

Dim (Instrucción)

Declara [variables](#) y les asigna espacio de almacenamiento.

Sintaxis

Dim [**WithEvents**] *nombre_variable*[(*subíndices*)] [**As** [**New**] *tipo*] [, [**WithEvents**] *nombre_variable*[(*subíndices*)] [**As** [**New**] *tipo*]] ...

La sintaxis de la instrucción **Dim** consta de las siguientes partes:

| Parte | Descripción |
|------------------------|--|
| WithEvents | Opcional. Palabra clave que especifica que <i>nombre_variable</i> es una variable de objeto utilizada para responder a eventos desencadenados por un objeto ActiveX . WithEvents solamente es válido en módulos de clase . Puede declarar tantas variables individuales como desee mediante WithEvents , pero no puede crear matrices con WithEvents . No puede utilizar New con WithEvents . |
| <i>nombre_variable</i> | Requerido. Nombre de la variable; sigue las convenciones estándar de nombre de variable. |
| <i>subíndices</i> | Opcional. Dimensiones de la variable de matriz; se pueden declarar hasta 60 dimensiones múltiples. El argumento <i>subíndices</i> utiliza la sintaxis siguiente: [<i>inferior To superior</i>] [, [<i>inferior To superior</i>] ... Cuando no se declara específicamente en <i>inferior</i> , el límite inferior de una matriz se controla mediante la instrucción Option Base . Este límite inferior es cero si no hay ninguna instrucción Option Base . |
| New | Opcional. Palabra clave que habilita la creación implícita de un objeto. Si utiliza New cuando declara la variable de objeto, se crea una nueva instancia del objeto como primera referencia, de forma que no tiene que utilizar la instrucción Set para asignar la referencia del objeto. La palabra clave New no se puede utilizar para declarar variables de cualquier tipo de datos , intrínseco, para declarar instancias de objetos dependientes ni con WithEvents . |
| <i>tipo</i> | Opcional. Tipo de datos de la variable; puede ser Byte , Boolean , Integer , Long , Currency , Single , Double , Decimal (actualmente no admitida), Date , String (para cadenas de longitud variable), String * <i>length</i> (para cadenas de longitud fija), Object , Variant , un tipo definido por el usuario , o un tipo de objeto . Utilice una cláusula distinta As tipo para cada variable que defina. |

Comentarios

Las variables declaradas con **Dim** en el [nivel de módulo](#) están disponibles para todos los procedimientos disponibles sólo dentro de ese [módulo](#). En el [nivel de procedimiento](#), las variables sólo están disponibles dentro de ese procedimiento.

Utilice la instrucción **Dim** en el nivel de módulo o de procedimiento para declarar el tipo de datos de una variable. Por ejemplo, la siguiente instrucción declara una variable como **Integer**.

```
Dim NúmeroDeEmpleados As Integer
```

También puede utilizar una instrucción **Dim** para declarar el tipo de objeto de una variable. La siguiente línea declara una variable para una nueva instancia de una hoja de cálculo.

```
Dim X As New Worksheet
```

Si no utiliza la palabra clave **New** al declarar una variable de objeto, la variable que se refiere al objeto debe asignarse a un objeto existente mediante la instrucción **Set** antes de su uso. Hasta que se le asigne un objeto, la variable de objeto declarada tiene el valor especial **Nothing**, el cual indica que no se refiere a ninguna instancia en particular de un objeto.

También puede utilizar la instrucción **Dim** con paréntesis vacíos para declarar matrices dinámicas. Después de declarar una matriz dinámica, use la instrucción **ReDim** dentro de un procedimiento para definir el número de dimensiones y elementos de la matriz. Si intenta volver a declarar una dimensión para una variable de matriz cuyo tamaño se ha especificado explícitamente en una instrucción **Private**, **Public** o **Dim**, ocurrirá un error.

Si no especifica un tipo de datos o un tipo de objeto y no existe ninguna instrucción **DefTipo** en el módulo, la variable predeterminada será **Variant**.

Cuando se inicializan variables, una variable numérica se inicializa con 0, una cadena de longitud variable se inicializa con una cadena de longitud 0 ("") y una cadena de longitud fija se llena con ceros. Las variables **Variant** se inicializan con [Empty](#). Cada elemento de una variable de un tipo definido por el usuario se inicializa como si fuera una variable distinta.

Nota Cuando utiliza la instrucción **Dim** en un procedimiento, generalmente pone la instrucción **Dim** al principio del mismo.

Unload (Instrucción)

Descarga de memoria un formulario o un control.

Sintaxis

Unload *objeto*

El marcador de posición *objeto* es el nombre de un objeto **Form** o de un elemento de una [matriz de controles](#) para descargar.

Comentarios

La descarga de un formulario o de un control puede ser necesaria o conveniente en aquellos casos en los que la memoria utilizada sea necesaria para alguna otra tarea o cuando sea necesario restablecer las propiedades a sus valores originales.

Antes de descargar un formulario se produce el evento Query_Unload, seguido del procedimiento de evento Form_Unload. Si establece el argumento *cancelar* a **True** en alguno de estos eventos no se descargará el formulario. En los objetos **MDIForm** se produce el procedimiento de evento Query_Unload del objeto **MDIForm**, seguido del procedimiento de evento Query_Unload y del procedimiento de evento Form_Unload de cada formulario [secundario MDI](#); finalmente se produce el procedimiento de evento Form_Unload del objeto **MDIForm**.

