria para instalar una determinada aplicación:

```
If mySystem.memory < 64000 Then NotEnoughMemory = True
```

Tratamiento de fechas

https://docs.microsoft.com/es-es/dotnet/api/microsoft.visualbasic.dateandtime?view=netframework-4.7.2

El módulo `DateAndTime` contiene los procedimientos y propiedades que se utilizan en las operaciones de fecha y hora.

```
Dim thisDate As Date

thisDate = Today
```

Propiedades

DateString	Devuelve o establece un valor `String` que representa la fecha actual d con el sistema.

Now	Devuelve un valor `Date` que contiene la fecha y la hora actuales de ac el sistema.
	`Dim ThisMoment As Date`
	`ThisMoment = Now`

TimeOfDay	Devuelve o establece un valor `Date` que contiene la hora del día actua acuerdo con el sistema.
Timer	Devuelve un valor `Double` que representa el número de segundos tran desde la medianoche.
TimeString	Devuelve o establece un valor de tipo `String` que representa la hora a día según el sistema.
Today	Devuelve o establece un valor `Date` que contiene la fecha actual de ac el sistema.

Métodos

DateAdd(DateInterval, Double, DateTime)	Devuelve un valor de `Date` que contiene un valor de fecha y hora al qu sumado un intervalo de tiempo especificado.
DateAdd(String, Double, Object)	Devuelve un valor de `Date` que contiene un valor de fecha y hora al qu sumado un intervalo de tiempo especificado.

DateDiff(DateInterval, DateTime, DateTime, FirstDayOfWeek, FirstWeekOfYear)	Devuelve un valor `Long` que especifica el número de intervalos de tien dos valores de `Date`.
DateDiff(String, Object, Object, FirstDayOfWeek, FirstWeekOfYear)	Devuelve un valor `Long` que especifica el número de intervalos de tien dos valores de `Date`.
DatePart(DateInterval, DateTime, FirstDayOfWeek, FirstWeekOfYear)	Devuelve un valor `Integer` que contiene el componente especificado valor de `Date` dado.
DatePart(String, Object, FirstDayOfWeek, FirstWeekOfYear)	Devuelve un valor `Integer` que contiene el componente especificado valor de `Date` dado.
DateSerial(Int32, Int32, Int32)	Devuelve un valor de `Date` que representa un año, mes y día especific la información de hora establecida en medianoche (00:00:00).
DateValue(String)	Devuelve un valor de `Date` que contiene la información de fecha repre por una cadena, con la información de hora establecida en medianoche (00:00:00).
Day(DateTime)	Devuelve un valor de tipo `Integer` del 1 al 31 que representa el día de
Equals(Object)	Determina si el objeto especificado es igual al objeto actual. (Inherited from Object)
GetHashCode()	Sirve como la función hash predeterminada. (Inherited from Object)
GetType()	Obtiene el Type de la instancia actual. (Inherited from Object)
Hour(DateTime)	Devuelve un valor de tipo `Integer` del 0 al 23 que representa la hora

MemberwiseClone()	Crea una copia superficial del Object actual. (Inherited from Object)
Minute(DateTime)	Devuelve un valor de tipo `Integer` del 0 al 59 que representa el minu hora.
Month(DateTime)	Devuelve un valor de tipo `Integer` del 1 al 12 que representa el mes d
MonthName(Int32, Boolean)	Devuelve un valor de tipo `String` que contiene el nombre del mes esp
Second(DateTime)	Devuelve un valor de tipo `Integer` del 0 al 59 que representa el segun minuto.
TimeSerial(Int32, Int32, Int32)	Devuelve un valor de `Date` que representa una hora, un minuto y un se especificados, con la información de fecha establecida con respecto al 1 del año 1.
TimeValue(String)	Devuelve un valor de `Date` que contiene la información de hora repres por una cadena, con la información de fecha establecida en el 1 de ener 1.
ToString()	Devuelve una cadena que representa el objeto actual. (Inherited from Object)
Weekday(DateTime, FirstDayOfWeek)	Devuelve un valor de tipo `Integer` que contiene un número que repre día de la semana.
WeekdayName(Int32, Boolean, FirstDayOfWeek)	Devuelve un valor de tipo `String` que contiene el nombre del día de la especificado.
Year(DateTime)	Devuelve un valor de tipo `Integer` del 1 al 9999 que representa el año

Ejemplos de uso con las funciones de DATE

- ara crear una valor de fecha proporcionando por separado el año, el mes y el día usaremos:

```
Dim fecha as Date = New Date(2014, 2, 21)
```

Nota: *Date* y *DateTime* son la misma función, en la anterior línea de código podríamos haber usado *DateTime* en vez de *Date*.

