

Objetivos:

- ❖ Practicar con algoritmos típicos con **vectores** (arrays de una dimensión)

Operaciones con vectores

Interfaz

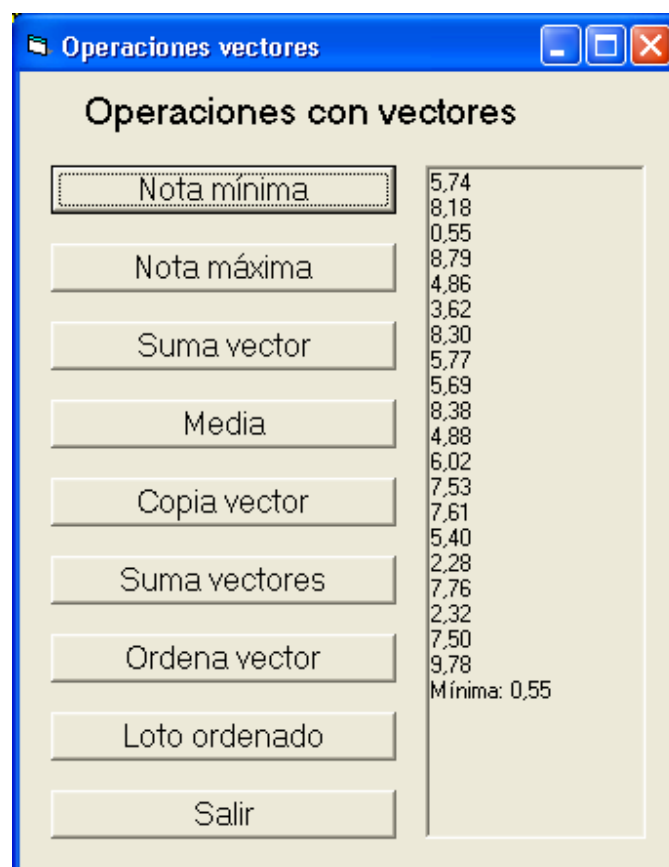


Figura 12.1. Interfaz del programa de operaciones con vectores.

Funcionamiento

En el laboratorio anterior se propusieron ejercicios para obtener listas de notas aleatorias. En este laboratorio vamos a retomar parte del código utilizado para no andar introduciendo los datos del teclado. En general utilizaremos vectores de notas (números reales del 0 al 10) aunque en un último ejercicio retomaremos la generación de una lista de seis números enteros distintos del 1 al 49 para rellenar un boleto de la “loto” de manera ordenada.

El funcionamiento básico de cada botón consistirá en generar una o dos listas de números aleatorios del 0 al 10 y obtener el mínimo, máximo, suma y media de los elementos, copia de un vector a otro, suma de dos vectores, ordenación de un vector y la mencionada lista de números de la lotería. Veámoslos por partes.

Botón “Nota mínima”

Declaramos un vector `v()` de 20 elementos y lo rellenamos de números aleatorios del 0 al 10 mediante el procedimiento `IniVectorNotas`, que es el mismo que vimos en el laboratorio 11. Llamamos a la función `iMin`, que nos devuelve el índice del menor de los `n` elementos de un vector, mostramos el vector mediante `MuestraVectorDbl`, que es similar al procedimiento `sacaVectorDbl` visto en el laboratorio anterior pero éste lo muestra en un cuadro de dibujo en vez de utilizar `MsgBox`; finalmente mostramos el elemento mínimo

```
Sub cmd1_Click()  
    Dim v(1 To 20) As Double  
    Dim i As Integer  
    Dim min As Double  
    pctRes.Cls ' Limpiar el cuadro de dibujo  
    Randomize ' Inicializar la semilla con el reloj  
    Call IniVectorNotas(v)  
    i = iMin(20, v)  
    min = v(i)  
    Call MuestraVectorDbl(v)  
    pctRes.Print "Mínima: " & Format(min, "0.00")  
End Sub
```

Recordemos el procedimiento para inicializar el vector de notas:

```
Sub IniVectorNotas(ByRef v() As Double)  
    Dim i As Integer  
    For i = LBound(v) To UBound(v) Step 1  
        v(i) = Rnd * 10  
    Next i  
End Sub
```

El diagrama de flujo de la función `iMin` que obtiene el índice del menor elemento del vector se muestra en la Figura 12.2. A partir de este índice ya podremos obtener el menor de los números, `min`. Nótese que un número puede estar repetido, aunque no sea probable ya que el generador de números aleatorios nos da series de números distintos. Por mucho que demos formato con dos decimales los números estarán guardados con mayor precisión.