Cuando se descarga un formulario, todos los controles colocados en el formulario en [tiempo de ejecución](#) no son accesibles. Los controles colocados en el formulario en [tiempo de diseño](#) permanecen intactos; sin embargo, cualquier cambio en tiempo de ejecución sobre dichos controles y sus propiedades se pierden cuando se vuelve a cargar el formulario. También se pierden todos los cambios realizados en las propiedades del formulario. El acceso a algún control del formulario hace que éste se vuelva a cargar.

Nota Cuando se descarga un formulario, sólo se descarga el componente mostrado. El código asociado al módulo del formulario permanece en memoria.

Con la instrucción **Unload** sólo se pueden descargar los elementos de las matrices de controles agregados a un formulario en tiempo de ejecución. Las propiedades de los controles descargados se reinician cuando se cargan de nuevo los controles.

MsgBox (Función)

Muestra un mensaje en un cuadro de diálogo, espera a que el usuario haga clic en un botón y devuelve un tipo **Integer** correspondiente al botón elegido por el usuario.

Sintaxis

MsgBox(*prompt*[, *buttons*][, *title*][, *helpfile*, *context*])

La sintaxis de la función **MsgBox** consta de estos [argumentos con nombre](#):

| Parte | Descripción |
|-----------------|--|
| <i>prompt</i> | Requerido. Expresión de cadena que representa el <i>prompt</i> en el cuadro de diálogo. La longitud máxima de <i>prompt</i> es de aproximadamente 1024 caracteres, según el ancho de los caracteres utilizados. Si <i>prompt</i> consta de más de una línea, puede separarlos utilizando un carácter de retorno de carro (Chr (13)) o un carácter de avance de línea (Chr (10)), o una combinación de caracteres de retorno de carro – avance de línea (Chr (13) y Chr (10)) entre cada línea y la siguiente. |
| <i>buttons</i> | Opcional. Expresión numérica que corresponde a la suma de los valores que especifican el número y el tipo de los botones que se pretenden mostrar, el estilo de icono que se va a utilizar, la identidad del botón predeterminado y la modalidad del cuadro de mensajes. Si se omite este argumento, el valor predeterminado para <i>buttons</i> es 0. |
| <i>title</i> | Opcional. Expresión de cadena que se muestra en la barra de título del cuadro de diálogo. Si se omite <i>title</i> , en la barra de título se coloca el nombre de la aplicación. |
| <i>helpfile</i> | Opcional. Expresión de cadena que identifica el archivo de Ayuda que se utiliza para proporcionar ayuda interactiva en el cuadro de diálogo. Si se especifica <i>helpfile</i> , también se debe especificar <i>context</i> . |
| <i>context</i> | Opcional. Expresión numérica que es igual al número de contexto de Ayuda asignado por el autor al tema de Ayuda correspondiente. Si se especifica <i>context</i> , también se debe especificar <i>helpfile</i> . |

Valores

El [argumento](#) *buttons* tiene estos valores:

| Constante | Valor | Descripción |
|-----------------|-------|---|
| VbOKOnly | 0 | Muestra solamente el botón Aceptar . |

| | | |
|------------------------------|---------|---|
| VbOKCancel | 1 | Muestra los botones Aceptar y Cancelar . |
| VbAbortRetryIgnore | 2 | Muestra los botones Anular , Reintentar e Ignorar . |
| VbYesNoCancel | 3 | Muestra los botones Sí , No y Cancelar . |
| VbYesNo | 4 | Muestra los botones Sí y No . |
| VbRetryCancel | 5 | Muestra los botones Reintentar y Cancelar . |
| VbCritical | 16 | Muestra el icono de mensaje crítico . |
| VbQuestion | 32 | Muestra el icono de pregunta de advertencia . |
| VbExclamation | 48 | Muestra el icono de mensaje de advertencia . |
| VbInformation | 64 | Muestra el icono de mensaje de información . |
| VbDefaultButton1 | 0 | El primer botón es el predeterminado. |
| VbDefaultButton2 | 256 | El segundo botón es el predeterminado. |
| VbDefaultButton3 | 512 | El tercer botón es el predeterminado. |
| VbDefaultButton4 | 768 | El cuarto botón es el predeterminado. |
| VbApplicationModal | 0 | Aplicación modal; el usuario debe responder al cuadro de mensajes antes de poder seguir trabajando en la aplicación actual. |
| VbSystemModal | 4096 | Sistema modal; se suspenden todas las aplicaciones hasta que el usuario responda al cuadro de mensajes. |
| VbMsgBoxHelpButton | 16384 | Agrega el botón Ayuda al cuadro de mensaje. |
| VbMsgBoxSetForeground | 65536 | Especifica la ventana del cuadro de mensaje como la ventana de primer plano. |
| VbMsgBoxRight | 524288 | El texto se alinea a la derecha. |
| VbMsgBoxRtlReading | 1048576 | Especifica que el texto debe aparecer para ser leído de derecha a izquierda en sistemas hebreo y árabe. |

El primer grupo de valores (0 a 5) describe el número y el tipo de los botones mostrados en el cuadro de diálogo; el segundo grupo (16, 32, 48, 64) describe el estilo del icono, el

tercer grupo (0, 256, 512) determina el botón predeterminado y el cuarto grupo (0, 4096) determina la modalidad del cuadro de mensajes. Cuando se suman números para obtener el valor final del argumento *buttons*, se utiliza solamente un número de cada grupo.