- Para obtener la fecha actual del sistema usaremos:

```
Dim fechaActual as Date = Date.Now
```

- Para obtener la hora actual del sistema (incluyendo los ticks):

```
Dim hora as TimeSpan = Date.Now.TimeOfDay
```

- Para obtener los días transcurridos desde el 1 de enero del año actual:

```
Dim dias as Integer = Date.Today.DayOfYear
```

- Para sumar un día a una fecha:

```
Dim fechaManana as Date = Date.Today.AddDays(1)
```

- Para restar un día a una fecha:

```
Dim fechaAyer as Date = Date.Today.AddDays(-1)
```

- Para obtener la hora que será cuando transcurran dos horas y media de la hora actual:

```
Dim hora as TimeSpan = Date.Now.AddHours(2.5)
```

- Para obtener la hora que será cuando transcurran dos días, 10 horas y 30 minutos desde la fecha actual:

```
Dim hora as TimeSpan = Date.Now.Add(New TimeSpan(2, 10, 30, 0))
```

- Para obtener la hora que fue hace un día, 12 horas y 30 minutos desde la fecha actual:

```
Dim hora as TimeSpan = Date.Now.Subtract(New TimeSpan(1, 12, 30, 0))
```

- Para obtener el número de días de un mes determinado:

```
Dim dias as Integer = Date.DaysInMonth(2014,2)
```

- Para recuperar los nombres estándar o abreviados de los días de la semana y de los meses de acuerdo con la configuración local o el idioma:
    - ```
      Dim mes as String
      For Each mes In DateTimeFormatInfo.CurrentInfo.AbbreviatedMonthNames
        Console.WriteLine(mes)
      Next
      ```

- Para obtener fechas en varios formatos:
    - ```
      Dim fecha as Date = New Date(2014, 2, 21, 10, 12, 20, 500)
      Console.WriteLine(fecha.ToShortDateString)
      Console.WriteLine(fecha.ToLongDateString)
      Console.WriteLine(fecha.ToShortTimeString)
      ```

Conversiones de tipo

Cada función convierte una expresión a un tipo de datos específico.

Nombre de la función	Tipo de datos devuelto	Intervalo para `expression` argumento
`CByte`	Byte (tipo de datos)	0 y 255 (sin signo); partes fraccionarias se redondean. [1]
`CChar`	Char (tipo de datos)	Cualquier `Char` o `String` expresión; sólo primer carácter de un String se convierte; puede oscilar entre 0 y 65535 (sin signo).
`CDate`	Date (tipo de datos)	Cualquier representación válida de una fecha y hora.
`CDbl`	Double (tipos de datos)	-1, 79769313486231570E + 308 a - 4, 9406564584124654 4E-324 para negativos; 4, 9406564584124654 4E-324 a 1, 79769313486231570E + 3 para valores positivos.
`CDec`	Decimal (tipo de datos)	+/-79,228,162,514,264,337,593,543,950,335 para números a partir de decir, números sin decimales. Para números con 28 posiciones decima

Nombre de la función	Tipo de datos devuelto	Intervalo para `expression` argumento
		intervalo es +/-7,9228162514264337593543950335. El menor número posible distinto de cero es 0,0000000000000000000000000001 (+/-1E
`CInt`	Integer (tipo de datos)	-2.147.483.648 y 2.147.483.647; partes fraccionarias se redondean. [1]
`CLng`	Long (tipo de datos)	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807; partes fraccio se redondean. [1]
`CObj`	Tipo de objeto de datos	Cualquier expresión válida.
`CSByte`	SByte (tipo de datos)	-128 a 127; partes fraccionarias se redondean. [1]
`CShort`	Short (tipo de datos)	-32.768 a 32.767; partes fraccionarias se redondean. [1]
`CSng`	Single (tipo de datos)	-3, 402823E + 38 a - 1, 401298E-45 para los valores negativos; 1, 4012 a 3, 402823E + 38 para valores positivos.
`CStr`	String (tipo de datos)	Devuelve para `CStr` dependen de la `expression` argumento. Vea v devueltos para la función CStr.
`CUInt`	UInteger (tipo de datos)	de 0 a 4.294.967.295 (sin signo); partes fraccionarias se redondean. [1]