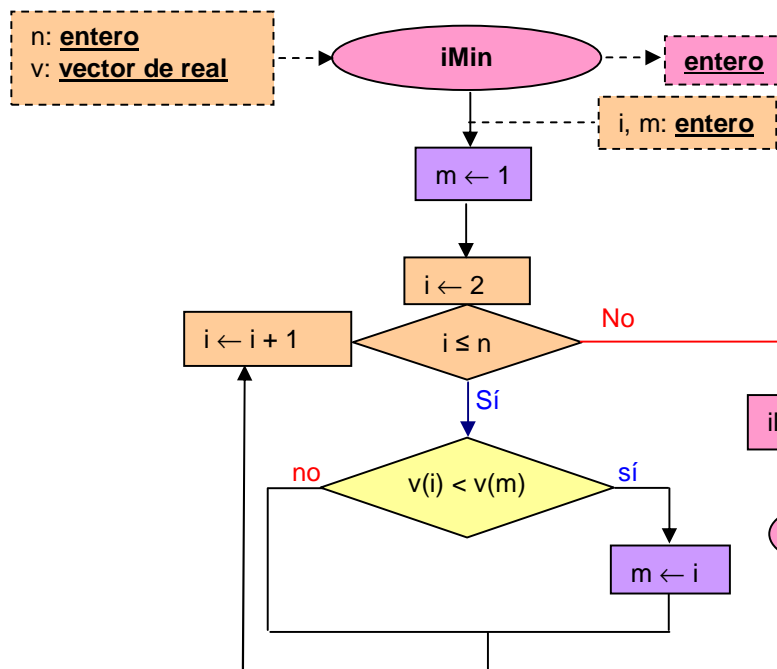


Figura 12.2. Diagrama de flujo de la función **iMin**.

El procedimiento para mostrar en el cuadro de dibujo el vector formateado puede ser:

```

Sub MuestraVectorDbl(ByRef v() As Double)
    Dim i As Integer
    For i = LBound(v) To UBound(v) Step 1
        pctRes.Print Format(v(i), "0.00")
    Next i
End Sub
  
```

Botón “Nota máxima”

Es similar al botón “Nota mínima”.

Botón “Suma vector”

Obtiene la suma de los elementos del vector:

```

Sub cmd3_Click()
    Dim v(1 To 20) As Double
    Dim sum As Double
    pctRes.Cls
    Randomize
    Call IniVectorNotas(v)
    sum = SumaVector(20, v)
    Call MuestraVectorDbl(v)
    pctRes.Print "Suma: " & Format(sum, "0.00")
End Sub
  
```

Botón “Media”

Obtiene la media de los elementos del vector. Puede obtenerse directamente dividiendo el valor obtenido por la función anterior **SumaVector** dividido por el número de elementos.

```

Sub cmd4_Click()
    Dim v(1 To 20) As Double
    Dim med As Double
    pctRes.Cls
    Randomize
    Call IniVectorNotas(v)
    med = MediaVector(10, v)
    Call MuestraVectorDbl(v)
    pctRes.Print "Media: " & Format(med, "0.00")
End Sub

```

Botón “Copia vector”

Obtiene la copia de un vector. Definiremos un vector **v1** que inicializaremos mediante el procedimiento **IniVectorNotas** y lo copiaremos a un vector destino **v2**, mostrando el contenido de ambos en el cuadro de dibujo.

```

Sub cmd5_Click()
    Dim v1(1 To 10) As Double, v2(1 To 10) As Double
    pctRes.Cls
    Randomize
    Call IniVectorNotas(v1)
    Call CopiaVector(10, v1, v2)
    pctRes.Print "Vector origen:"
    Call MuestraVectorDbl(v1)
    pctRes.Print "Vector destino:"
    Call MuestraVectorDbl(v2)
End Sub

```

El procedimiento de copia tendrá la cabecera que se muestra en la Figura 12.3.



Figura 12.3. Cabecera del procedimiento de copia de un vector a otro.

Botón “Suma vectores”

Obtiene la suma de dos vectores. Definiremos dos vector **v1** y **v2** que inicializaremos mediante el procedimiento **IniVectorNotas** y los sumaremos a un vector resultado **v**, mostrando el contenido de los tres en el cuadro de dibujo.

```

Sub cmd6_Click()
    Dim v1(1 To 7) As Double, v2(1 To 7) As Double
    Dim v(1 To 7) As Double
    pctRes.Cls
    Randomize
    Call IniVectorNotas(v1)
    Call IniVectorNotas(v2)
    Call SumaVectores(7, v1, v2, v)
    pctRes.Print "Vector 1:"

```

```

Call MuestraVectorDbl(v1)
pctRes.Print "Vector 2:"
Call MuestraVectorDbl(v2)
pctRes.Print "Vector suma:"
Call MuestraVectorDbl(v)
End Sub

```

El procedimiento de suma tendrá la cabecera que se muestra en la Figura 12.4.



Figura 12.4. Cabecera del procedimiento de suma de vectores.

Botón “Ordena vector”

Genera un vector, lo copia y ordena la copia, mostrando ambos vectores.

```

Sub cmd7_Click()
  Dim v1(1 To 10) As Double, v2(1 To 10) As Double
  pctRes.Cls
  Randomize
  Call IniVectorNotas(v1)
  Call CopiaVector(10, v1, v2)
  Call OrdenaVector(10, v2)
  pctRes.Print "Vector origen:"
  Call MuestraVectorDbl(v1)
  pctRes.Print "Vector ordenado:"
  Call MuestraVectorDbl(v2)
End Sub

```

El procedimiento de ordenación tendrá la cabecera que se muestra en la Figura 12.5.