Nota Estas [constantes](#) las especifica Visual Basic for Applications. Por tanto, el nombre de las mismas puede utilizarse en cualquier lugar del código en vez de sus valores reales.

Valores devueltos

| Constante | Valor | Descripción |
|-----------------|-------|-------------------|
| vbOK | 1 | Aceptar |
| vbCancel | 2 | Cancelar |
| vbAbort | 3 | Anular |
| vbRetry | 4 | Reintentar |
| vbIgnore | 5 | Ignorar |
| vbYes | 6 | Sí |
| vbNo | 7 | No |

Comentarios

Cuando se proporcionan tanto *helpfile* como *context*, el usuario puede presionar F1 para ver el tema de Ayuda correspondiente al *context*. Algunas [aplicaciones host](#), por ejemplo Microsoft Excel, también agregan automáticamente un botón **Ayuda** al cuadro de diálogo.

Si el cuadro de diálogo cuenta con un botón **Cancelar**, presionar la tecla ESC tendrá el mismo efecto que hacer clic en este botón. Si el cuadro de diálogo contiene un botón **Ayuda**, se suministra ayuda interactiva para ese cuadro de diálogo. Sin embargo, no se devuelve valor alguno hasta que se hace clic en uno de estos botones.

Nota Si desea especificar más que el primer argumento con nombre, debe utilizar **MsgBox** en una [expresión](#). Si desea omitir algún [argumento](#) de posición, debe incluir el delimitador de coma correspondiente.

If...Then...Else (Instrucción)

Ejecuta condicionalmente un grupo de [instrucciones](#), dependiendo del valor de una [expresión](#).

Sintaxis

If *condición* **Then** [*instrucciones*]-**[Else** *instrucciones_else*]

Puede utilizar la siguiente sintaxis en formato de bloque:

If *condición* **Then**
[*instrucciones*]

[ElseIf *condición-n* **Then**
[*instrucciones_elseif*] . . .

[Else
[*instrucciones_else*]]

End If

La sintaxis de la instrucción **If...Then...Else** consta de tres partes:

| Parte | Descripción |
|-----------------------------|---|
| <i>condición</i> | Requerido. Uno o más de los siguientes dos tipos de expresiones: |
| | Una expresión numérica o expresión de cadena que puede ser evaluada como True o False . Si <i>condición</i> es Null , <i>condición</i> se considera False . |
| | Una expresión del formulario TypeOf <i>nombre_objeto</i> Is <i>tipo_objeto</i> . El <i>nombre_objeto</i> es cualquier referencia al objeto y <i>tipo_objeto</i> es cualquier tipo de objeto válido. La expresión es True si <i>nombre_objeto</i> es del tipo de objeto especificado por <i>tipo_objeto</i> ; en caso contrario es False . |
| <i>instrucciones</i> | Opcional en formato de bloque; se requiere en formato de línea sencilla que no tenga una cláusula Else . Una o más instrucciones separadas por dos puntos ejecutados si la <i>condición</i> es True . |
| <i>condición-n</i> | Opcional. Igual que <i>condición</i> . |
| <i>instrucciones_elseif</i> | Opcional. Una o más instrucciones ejecutadas si la <i>condición-n</i> asociada es True . |
| <i>instrucciones_else</i> | Opcional. Una o más instrucciones ejecutadas si ninguna de las |

| | |
|--|---|
| | expresiones anteriores <i>condición</i> o <i>condición-n</i> es True . |
|--|---|

Comentarios

Puede utilizar la forma de una sola línea (Sintaxis 1) para pruebas cortas y sencillas. Sin embargo, el formato de bloque (Sintaxis 2) proporciona más estructura y flexibilidad que la forma de línea simple y, generalmente, es más fácil de leer, de mantener y de depurar.

Nota Con la sintaxis es posible ejecutar múltiples *instrucciones* como resultado de una decisión **If...Then**, pero todas deben estar en la misma línea y separadas por dos puntos, como en la instrucción siguiente:

```
If A > 10 Then A = A + 1 : B = B + A : C = C + B
```

Una instrucción con formato de bloque **If** debe ser la primera de la línea. Las partes **Else**, **ElseIf** y **End If**, de la instrucción, solamente pueden ir precedidas de un [número de línea](#) o una [etiqueta de línea](#). El bloque **If** debe terminar con una instrucción **End If**.

Para determinar si una instrucción **If** es un bloque, examine lo que sigue a la [palabra Then](#). Si lo que aparece detrás de **Then** en la misma línea no es un [comentario](#), la instrucción se considera como una instrucción **If** de una sola línea.

Las cláusulas **Else** y **ElseIf** son opcionales. Puede tener en un bloque **ElseIf**, tantas cláusulas **If** como desee, pero ninguna puede aparecer después de una cláusula **Else**. Las instrucciones de bloque **If** se pueden anidar; es decir, unas pueden contener a otras.