Nombre de la función	Tipo de datos devuelto	Intervalo para `expression` argumento
`CULng`	ULong (tipo de datos)	de 0 a 18.446.744.073.709.551.615 (sin signo); partes fraccionarias se redondean. [1]
`CUShort`	UShort (tipo de datos)	de 0 a 65.535 (sin signo); partes fraccionarias se redondean. [1]

Validación de campos

```
public static bool IsNumeric (object Expression);
```

Devuelve un valor de tipo `Boolean` que indica si una expresión puede evaluarse como un número

```
Dim testVar As Object
Dim numericCheck As Boolean
testVar = "53"
' The following call to IsNumeric returns True.
numericCheck = IsNumeric(testVar)
testVar = "459.95"
' The following call to IsNumeric returns True.
numericCheck = IsNumeric(testVar)
testVar = "45 Help"
' The following call to IsNumeric returns False.

numericCheck = IsNumeric(testVar)
```

```
public static bool IsNothing (object Expression);
```

Devuelve un valor de tipo `Boolean` que indica si una expresión no tiene ningún objeto asignado.

```
Dim testVar As Object
' No instance has been assigned to variable testVar yet.
Dim testCheck As Boolean
' The following call returns True.
testCheck = IsNothing(testVar)
 Assign a string instance to variable testVar.
testVar = "ABCDEF"
' The following call returns False.
```

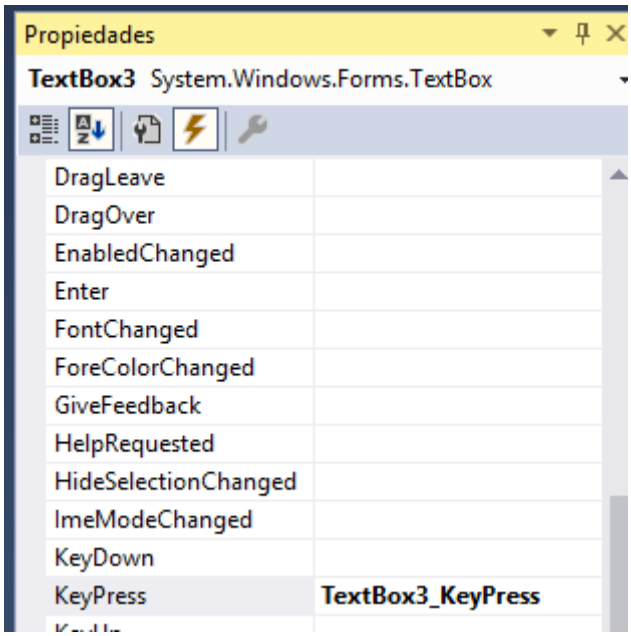
```
testCheck = IsNothing(testVar)
' Disassociate variable testVar from any instance.
testVar = Nothing
' The following call returns True.

testCheck = IsNothing(testVar)
```

Tambien podemos comprobar que los campos no están vacios con :

```
If Trim(Text1.Text) = "" Then
MsgBox "vacio"
End If
```

Tambien desde el código:



Private Sub TextBox3_KeyPress(sender As Object, e As KeyPressEventArgs) Handles TextBox3.KeyPress

```
        If Char.IsNumber(e.KeyChar) Then
            e.Handled = False
        ElseIf Char.IsControl(e.KeyChar) Then
            e.Handled = False
        ElseIf Char.IsSeparator(e.KeyChar) Then
            e.Handled = False

        Else
            e.Handled = True
        End If
    End Sub
```

Private Sub TextBox2_KeyPress(sender As Object, e As KeyPressEventArgs) Handles TextBox2.KeyPress

```
        If Char.IsLetter(e.KeyChar) Then
            e.Handled = False
        ElseIf Char.IsControl(e.KeyChar) Then
            e.Handled = False
```

```
        ElseIf Char.IsSeparator(e.KeyChar) Then
            e.Handled = False

        Else
            e.Handled = True
        End If

    End Sub
```

Estructuras de control:

Las **estructuras de control** se utilizan para controlar el flujo del programa en una rutina o función. Mediante ellas podemos controlar, mediante una condición, que se ejecute una determinada línea o bloque de líneas de código .. o que no se ejecuten.

