aprenderaprogramar.com

Arrays dinámicos y arrays estáticos con Visual Basic y .NET. Redim, Redim Preserve, Erase. Ejemplos (CU00342A)

Sección: Cursos

Categoría: Curso Visual Basic Nivel I

Fecha revisión: 2029

Autor: Mario R. Rancel

Resumen: Entrega nº41 del Curso Visual Basic Nivel I

29

ARRAYS DINÁMICOS Y ARRAYS ESTÁTICOS. INSTRUCCIÓN REDIMENSIONAR (REDIM Y REDIM PRESERVE)

En el apartado correspondiente a declaración de variables ya hemos tratado la forma de declarar arrays estáticos unidimensionales y multidimensionales a través de la instrucción Dim. Usaremos esta misma instrucción para declarar arrays dinámicos. Recordemos que habíamos definido array dinámico como aquel que es declarado con un número de elementos componentes indefinido. La sintaxis a emplear será:

Dim [Nombre del array]() As [Tipo de datos]

Redim [Nombre del array]([dimensiones])

Es decir, en la cabecera de código del formulario o de un procedimiento haremos una declaración sin especificar dimensiones que luego concretaremos a través de la instrucción Redim. Por ejemplo:

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim A() As Integer

Private Sub Form_Load()
ReDim A(2)
A(1) = 3
Label1 = "El valor de A(1) es " & A(1)
End Sub
```

Código versiones más recientes VB:

```
REM Curso Visual Basic aprenderaprogramar.com
Option Explicit On

Public Class Form1
    Dim A() As Integer
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        ReDim A(2)
        A(1) = 3
        Label1.Text = "El valor de A(1) es " & A(1)
End Sub
End Class
```

Dim A() As Integer supone la declaración del array dinámico (sin tamaño definido inicialmente).

ReDim A(2) establece que el array dinámico definido anteriormente tiene índice máximo 2.

En el caso de querer declarar una matriz (array o arreglo de varias dimensiones) como M (5, 7) que podría representar una tabla con datos como por ejemplo M (0,0) = 33, M (0,1) = 18, M (0,2) = 41 … etc. debemos hacerlo de la siguiente manera:

a) En las versiones menos recientes de Visual Basic declararemos inicialmente Dim M() As Integer y posteriormente ReDim M (5, 7). Una vez fijado que M tiene dos dimensiones ya no es posible cambiar el número de dimensiones por otro número.

b) En las versiones más recientes de Visual Basic declararemos inicialmente Dim M(,) As Integer, donde la coma indica que van a existir dos dimensiones que se declararán posteriormente. Si fuéramos a declarar posteriormente tres dimensiones escribiríamos Dim M (, ,) As Integer, para 4 dimensiones Dim M (, , ,) y así sucesivamente. No es posible cambiar el número de dimensiones de un array dinámico, que será el declarado inicialmente.

Un array estático no puede ser redimensionado. En cambio, uno dinámico puede sufrir múltiples redimensionamientos. No se puede declarar el tipo de dato cada vez que redimensionemos. Hay que hacerlo al declarar la matriz dinámica, una sola vez.

Por defecto, el uso de Redim implicará perder todos los datos anteriormente contenidos en la matriz (array, arreglo) que se redimensiona. Para conservar los datos anteriores podemos hacerlo de dos maneras. Por un lado, usando la cláusula Preserve según esta sintaxis:

```
Redim Preserve [Nombre del array]([dimensiones])
```

La limitación de usar Preserve viene dada porque únicamente se nos permite cambiar la última dimensión del array (la situada más a la derecha). Tampoco se permite cambiar el número de dimensiones. Si estamos trabajando con un array unidimensional no habrá problemas, pero con otro tipo de arrays puede haberlos.

Ejemplo: si declaramos un array multidimensional denominado B como Dim B(, ,) As Integer (en versiones menos recientes simplemente Dim B () As Integer), luego redimensionamos ReDim B(4, 2, 6) y posteriormente intentamos hacer ReDim B (2,2,2) obtendremos un mensaje de error ya que sólo puede cambiarse la dimensión más a la derecha. Es decir, ReDim (4, 2, x) donde x es la dimensión cuyo tamaño cambiamos es válido. Pero cambiar las dimensiones anteriores no es válido.

Cuando se utiliza Preserve aumentando el número de elementos del array, los nuevos elementos tienen valor cero o nulo y los que existían anteriormente conservan su valor. Cuando se trata de una disminución de los elementos de un array, aquellos contenidos en índices que desaparecen dejan de existir a todos los efectos, mientras que los existentes en dimensiones que se conservan mantienen su valor. En caso de arrays de varias dimensiones, con Redim Preserve sólo podemos cambiar la última dimensión, al igual que hemos visto antes. Para un array dinámico que se ha redimensionado como A(5, 5, 5):

a) Es válido: Redim Preserve A(5, 5, 8)
   Modifica la última dimensión.

b) No es válido: Redim Preserve A(5, 8, 5)
   Modifica una dimensión que no es la última.

Para conservar el contenido de arrays cuando no es posible a través de Preserve habremos de hacerlo declarando otro array con las dimensiones deseadas y transfiriendo elemento a elemento los valores desde un array a otro.

Una cuestión a tener en cuenta es que la instrucción Erase, cuyo funcionamiento habíamos visto que con matrices estáticas daba lugar a que todos los elementos de la matriz se convirtieran en ceros o cadenas vacías, al ser aplicada a un array dinámico da lugar a que este desaparezca totalmente. Por tanto, no podremos invocar la matriz después de usar Erase a no ser que la declaremos de nuevo usando ReDim nuevamente. Para borrar el contenido de una matriz dinámica bastará con usar la instrucción Redim sin variar las dimensiones, o bien establecer el valor de cada uno de los elementos a cero o cadena vacía a través del código.

En las versiones menos recientes de Visual Basic era posible variar el número de localizadores de un array con ReDim (sin Preserve), tanto para aumentarlos como para disminuirlos, usando código de este tipo (siempre sin Preserve):

```
'Curso Visual Basic aprenderaprogramar.com
Option Explicit
Dim A() As Integer

Private Sub Form_Load()
ReDim A(2, 3, 4, 5)
A(1, 1, 1, 1) = 3
Label1 = "El valor de A(1,1,1,1) es " & A(1, 1, 1, 1)
ReDim A(2)  '[Reducción del número de localizadores]
End Sub
```

Donde observamos que el array *A*, declarado inicialmente como de cuatro dimensiones, es redimensionado para dejarlo como unidimensional. En las versiones más recientes de Visual Basic esto no es posible, porque la declaración del array A tiene que establecer un número de dimensiones fijas para el mismo. En este ejemplo tendríamos que declarar Dim A(, , ,) As Integer y de este modo el número de dimensiones de A será siempre 4.

EJERCICIO

Crear el código correspondiente al siguiente pseudocódigo de un programa MATRIZ01 como ejemplo del redimensionamiento de arrays. Se pretende declarar un array de dimensiones (3,3) donde se usan los índices 1 a 3 y cada elemento tiene como contenido el valor de la suma de sus índices. Por ejemplo Matriz (1,1) = 2, Matriz (1,2) = 3, Matriz (1,3) = 4, Matriz (2,1) = 3, Matriz (2,2) = 4, etc. Una vez creado, se quiere reducir el tamaño de la matriz a (2,2) conservando en ella los valores que inicialmente existían en los índices que no se han eliminado y posteriormente volver a redimensionar a dimensiones (3,3) y mostrar el contenido de todos los elementos de la matriz.