Cuando se ejecuta un bloque **If** (Sintaxis 2), se prueba *condición*. Si *condición* es **True**, se ejecutan las instrucciones que están a continuación de **Then**. Si *condición* es **False**, se evalúan una a una las condiciones **ElseIf** (si existen). Cuando se encuentra una condición **True** se ejecutan las instrucciones que siguen inmediatamente a la instrucción **Then** asociada. Si ninguna de las condiciones **ElseIf** es **True** (o si no hay cláusulas **ElseIf**), se ejecutan las instrucciones que siguen a **Else**. Después de la ejecución de las instrucciones que siguen a **Then** o **Else**, la ejecución continúa con la instrucción que sigue a **End If**.

Sugerencia **Select Case** puede ser más útil cuando se evalúa una única expresión que tiene varias acciones posibles. Sin embargo, la cláusula **TypeOf nombre_objeto Is tipo_objeto** no se puede utilizar en una instrucción **Select Case**.

Nota No se puede usar **TypeOf** con tipos de datos predefinidos como Long, Integer y así sucesivamente, excepto en el tipo de datos Object.

Select Case (Instrucción)

Ejecuta uno de varios grupos de [instrucciones](#), dependiendo del valor de una [expresión](#).

Sintaxis

```
Select Case expresión_prueba  
[Case lista_expresión-n  
[instrucciones-n]] . . .  
[Case Else  
[instrucciones_else]]
```

End Select

La sintaxis de la instrucción **Select Case** consta de las siguientes partes:

| Parte | Descripción |
|---------------------------|---|
| <i>expresión_prueba</i> | Requerido. Cualquier expresión numérica o expresión de cadena . |
| <i>lista_expresión-n</i> | Requerido si aparece la palabra clave Case . Lista delimitada por comas de una o más de las formas siguientes: <i>expresión</i> , <i>expresión To expresión</i> , <i>Is expresión operador_de_comparación</i> . La palabra clave especifica un intervalo de valores. Si se utiliza la palabra clave To , el valor menor debe aparecer antes de To . Utilice la palabra clave Is con operadores de comparación (excepto Is y Like) para especificar un intervalo de valores. Si no se escribe, la palabra clave Is se insertará automáticamente. |
| <i>instrucciones-n</i> | Opcional. Una o más instrucciones ejecutadas si <i>expresión_prueba</i> coincide con cualquier parte de <i>lista_expresión-n</i> . |
| <i>instrucciones_else</i> | Opcional. Una o más instrucciones que se ejecuten si <i>expresión_prueba</i> no coincide con nada de la cláusula Case . |

Comentarios

Si *expresión_prueba* coincide con cualquier *lista_expresión* asociada con una cláusula **Case**, las instrucciones que siguen a esa cláusula **Case** se ejecutan hasta la siguiente cláusula **Case** o, para la última cláusula, hasta la instrucción **End Select**. El control pasa después a la instrucción que sigue a **End Select**. Si *expresión_prueba* coincide con una expresión de *lista_expresión* en más de una cláusula **Case**, sólo se ejecutan las instrucciones que siguen a la primera coincidencia.

La cláusula **Case Else** se utiliza para indicar las instrucciones que se van a ejecutar si no se encuentran coincidencias entre *expresión_prueba* y una *lista_expresión* en cualquiera de las otras selecciones de **Case**. Aunque no es necesario, es buena idea tener una instrucción **Case Else** en el bloque **Select Case** para controlar valores imprevistos de *expresión_prueba*. Cuando no hay una instrucción **Case Else** y ninguna expresión de la lista en las cláusulas **Case** coincide con la expresión de prueba, la ejecución continúa en la instrucción que sigue a **End Select**.

Se pueden utilizar expresiones múltiples o intervalos en cada cláusula **Case**. Por ejemplo, la línea siguiente es válida:

```
Case 1 To 4, 7 To 9, 11, 13, Is > MaxNumber
```

Nota El operador de comparación **Is** no es lo mismo que la palabra clave **Is** utilizada en la instrucción **Select Case**.

También puede especificar intervalos y expresiones múltiples para cadenas de caracteres. En el siguiente ejemplo, **Case** coincide con las cadenas que son exactamente iguales a todo, cadenas que están entre nueces y sopa en orden alfabético y el valor actual de `ElemPrueba`:

```
Case "iguales a todo", "nueces" To "sopa", ElemPrueba
```

Las instrucciones **Select Case** se pueden anidar. Cada instrucción **Select Case** debe tener su correspondiente instrucción **End Select**.

Sub (Instrucción)

Declara el nombre, los [argumentos](#), y el código que componen el cuerpo de un [procedimiento Sub](#).

Sintaxis

[**Private** | **Public** | **Friend**] [**Static**] **Sub** *nombre* [(*lista_argumentos*)]
[*instrucciones*]
[**Exit Sub**]
[*instrucciones*]

End Sub

La sintaxis de la instrucción **Sub** consta de las siguientes partes:

| Parte | Descripción |
|-------------------------|---|
| Public | Opcional. Indica que el procedimiento Sub es accesible para todos los demás procedimientos de todos los módulos . Si se usa en un módulo que contiene una instrucción Option Private , el procedimiento no está disponible fuera del proyecto . |
| Private | Opcional. Indica que el procedimiento Sub es accesible sólo para otros procedimientos del módulo en el que se declara. |
| Friend | Opcional. Se utiliza solamente en un módulo de clase . Indica que el procedimiento Sub es visible a través del proyecto , pero no por un controlador de una instancia de un objeto. |
| Static | Opcional. Indica que las variables locales del procedimiento Sub se conservan entre distintas llamadas. El atributo Static no afecta a las variables que se declaran fuera de Sub , incluso aunque se usen en el procedimiento. |
| <i>nombre</i> | Requerido. Nombre del procedimiento Sub ; sigue las convenciones estándar de nombres de variable . |
| <i>lista_argumentos</i> | Opcional. Lista de variables que representan los argumentos que se pasan al procedimiento Sub cuando se le llama. Las distintas variables se separan mediante comas. |
| <i>instrucciones</i> | Opcional. Cualquier grupo de instrucciones que se ejecutan dentro del procedimiento Sub . |