Estructura If - Then - Else

Esta estructura permite mediante una condición, que se ejecute o no se ejecute determinada tarea o línea de código.

Por ejemplo supongamos que tenemos en un formulario un control Label con la **propiedad Caption** con el valor **50**

```
If Label1.Caption = "50" Then
msgbox "mi label tiene un valor de 50"
else
msgbox "mi label NO tiene un valor de 50"
end if
```

No es obligatorio usar If y else juntos en la misma cláusula

Estructura While - Wend

Esta estructura de control repetirá sin detenerse un determinado código mientras se cumpla una condición.

Por ejemplo supongamos que tenemos una variable llamada **x** que tiene un **valor de 100**.

```
while x = 100
...se ejecutan todas las líneas de código que estén aquí
wend
```

```
Module CicloWhile
    Sub Main()
        Dim contador As Integer
        contador = 0
        While (contador <= 10)
            Console.WriteLine(contador)
            contador = contador + 1
        End While
        Console.WriteLine("Termino el programa")
        Console.ReadLine()
    End Sub
End Module
```

```
Dim index As Integer = 0
While index < 100000
    index += 1

    ' If index is between 5 and 7, continue
    ' with the next iteration.
    If index >= 5 And index <= 8 Then
        Continue While
    End If

    ' Display the index.
    Debug.Write(index.ToString & " ")

    ' If index is 10, exit the loop.
    If index = 10 Then
        Exit While
    End If
End While
```

Estructura For - next

La estructura es utilizada para generar una repetición de instrucciones o bucle, pero no evalúa una condición como en el caso del bucle While, si no que lo hace entre un número inicial y un número final que le debemos indicar al mismo.

Por ejemplo, tenemos un formulario con un Label1 y declaramos una variable de tipo integer llamada "contador" como en el caso anterior

```
Private Sub Command1_Click()
Dim contador As Integer

For contador = 0 To 1000

    Label1.Caption = contador

Next contador
```

```
End Sub
```

Step VALOR_DE_PASO: opcional. Valor numérico que irá sumándose al CONTADOR a lo largo del bucle. Es un numero natural.

En este ejemplo añadimos la propiedad “step 2” que incrementará el contador de 2 en 2. Es decir, inicialmente el contador valdrá 1, luego 3, luego 5 y así sucesivamente.

```
For CONTADOR = 1 To 10 Step 2
   fila = CONTADOR
   Cells(fila, 2) = CONTADOR
Next
```

En este ejemplo el bucle for next en vba va contando hacia atrás gracias al step -1 y al contador que empieza en vez de en 1 en 10 y acaba en 1.

```
For CONTADOR = 10 To 1 Step -1
   fila = CONTADOR
   Cells(fila, 3) = CONTADOR
Next
```

En este ejemplo utilizamos la expresión Exit For cuando el contador llega hasta 49 y lanzamos un mensaje en pantalla en este momento.

```
For CONTADOR = 10 To 100
   If CONTADOR = 49 Then
      MsgBox "El contador ha llegado al número " & CONTADOR
      Exit For
   End If
Next
```

Do… Loop Until

Esta estructura de control se puede usar para ejecutar un bloque de instrucciones un número indefinido de veces. Las instrucciones se repiten hasta que una condición llegue a ser **True**.

Ejemplo:

```
Sub Ejemplo1()
  Dim contador As Integer
  Dim numero As Integer
  numero = 9
  Do Until numero = 10
   If numero = 0 Then Exit Do
     numero = numero – 1
     contador = contador + 1
  Loop
  MsgBox “Se alcanzó el valor ” &amp; numero &amp; ” ” &amp; contador
End Sub
```

Do While… Loop

En este caso, las instrucciones se repiten mientras una condición sea **True** (al contrario que con el **Do… Loop Until**).