```
[Curso VB aprenderaprogramar.com]

    Variables
        Enteras: Matriz(), i, j

    1. Inicio
        2. Redimensionar Matriz(3,3)
        3. Desde i = 1 hasta 3
            3.1 Desde j = 1 hasta 3
                3.1.1 Matriz(i,j) = i + j
            Siguiente
        Siguiente
        4. Redimensionar Matriz(2,2) conservando elementos
        5. Mostrar "Se han eliminado 9 – 4 = 5 elementos"
        6. Redimensionar Matriz(3,3) conservando elementos
        6. Mostrar todos los elementos de la matriz
    8. Fin
```

SOLUCIÓN

Código versiones menos recientes VB:

```
'Curso VB aprenderaprogramar.com
Option Explicit
Dim Matriz() As Integer
Dim Transitoria() As Integer
Dim i%, j As Integer

Private Sub Form_Load()
ReDim Matriz(3, 3)
Label1 = ""
For i = 1 To 3
   For j = 1 To 3
      Matriz(i, j) = i + j
   Next j
Next i
ReDim Transitoria(2, 2)
For i = 1 To 2
   For j = 1 To 2
      Transitoria(i, j) = Matriz(i, j)
   Next j
Next i
ReDim Matriz(2, 2)
Label1 = Label1 & "Se han eliminado 9 - 4 = 5
elementos" & vbCrLf & vbCrLf
ReDim Matriz(3, 3)
For i = 1 To 2
   For j = 1 To 2
      Matriz(i, j) = Transitoria(i, j)
   Next j
Next i
Label1 = Label1 & "La variable Matriz(3,2) vale "
& Matriz(3, 2) & vbCrLf & vbCrLf
For i = 1 To 3
   For j = 1 To 3
      Label1 = Label1 & "Matriz (" & i & "," & j &
") =" & Matriz(i, j) & "    "
   Next j
Next i
End Sub
```

Código versiones más recientes VB

```
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 Matriz(,) As Integer
        Dim Transitoria(,) As Integer
        Dim i, j As Integer

        ReDim Matriz(3, 3)
        Label1.Text = ""
        For i = 1 To 3
            For j = 1 To 3
                Matriz(i, j) = i + j
            Next j
        Next i
        ReDim Transitoria(2, 2)
        For i = 1 To 2
            For j = 1 To 2
                Transitoria(i, j) = Matriz(i, j)
            Next j
        Next i
        ReDim Matriz(2, 2)
        Label1.Text = Label1.Text & "Se han
eliminado 9 - 4 = 5 elementos" & vbCrLf & vbCrLf
        ReDim Matriz(3, 3)
        For i = 1 To 2
            For j = 1 To 2
                Matriz(i, j) = Transitoria(i, j)
            Next j
        Next i
        Label1.Text = Label1.Text & "La variable
Matriz(3,2) vale " & Matriz(3, 2) & vbCrLf &
vbCrLf
        For i = 1 To 3
            For j = 1 To 3
                Label1.Text = Label1.Text &
"Matriz (" & i & "," & j & ") =" & Matriz(i, j) &
"    "
            Next j
        Next i
    End Sub
End Class
```

Comentarios: Dado que pretendemos modificar varias dimensiones de la matriz, no podemos utilizar Preserve y hacerlo automáticamente. Por ello creamos una matriz transitoria que utilizamos para evitar que se pierdan los valores que nos interesan.

El resultado es el que esperábamos con el pseudocódigo pero el tener que usar matrices transitorias complica un poco la programación.

Tras ejecutar el programa obtendremos este resultado:

Se han eliminado 9 - 4 = 5 elementos

La variable Matriz (3,2) vale 0

Matriz (1,1) =2 Matriz (1,2) =3 Matriz (1,3) =0 Matriz (2,1) =3 Matriz (2,2) =4 Matriz (2,3) =0
Matriz (3,1) =0 Matriz (3,2) =0 Matriz (3,3) =0

Podemos comprobar que todos los elementos de índice 3 tienen valor cero.

Próxima entrega: CU00343A

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