El argumento *lista_argumentos* consta de las siguientes partes y sintaxis:

[**Optional**] [**ByVal** | **ByRef**] [**ParamArray**] *nombre_variable*[()] [**As** *tipo*] [= *valor_predeterminado*]

| Parte | Descripción |
|-----------------------------|---|
| Optional | Opcional. Palabra clave que indica que no se requiere ningún argumento. Si se usa, todos los argumentos subsiguientes de <i>lista_argumentos</i> también deben ser opcionales y declararse mediante la palabra clave Optional . Optional no se puede utilizar para ningún argumento si se usa ParamArray . |
| ByVal | Opcional. Indica que el argumento se pasa por valor . |
| ByRef | Opcional. Indica que el argumento se pasa por referencia . ByRef es el modo predeterminado en Visual Basic. |
| ParamArray | Opcional. Sólo se utiliza como el último argumento de <i>lista_argumentos</i> para indicar que el argumento final es una matriz Optional de elementos tipo Variant . La palabra clave ParamArray le permite proporcionar un número arbitrario de argumentos. No se puede utilizar con ByVal , ByRef u Optional . |
| <i>nombre_variable</i> | Requerido. Nombre de la variable que representa el argumento; sigue las convenciones estándar de nombres de variables. |
| <i>tipo</i> | Opcional. El tipo de datos del argumento que se pasa al procedimiento; puede ser Byte , Boolean , Integer , Long , Currency , Single , Double , Decimal (no admitido actualmente), Date , String (solamente longitud variable), Object , Variant , o un tipo de objeto específico. Si el parámetro no es Optional , se puede especificar también un tipo definido por el usuario . |
| <i>valor_predeterminado</i> | Opcional. Cualquier constante o expresión de constante. Sólo es válido para parámetros Optional . Si el tipo es Object , un valor predeterminado explícito sólo puede ser Nothing . |

Comentarios

Si no se especifica explícitamente mediante **Public**, **Private** o **Friend**, los procedimientos **Sub** son públicos de manera predeterminada. Si no se usa **Static**, el valor de las variables locales no se mantiene entre distintas llamadas. La palabra clave **Friend** solamente se puede usar en módulos de clase. Sin embargo, los procedimientos en cualquier módulo de un proyecto pueden acceder a los procedimientos **Friend**. Un procedimiento **Friend** no aparece en la [biblioteca de tipo](#) de su clase primaria, ni se puede enlazar posteriormente.

Precaución Los procedimientos **Sub** pueden ser recursivos; es decir, se pueden llamar a sí mismos para realizar una tarea determinada. Sin embargo, esto puede llevar al desbordamiento de la pila. La palabra clave **Static** generalmente no se utiliza con procedimientos recursivos **Sub**.

Todo código ejecutable debe estar en [procedimientos](#). No puede definir un procedimiento **Sub** dentro de otro procedimiento **Sub**, **Function** o **Property**.

Las palabras clave **Exit Sub** causan la inmediata salida de un procedimiento **Sub**. La ejecución del programa continúa con la instrucción que sigue a la instrucción que llamó el procedimiento **Sub**. Cualquier número de instrucciones **Exit Sub** puede aparecer en cualquier lugar de un procedimiento **Sub**.

Al igual que un procedimiento **Function**, un procedimiento **Sub** es un procedimiento distinto que toma argumentos, lleva a cabo una serie de instrucciones y cambia el valor de sus argumentos. Sin embargo, a diferencia de un procedimiento **Function**, el cual devuelve un valor, un procedimiento **Sub** no se puede utilizar en una expresión.

Para llamar a un procedimiento **Sub**, use el nombre del procedimiento seguido de la lista de argumentos. Consulte la instrucción **Call** para obtener información específica acerca de cómo llamar a los procedimientos **Sub**.

Las variables usadas en procedimientos **Sub** se dividen en dos categorías: las que están explícitamente declaradas dentro del procedimiento y las que no lo están. Las variables declaradas explícitamente en un procedimiento (mediante **Dim** o un equivalente) siempre son locales del procedimiento. Otras variables usadas pero no declaradas explícitamente en un procedimiento también son locales, a menos que se declaren explícitamente en algún nivel superior fuera del procedimiento.

Precaución Un procedimiento puede usar una variable que no esté declarada explícitamente en el procedimiento, pero puede ocurrir un conflicto de nombres si cualquier cosa que ha definido en el [nivel de módulo](#) tiene el mismo nombre. Si el procedimiento se refiere a una variable no declarada que tiene el mismo nombre que otro procedimiento, constante o variable, se supone que el procedimiento se está refiriendo al nombre de ese nivel de módulo. Para evitar este tipo de conflictos, declare las variables explícitamente. Puede usar una instrucción **Option Explicit** para forzar la declaración explícita de variables.