Ejemplo:

```
Sub Ejemplo2()
  Dim Escribir As Integer
  Escribir = 1
  Do While Escribir < 7
    ActiveCell.FormulaR1C1 = “Excel”
    lastrow = Cells(Rows.Count, 1).End(xlUp).Row
    Cells(lastrow, 1).Offset(1, 0).Select
    Escribir = Escribir + 1
  Loop
End Sub
```

Estructura Select case

La estructura **Select Case** se suele utilizar para evitar el uso de muchas cláusulas If y de esta manera no tener que anidarlas. La claúsula **Select** evalúa una condición y las cláusulas **case**contienen valores, si el valor que contiene la cláusula **case es igual** a la condición que se evaluó, ejecutará las instrucciones en dicha cláusula.

```
Dim Nombre As String

Nombre = Text1

Select Case Nombre

  Case "Jorge"

       MsgBox "Se ejecutó la cláusula case: " & Nombre
  Case "Pedro"

       MsgBox "Se ejecutó la cláusula case: " & Nombre

  Case "Carolina"

       MsgBox "Se ejecutó la cláusula case: " & Nombre

End Select
```

En el caso anterior solo hemos colocado un valor para cada Case. Si quisieramos colocar mas valores podríamos hacerlo de esta forma:

```
Private Sub Form_Load()

Dim x As Integer

x = 10 'Le ponemos un valor a x

Select Case x
```

```
Case 5, 10
MsgBox "Se ejecutó el case que tiene el 5 y el 10"
'Se ejecuta esta sección

Case 15, 20
'Esto no se ejecuta
Case 25, 30
'Esto tampoco
End Select

End Sub
```

Cadenas de caracteres

El módulo `Strings` contiene procedimientos que se utilizan para llevar a cabo operaciones con cadenas.

```
Dim TestString As String = "Look at these!"
```

Los métodos son:

https://docs.microsoft.com/es-es/dotnet/api/microsoft.visualbasic.strings?view=netframework-4.7.2

Asc(Char)	Devuelve un valor de tipo `Integer` que representa el código de que corresponde a un carácter.
	`Dim codeInt As Integer`
	`' The following line of code sets codeInt to 65.`
	`codeInt = Asc("A")`

Asc(String)	Devuelve un valor de tipo `Integer` que representa el código de que corresponde a un carácter.
	`Dim codeInt As Integer`
	`' The following line of code sets codeInt to 65.`
	`codeInt = Asc("Apple")`

AscW(Char)	Devuelve un valor de tipo `Integer` que representa el código de que corresponde a un carácter.

AscW(String)	Devuelve un valor de tipo `Integer` que representa el código de que corresponde a un carácter.

Chr(Int32)	Devuelve el carácter asociado al código de carácter especificado
	`Dim associatedChar As Char`
	`' Returns "A".`
	`associatedChar = Chr(65).`

ChrW(Int32)	Devuelve el carácter asociado al código de carácter especificado.
Equals(Object)	Determina si el objeto especificado es igual al objeto actual. (Inherited from Object)
Filter(Object[], String, Boolean, CompareMethod)	Devuelve una matriz basada en cero que contiene un subconjunt matriz `String` basada en criterios de filtro especificados.
Filter(String[], String, Boolean, CompareMethod)	Devuelve una matriz basada en cero que contiene un subconjunt matriz `String` basada en criterios de filtro especificados.
Format(Object, String)	Devuelve una cadena con el formato que especifiquen las instruc contenidas en una expresión `String` de formato.
FormatCurrency(Object, Int32, TriState, TriState, TriState)	Devuelve una expresión con formato de moneda en la que se uti símbolo de moneda que se haya definido en el panel de control sistema.
FormatDateTime(DateTime, DateFormat)	Devuelve una expresión de cadena que representa un valor de fe hora.
FormatNumber(Object, Int32, TriState, TriState, TriState)	Devuelve una expresión con formato de número.
FormatPercent(Object, Int32, TriState, TriState, TriState)	Devuelve una expresión con formato de porcentaje (es decir, mu por 100) con un carácter % final.

