aprenderaprogramar.com

Arrays unidimensionales. Dim.Option Base. Erase. Ejemplos en Visual Basic (CU00311A)

Sección: Cursos

Categoría: Curso Visual Basic Nivel I

Fecha revisión: 2029

Autor: Mario R. Rancel

Resumen: Entrega nº10 del Curso Visual Basic Nivel I

29

VARIABLES CON ÍNDICE O LOCALIZADOR. ARRAYS.

El concepto de array con Visual Basic coincide con el que se expone en el curso "Bases de la programación nivel I" de aprenderaprogramar.com en pseudocódigo. Veremos ahora cómo declarar arrays estáticos de una dimensión. La gestión de arrays multidimensionales y dinámicos la veremos más adelante.

ARRAYS UNIDIMENSIONALES

La sintaxis a emplear será:

```
Dim [Nombre del array]([dimensión]) As [Tipo de variable]
```

Si no se declara el tipo de variable el array será tipo Variant ó tipo Object (según la versión de Visual Basic que estemos empleando) por defecto. Conviene tener cuidado con esto y seguir las mismas pautas que hemos indicado para las variables. Ejemplos de declaración de arrays serían:

- Dim Vez(10) As Integer
- Dim Amigo(1000) As String
- Dim Value(25) As Single
- Dim NúmerodeCoches(24) As Integer
- Dim Jugador(8) As Long
- Dim TCP(3) As Boolean

Cuando creamos un array del tipo A(n) podríamos pensar que estamos creando n variables que son A(1), A(2), A(3), ..., A(n). Pero en realidad estamos creando n+1 variables porque Visual Basic crea también A(0). Esto conviene tenerlo en cuenta porque puede dar lugar a alguna confusión. Disponer de un valor con índice cero puede ser de utilidad en situaciones como considerar cada variable asociada a una hora del día, empezando a contar desde la hora cero hasta la 23 (total de 24 horas), cosa que es habitual en algunos países. En lugar de 1, 2, 3, ..., 24 estaríamos usando 0, 1, 2, ..., 23.

En la mayoría de lenguajes de programación se usa el índice cero como primer índice de un array. No obstante, en las versiones menos recientes de Visual Basic se permitía establecer como primer índice de un array el 1 mediante la instrucción ***Option Base***. A través de ella se puede establecer como primer índice del array el uno en vez de el cero en aquellas versiones que lo admiten. Para ello se escribe en la cabecera del programa:

```
Option Base 1
```

Option Base afecta a todos los arrays que se declaren. También podemos indicar que el primer índice de los arrays es cero escribiendo Option Base 0 en las versiones de Visual Basic que admiten esta instrucción. Esto equivale a no poner nada (por defecto el primer índice será cero). Vamos a trabajarlo sobre el ordenador. En la ventana de código escribe lo siguiente:

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Numerodecoches(24) As Integer

Private Sub Form_Load()
Numerodecoches(0) = 14
MsgBox ("El número de coches en la hora
cero fue " & Numerodecoches(0))
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
   Private Sub Form1_Load (ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      Dim Numerodecoches(24) As Integer
      Numerodecoches(0) = 14
      MsgBox("El número de coches en la hora cero fue " &
Numerodecoches(0))
   End Sub
End Class
```

Al ejecutar el código (Ctrl+F5), te aparecerá el mensaje "El número de coches en la hora cero fue 14". Modifica el programa indicando Numerodecoches(29) = 14 en lugar de Numerodecoches(0) = 14. Prueba a ejecutarlo y te aparecerá un mensaje del tipo: "Error en tiempo de ejecución. El subíndice está fuera del intervalo". Efectivamente, se debe a haber puesto un índice fuera del rango y ser los valores que puedes usar en el código como localizador de una variable del array los comprendidos entre 0 y 24. Al estar usando la variable Numerodecoches(29), Visual Basic detecta la incoherencia y genera un error.

Otra opción que nos dan algunas versiones menos recientes de Visual Basic es tener un array de variables cuyos localizadores no comienzan en cero ni en uno, sino en un número específico. La sintaxis para ello en las versiones que lo permiten es:

```
Dim [Nombre de variable]([Valor inicial] To [Valor final]) As [Tipo de variable]
```

Supongamos que queremos definir una serie de variables asociadas a las horas comprendidas entre las 10 y las 18 horas. En las versiones que lo permiten podríamos crear un array del tipo:

```
Dim Magnitud (10 To 18) As Integer
```

En este caso si la versión admite esta sintaxis los valores válidos como índices del array serían 10, 11, 12, 13, 14, 15, 16, 17 y 18. Cualquier número que no sea de esta serie usado como localizador del array daría lugar a un error por subíndice fuera del intervalo. En las versiones más recientes de Visual Basic no es posible definir índices en un rango, de modo que hemos de declarar el array como un array normal cuyo primer índice es cero, pudiendo dejar vacíos o sin usar aquellos índices que no nos resulten útiles.

El contenido de una matriz estática puede ser borrado utilizando la instrucción **Erase**. La sintaxis será la siguiente:

```
Erase [Nombre de array1], [Nombre de array2], ..., [Nombre de array n]
```

Ejemplos de uso de Erase pueden ser:

- Erase Númerodecoches
- Erase Númerodecoches, Usuarios, Calles

Al invocar a Erase sobre una variable de tipo array el resultado depende de la versión de Visual Basic que estemos utilizando:

a) En las versiones menos recientes el contenido de las variables que constituyen el array estático se transforma en cero si son de tipo numérico o cadena vacía si son de tipo texto (String), pero el array sigue existiendo y teniendo un contenido válido.
b) En las versiones más recientes el contenido de las variables que constituyen el array estático queda eliminado y no es posible invocar un elemento del array. La variable del array pasa a tener valor Nothing y para volver a usarla tendríamos que establecer una redefinición del array con la instrucción ReDim, que estudiaremos más adelante.

El comportamiento de Erase lo volveremos a estudiar pues tiene distintas aplicaciones. Ejecuta el siguiente programa para comprobar el funcionamiento de Erase.

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com

Option Explicit
Dim Numerodecoches(24) As Integer

Private Sub Form_Load()
Numerodecoches(0) = 14
Erase Numerodecoches
MsgBox ("El número de coches en la hora
cero fue " & Numerodecoches(0))
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
   Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      Dim Numerodecoches(24) As Integer
      Numerodecoches(0) = 14
      Erase Numerodecoches
      MsgBox("El número de coches en la hora cero fue " &
Numerodecoches(0))
   End Sub
End Class
```

El resultado será:

a) Para las versiones menos recientes será: El número de coches en la hora cero fue 0. A pesar de haber definido Numerodecoches para la hora cero como 14, este valor no se llega a mostrar porque se ejecuta un Erase, que da lugar a que todos los valores del array se establezcan a cero por ser el array de tipo Integer.

b) Para las versiones más recientes: Error. No se controló NullReferenceException. Referencia a objeto no establecida como instancia de un objeto. Este error se genera porque se considera que el array no tiene índices establecidos y tendríamos que volver a establecerlos usando la instrucción ReDim que estudiaremos más adelante.

Si eliminas el Erase, te deberá aparecer en pantalla "El número de coches en la hora cero fue 14".

EJERCICIO:

Crea el código de dos programas que cumplan las siguientes premisas:

a) ***Programa 1***.

Declara un array tipo Integer denominado Numerodecoches cuyo número de elementos sea 24. Declara una variable tipo Integer que se llame R. En el procedimiento de carga del formulario establece el valor de R en 2 y el valor de Numerodecoches para un localizador de valor R en 233. Procede a mostrar en pantalla un mensaje que indique cuál es la hora R y el número de coches para la hora R. Finalmente, modifica únicamente la asignación de valor a R de modo que en vez de 2 sea 21 y ejecuta de nuevo el programa.

b) ***Programa 2.***

Sobre el programa anterior realiza los siguientes cambios. Mantén el número de elementos de Numerodecoches en 24. Declara dos variables A y B de tipo Integer. Establece A con valor 8, B con valor 4 y R con valor A dividido entre B. Ejecuta el programa.

SOLUCIÓN:

El programa 1 será el siguiente. Si lo ejecutamos obtendremos "El número de coches en la hora 2 fue 233". Si cambiamos R = 2 por R=21 obtendremos "El número de coches en la hora 21 fue 233".

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim Numerodecoches(23) As Integer
Dim R As Integer

Private Sub Form_Load()
R = 2
Numerodecoches(R) = 233
MsgBox ("El número de coches en la hora "
& R & " fue " & Numerodecoches(R))
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
    Dim Numerodecoches(23) As Integer
    Dim R As Integer

    Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        R = 2
        Numerodecoches(R) = 233
        MsgBox("El número de coches en la hora " & R & " fue
" & Numerodecoches(R))
    End Sub
End Class
```

Nota: usamos Numerodecoches(23) en lugar de Numerodecoches(24) porque el número de elementos es uno más que el declarado debido a que se cuenta el cero. Así, si declaráramos Numerodecoches(24) tendríamos 25 elementos (correspondientes a los elementos del 1 al 24 más el elemento cero).

El programa 2 será el siguiente. Recuerda también que usar una declaración tipo Dim A, B As Integer no sería válida porque supondría que A queda declarada como tipo Variant o tipo Object. Para que no haya duda usaremos Dim A%, B% ó Dim A As Integer, B As Integer

Para las versiones menos recientes:

```
Rem Curso Visual Basic aprenderaprogramar.com

Option Explicit
Dim Numerodecoches(23) As Integer
Dim A%, B As Integer
Dim R As Integer

Private Sub Form_Load()
A = 8
B = 4
R = A / B
Numerodecoches(R) = 233
MsgBox ("El número de coches en la hora "
& R & " fue " & Numerodecoches(R))
End Sub
```

Para las versiones más recientes:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On
Public Class Form1
   Dim Numerodecoches(23) As Integer
   Dim A As Integer, B As Integer
   Dim R As Integer
   Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
      A = 8
      B = 4
      R = A / B
      Numerodecoches(R) = 233
      MsgBox("El número de coches en la hora " & R & " fue
" & Numerodecoches(R))
   End Sub
End Class
```

Próxima entrega: CU00312A

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61