Nota No se puede usar **GoSub**, **GoTo** o **Return** para obtener acceso o salir de un procedimiento **Sub**.

Chr (Función)

Devuelve un tipo [String](#) que contiene el carácter asociado con el [código de carácter](#) especificado.

Sintaxis

Chr(*códigocar*)

El [argumento](#) *códigocar* es un tipo [Long](#) que identifica a un carácter.

Comentarios

Los números del 0 al 31 son los mismos que los códigos [ASCII](#) estándar no imprimibles. Por ejemplo, **Chr**(10) devuelve un carácter de avance de línea. El intervalo normal de *códigocar* es 0–255. Sin embargo, en sistemas [DBCS](#), el intervalo real de *códigocar* es de -32768 a 65535.

Nota La función **ChrB** se utiliza con datos de byte incluidos en un tipo **String**. En lugar de devolver un carácter, que puede ser de uno o de dos bytes, **ChrB** siempre devuelve un único byte. La función **ChrW** devuelve un tipo **String** que contiene el carácter [Unicode](#) excepto en plataformas donde no se admite Unicode, en cuyo caso, el comportamiento es idéntico al de la función **Chr**.

IsNumeric (Función)

Devuelve un valor de tipo **Boolean** que indica si una [expresión](#) se puede evaluar como un número.

Sintaxis

IsNumeric(*expresión*)

El [argumento](#) *expresión* requerido, es un tipo de datos [Variant](#) que contiene una [expresión numérica](#) o una [expresión de tipo cadena](#).

Comentarios

La función **IsNumeric** devuelve **True** si la *expresión* completa se reconoce como un número; en otro caso, devuelve **False**.

La función **IsNumeric** devuelve **False** si *expresión* es una [expresión de fecha](#).

IsDate (Función)

Devuelve un valor de tipo **Boolean** que indica si una [expresión](#) se puede convertir en una fecha.

Sintaxis

IsDate(*expresión*)

El [argumento](#) *expresión* requerido, es un tipo de datos [Variant](#) que contiene una [expresión de fecha](#) o una [expresión de cadena](#) reconocible como una fecha o una hora.

Comentarios

La función **IsDate** devuelve **True** si la expresión es una fecha o se puede reconocer como una fecha válida; en caso contrario, devuelve **False**. En Microsoft Windows, el intervalo de fechas válidas va desde el 1 de enero de 100 D. de C. hasta el 31 de diciembre de 9999 D.de C.; los intervalos varían de un sistema operativo a otro.

Format (Función)

Devuelve un tipo **Variant (String)** que contiene una [expresión](#) formateada de acuerdo a las instrucciones contenidas en una expresión de formato.

Sintaxis

Format(*expresión*[, *formato*[, *primerdíadesemana*[, *primerdíadeaño*]]])

La sintaxis de la función **Format** consta de las siguientes partes:

| Parte | Descripción |
|--------------------------|--|
| <i>expresión</i> | Requerido. Cualquier expresión válida. |
| <i>formato</i> | Opcional. Una expresión de formato definida por el usuario o con nombre válida. |
| <i>primerdíadesemana</i> | Opcional. Una constante que especifica el primer día de la semana. |
| <i>primerdíadeaño</i> | Opcional. Una constante que especifica la primera semana del año. |

Valores

El [argumento](#) *primerdíadesemana* tiene estos valores:

| Constante | Valor | Descripción |
|--------------------|-------|------------------------------|
| vbUseSystem | 0 | Utiliza el valor de API NLS. |
| vbSunday | 1 | Domingo (predeterminado) |
| vbMonday | 2 | Lunes |
| vbTuesday | 3 | Martes |
| vbWednesday | 4 | Miércoles |
| vbThursday | 5 | Jueves |
| vbFriday | 6 | Viernes |
| vbSaturday | 7 | Sábado |

El argumento *primerdíadeaño* tiene estos valores:

| Constante | Valor | Descripción |
|------------------------|-------|---|
| vbUseSystem | 0 | Utiliza el valor de API NLS. |
| vbFirstJan1 | 1 | Comienza con la semana donde está el 1 de enero (predeterminado). |
| vbFirstFourDays | 2 | Comienza con la primera semana del año que tenga cuatro días como mínimo. |
| vbFirstFullWeek | 3 | Comienza con la primera semana completa del año. |

Comentarios

| Para dar formatp | Haga esto |
|----------------------------------|--|
| Números | Utilice formatos numéricos con nombre predefinidos o cree formatos numéricos definidos por el usuario. |
| Fechas y horas | Utilice formatos de fecha/hora con nombre predefinidos o cree formatos de fecha/hora definidos por el usuario. |
| Números seriales de fecha y hora | Utilice formatos de fecha y hora o formatos numéricos. |
| Cadenas | Cree sus propios formatos de cadena definidos por el usuario. |

Si intenta dar formato a un número sin especificar *formato*, **Format** proporciona una funcionalidad similar a la de la función **Str**. Sin embargo, los números positivos a los que se les ha dado formato de cadena de caracteres mediante **Format** carecen del espacio inicial reservado para mostrar el signo del valor, mientras que los convertidos utilizando **Str** conservan el espacio inicial.