GetChar(String, Int32)	Devuelve un valor de `Char` que representa el carácter correspon índice especificado en la cadena proporcionada.
GetHashCode()	Sirve como la función hash predeterminada. (Inherited from Object)
GetType()	Obtiene el Type de la instancia actual. (Inherited from Object)
InStr(Int32, String, String, CompareMethod)	Devuelve un entero que especifica la posición inicial de la primer aparición de una cadena dentro de otra.
InStr(String, String, CompareMethod)	Devuelve un entero que especifica la posición inicial de la primer aparición de una cadena dentro de otra.
InStrRev(String, String, Int32, CompareMethod)	Devuelve la posición de la primera aparición de una cadena dent otra, comenzando por el extremo derecho de la cadena.
Join(Object[], String)	Devuelve una cadena creada a partir de la combinación de varias subcadenas contenidas en una matriz.
Join(String[], String)	Devuelve una cadena creada a partir de la combinación de varias subcadenas contenidas en una matriz.
LCase(Char)	Devuelve una cadena o un carácter convertidos en minúscula.
LCase(String)	Devuelve una cadena o un carácter convertidos en minúscula.
Left(String, Int32)	Devuelve una cadena que contiene un número especificado de c a partir del lado izquierdo de una cadena.
Len(Boolean)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(Byte)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var

Len(Char)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(DateTime)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(Decimal)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(Double)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(Int16)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(Int32)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(Int64)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(Object)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(SByte)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(Single)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(String)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(UInt16)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
Len(UInt32)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var

Len(UInt64)	Devuelve un entero que contiene el número de caracteres de una o el número nominal de bytes necesarios para almacenar una var
LSet(String, Int32)	Devuelve una cadena alineada a la izquierda que contiene la cad especificada ajustada a la longitud indicada.
LTrim(String)	Devuelve una cadena que contiene una copia de una cadena esp sin espacios iniciales (`LTrim`), sin espacios finales (`RTrim`) o sin e iniciales ni finales (`Trim`).
MemberwiseClone()	Crea una copia superficial del Object actual. (Inherited from Object)
Mid(String, Int32)	Devuelve una cadena que contiene todos los caracteres a partir d posición especificada de una cadena.
Mid(String, Int32, Int32)	Devuelve una cadena que contiene un número de caracteres esp a partir de una posición especificada de una cadena.
Replace(String, String, String, Int32, Int32, CompareMethod)	Devuelve una cadena en la que la subcadena especificada se reem determinado número de veces por otra subcadena.
Right(String, Int32)	Devuelve una cadena que contiene un número especificado de c desde el lado derecho de una cadena.
RSet(String, Int32)	Devuelve una cadena alineada a la derecha que contiene la cade especificada y con la longitud especificada.
RTrim(String)	Devuelve una cadena que contiene una copia de una cadena esp sin espacios iniciales (`LTrim`), sin espacios finales (`RTrim`) o sin e iniciales ni finales (`Trim`).
Space(Int32)	Devuelve una cadena que consta del número especificado de esp
Split(String, String, Int32, CompareMethod)	Devuelve una matriz unidimensional basada en cero que contien número especificado de subcadenas.

StrComp(String, String, CompareMethod)	Devuelve un valor, -1, 0 ó 1, que indica el resultado de una comp de cadena.
StrConv(String, VbStrConv, Int32)	Devuelve una cadena convertida según se ha especificado.
StrDup(Int32, Char)	Devuelve una cadena o un objeto que se compone del carácter especificado repetido el número de veces especificado.
StrDup(Int32, Object)	Devuelve una cadena o un objeto que se compone del carácter especificado repetido el número de veces especificado.
StrDup(Int32, String)	Devuelve una cadena o un objeto que se compone del carácter especificado repetido el número de veces especificado.
StrReverse(String)	Devuelve una cadena en la que se invierte el orden de los caracte cadena especificada.
ToString()	Devuelve una cadena que representa el objeto actual. (Inherited from Object)
Trim(String)	Devuelve una cadena que contiene una copia de una cadena esp sin espacios iniciales (`LTrim`), sin espacios finales (`RTrim`) o sin e iniciales ni finales (`Trim`).
UCase(Char)	Devuelve una cadena o un carácter que contiene la cadena espec convertida en mayúsculas.
UCase(String)	Devuelve una cadena o un carácter que contiene la cadena espec convertida en mayúsculas.