Funciones de conversión de tipos

Cada función convierte una [expresión](#) a un [tipo de datos](#) específico.

Sintaxis

CBool(*expresión*)

CByte(*expresión*)

CCur(*expresión*)

CDate(*expresión*)

CDBl(*expresión*)

CDec(*expresión*)

CInt(*expresión*)

CLng(*expresión*)

CSng(*expresión*)

CStr(*expresión*)

CVar(*expresión*)

El [argumento](#) obligatorio *expresión* es cualquier [expresión de cadena](#) o [expresión numérica](#).

Tipos devueltos

El nombre de la función determina el tipo devuelto, como se muestra a continuación:

| Función | Tipo devuelto | Intervalo del argumento <i>expresión</i> |
|--------------|--------------------------|---|
| Cbool | Boolean | Cualquier expresión de cadena o numérica válida. |
| Cbyte | Byte | 0 a 255. |
| Ccur | Currency | -922.337.203.685.477,5808 a 922.337.203.685.477,5807. |
| Cdate | Date | Cualquier expresión de fecha . |

Cuando la parte fraccionaria es exactamente 0,5, **CInt** y **CLng** siempre redondean al número par más cercano. Por ejemplo, 0,5 redondea a 0, y 1,5 redondea a 2. **CInt** y **CLng** se diferencian de las funciones **Fix** y **Int** en que truncan, en lugar de redondear, la parte fraccionaria de un número. Además, **Fix** y **Int** siempre devuelven un valor del mismo tipo del que se le pasa.

Utilice la función **IsDate** para determinar si se puede convertir *date* a una fecha o una hora. **CDate** reconoce [literales de fecha](#) y literales de hora además de números comprendidos dentro del intervalo de fechas aceptables. Al convertir un número a una fecha, la parte numérica entera se convierte a una fecha. Cualquier parte fraccionaria del número se convierte a la hora del día, comenzando a medianoche.

CDate reconoce formatos de fecha que se ajusten a la configuración regional de su sistema. Es posible que no se determine el orden correcto del día, mes y año si se proporciona en un formato diferente del que reconoce la configuración de fecha. Además, no se puede reconocer un formato de fecha largo si contiene la cadena del día de la semana.

Se proporciona una función **CVDate** por compatibilidad con versiones anteriores de Visual Basic. La sintaxis de la función **CVDate** es idéntica a la de la función **CDate**; sin embargo, **CVDate** devuelve un **Variant** de subtipo **Date** en lugar del tipo **Date** real. Puesto que ahora hay un tipo de dato intrínseco **Date**, no es necesario **CVDate**. Se puede observar el mismo efecto al convertir una expresión a **Date** y asignarla después a un **Variant**. Esta técnica es coherente con la conversión de todos los demás tipos intrínsecos a sus equivalentes subtipos **Variant**.

Nota La función **CDec** no devuelve un tipo de dato discreto; en su lugar, siempre devuelve un **Variant** con los valores convertidos a un subtipo **Decimal**.

For...Next (Instrucción)

Repite un grupo de [instrucciones](#) un número especificado de veces.

Sintaxis

For *contador* = *principio* **To** *fin* [**Step** *incremento*]
[*instrucciones*]
Exit For
[*instrucciones*]

Next [*contador*]

La sintaxis de la instrucción **For...Next** consta de las siguientes partes:

| Parte | Descripción |
|----------------------|--|
| <i>contador</i> | Requerido. Variable numérica que se utiliza como contador de bucle. La variable no puede ser Booleana ni un elemento de matriz . |
| <i>principiot</i> | Requerido. Valor inicial del <i>contador</i> . |
| <i>fin</i> | Requerido. Valor final del <i>contador</i> . |
| <i>incremento</i> | Opcional. Cantidad en la que cambia el <i>contador</i> cada vez que se ejecuta el bucle. Si no se especifica, el valor predeterminado de <i>incremento</i> es uno. |
| <i>instrucciones</i> | Opcional. Una o más instrucciones entre For y Next que se ejecutan un número especificado de veces. |

Comentarios

El [argumento](#) *incremento* puede ser positivo o negativo. El valor del argumento *incremento* determina la manera en que se procesa el bucle, como se muestra a continuación:

| Valor | El bucle se ejecuta si |
|--------------|-------------------------------|
| Positivo o 0 | <i>contador</i> <= <i>fin</i> |
| Negativo | <i>contador</i> >= <i>fin</i> |

Una vez que se inicia el bucle y se han ejecutado todas las instrucciones en el bucle, *incremento* se suma a *contador*. En este punto, las instrucciones del bucle se pueden ejecutar de nuevo (si se cumple la misma prueba que causó que el bucle se ejecutara inicialmente) o bien se sale del bucle y la ejecución continúa con la instrucción que sigue a la instrucción **Next**.

Sugerencia Cambiar el valor de *contador* mientras está dentro de un bucle hace difícil su lectura y depuración.

Se pueden colocar en el bucle cualquier número de instrucciones **Exit For** como una manera alternativa de salir del mismo. La instrucción **Exit For**, que se utiliza a menudo en la evaluación de alguna condición (por ejemplo, **If...Then**), transfiere el control a la instrucción que sigue inmediatamente a la instrucción **Next**.

Se pueden anidar bucles **For...Next**, colocando un bucle **For...Next** dentro de otro. Para ello, proporcione a cada bucle un nombre de variable único como su *contador*. La siguiente construcción es correcta:

```
For I = 1 To 10
  For J = 1 To 10
    For K = 1 To 10
      ...
    Next K
  Next J
Next I
```

Nota Si omite un *contador* en una instrucción **Next**, la ejecución continúa como si se hubiera incluido. Se produce un error si se encuentra una instrucción **Next** antes de su instrucción **For** correspondiente.

For Each...Next (Instrucción)

Repite un grupo de [instrucciones](#) para cada elemento de una [matriz](#) o [colección](#).

Sintaxis

For Each *elemento* **In** *grupo*
[*instrucciones*]
[Exit For]
[*instrucciones*]

Next [*elemento*]

La sintaxis de la instrucción **For Each...Next** consta de las siguientes partes:

| Parte | Descripción |
|----------------------|--|
| <i>elemento</i> | Requerido. Variable que se utiliza para iterar por los elementos del conjunto o matriz. Para conjuntos, <i>elemento</i> solamente puede ser una variable del tipo Variant , una variable de objeto genérica o cualquier variable de objeto específica. Para matrices, <i>elemento</i> solamente puede ser una variable tipo Variant . |
| <i>grupo</i> | Requerido. Nombre de un conjunto de objetos o de una matriz (excepto una matriz de tipos definidos por el usuario). |
| <i>instrucciones</i> | Opcional. Una o más instrucciones que se ejecutan para cada elemento de un <i>grupo</i> . |

Comentarios

La entrada al bloque **For Each** se produce si hay al menos un elemento en *grupo*. Una vez que se ha entrado en el bucle, todas las instrucciones en el bucle se ejecutan para el primer elemento en *grupo*. Después, mientras haya más elementos en *grupo*, las instrucciones en el bucle continúan ejecutándose para cada elemento. Cuando no hay más elementos en el *grupo*, se sale del bucle y la ejecución continúa con la instrucción que sigue a la instrucción **Next**.

Se pueden colocar en el bucle cualquier número de instrucciones **Exit For**. La instrucción **Exit For** se utiliza a menudo en la evaluación de alguna condición (por ejemplo, **If...Then**) y transfiere el control a la instrucción que sigue inmediatamente a la instrucción **Next**.

Puede anidar bucles **For Each...Next**, colocando un bucle **For Each...Next** dentro de otro. Sin embargo, cada *elemento* del bucle debe ser único.

Nota Si omite *elemento* en una instrucción **Next**, la ejecución continúa como si se hubiera incluido. Si se encuentra una instrucción **Next** antes de su instrucción **For** correspondiente, se producirá un error.

No se puede utilizar la instrucción **For Each...Next** con una matriz de tipos definidos por el usuario porque un tipo **Variant** no puede contener un tipo definido por el usuario.

Operador typeof

[Vea también](#)

Descripción

Se utiliza para determinar el tipo de una expresión.

Sintaxis

typeof [(] *expresión* [)] ;

El argumento *expresión* es cualquier [expresión](#) para la que se busca la información del tipo.

Comentarios

El operador **typeof** devuelve información del tipo como una cadena.
El operador **typeof** devuelve uno de entre seis valores posibles:
"number," "string," "boolean," "object," "function," y "undefined."

Los paréntesis son opcionales en la sintaxis de **typeof**.

Precedencia de operadores

EN JScript los operadores se evalúan en un orden particular. Este orden se conoce como la precedencia de operadores. La siguiente tabla muestra una lista de los operadores en orden de precedencia de mayor a menor. Los operadores con igual precedencia se evalúan de izquierda a derecha en la expresión.

| Operador >> | Descripción |
|----------------------------------|---|
| . [] () | Acceso a campos, indexación de matrices y llamadas a funciones |
| ++ -- ~ ! delete new typeof void | Operadores unarios, tipos de datos devueltos, creación de objetos, valores no definidos |
| * / % | Multiplicación, división, división módulo |
| + - + | Adición, sustracción, concatenación de cadenas |
| << >> >>> | Desplazamiento de bits |
| < <= > >= | Menor que, menor o igual que, mayor que, mayor o igual que |
| == != === !== | Igualdad, desigualdad, identidad, desidentidad |
| & | AND de bits |
| ^ | XOR de bits |
| | OR de bits |
| && | AND lógico |
| | OR lógico |
| ?: | Condicional |
| = OP= | Asignación, asignación con operación |
| , | Evaluación múltiple |

Los paréntesis se usan para modificar el orden de evaluación. La expresión encerrada entre paréntesis se evalúa por completo antes de usar su valor en el resto de la instrucción.

Un operador con mayor precedencia se evalúa antes que uno con menor precedencia. Por ejemplo:

```
z = 78 * (96 + 3 + 45)
```

En esta expresión hay cinco operadores: =, *, (), + y +. Según las normas de precedencia, se evalúan en el siguiente orden: (), *, +, +, =.

1. Lo primero es la evaluación de la expresión que está encerrada entre paréntesis: Hay dos operadores de adición y tienen la misma precedencia: Se suma 96 y 3 y al total resultante se suma 45, dando como resultado un valor igual a 144.
 2. Lo siguiente es la multiplicación: Se multiplican 78 y 144, dando como resultado un valor igual a 10998.
 3. Por último se realiza la asignación: Se asigna 11232 a z.
-

.