Figura 12.5. Cabecera del procedimiento de ordenación de un vector.

Obsérvese que el vector v es un parámetro de entrada y salida.

Para ordenar el vector utilizaremos un algoritmo sencillo, cuyo diagrama de flujo se muestra en la figura 12.6. Descrito en palabras, buscaremos el índice del elemento menor y lo intercambiaremos con el primero, luego el del segundo menor y lo intercambiaremos con el segundo, y así sucesivamente.

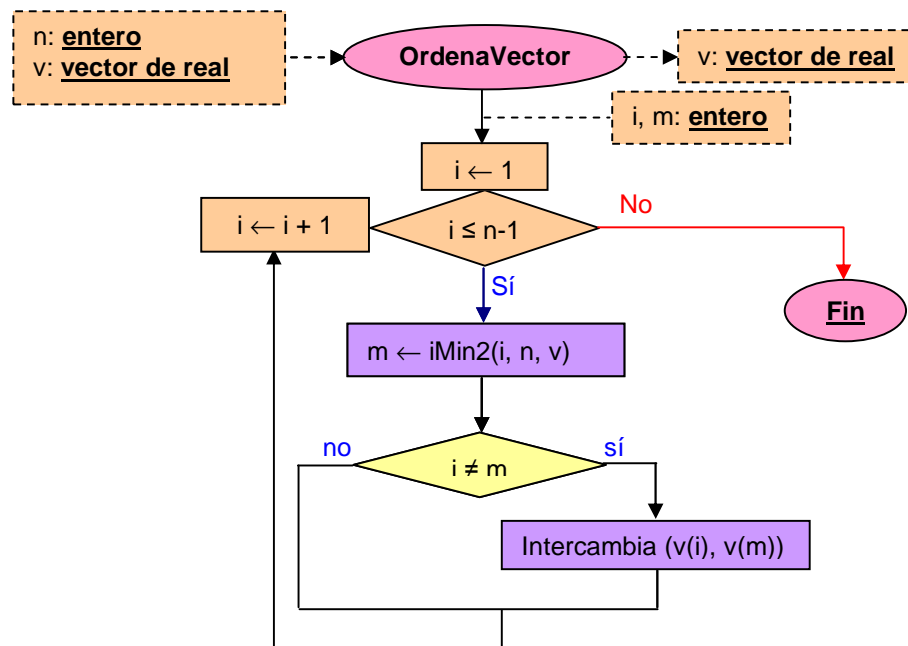


Figura 12.6. Diagrama de flujo del procedimiento de ordenación

Haremos uso de la función `iMin2` que nos devuelve el índice del menor elemento de un vector de entre dos índices, `i` y `n`. Esta función es similar a `iMin`, utilizada en el “Botón mínimo” cuyo diagrama de flujo se muestra en la Figura 12.2, con la diferencia que en vez de comenzar por 1 comienza por otro índice que recibe como parámetro.

También hacemos uso del procedimiento `Intercambia` que, tal y como expresa su nombre, intercambia el valor de sus argumentos. La figura 12.7 muestra su cabecera, con `x` e `y` como parámetros de entrada y salida.

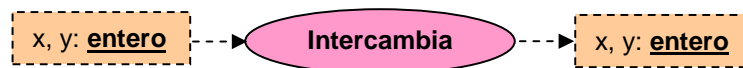


Figura 12.7. Diagrama de flujo de la cabecera del procedimiento de `Intercambia`.

Botón “Loto ordenado”

Genera un conjunto de seis números aleatorios distintos ordenados entre 1 y 49. Para ello utiliza el algoritmo de la *inserción ordenada*, es decir, para cada número nuevo generado calcula la posición en la que debe ir ordenado en el vector. Si ya existe no se insertará y si no existe se desplazarán a la derecha todos los números para hacer espacio al nuevo número en su posición.

A continuación se propone el código asociado al botón en cuestión. La función `posOrd` inserta el nuevo número `num` al final de la lista (llamado *centinela*) y devuelve la posición del primer elemento que sea mayor o igual a `num`. De esta manera sabemos que vamos a encontrar siempre uno mayor o igual. Si existe anteriormente nos dará la posición en la que se encuentra (y no

haremos nada). Si había que añadirlo al final sólo tendremos que incrementar en uno el número de elementos puesto que ya está introducido.

```

Sub cmd8_Click()
    Dim n As Integer, i As Integer
    Dim p As Integer
    Dim num As Integer
    Dim v(1 To 6) As Integer

    pctRes.Cls
    n = 0
    Do
        num = Aleatorio(1, 49)
        p = PosOrd(num, n, v)
        If p <= n Then
            If v(p) <> num Then ' si es igual num ya estaba en el vector
                For i = n To p Step -1 ' desplazar a la derecha a partir de pos
                    v(i + 1) = v(i)
                Next i
                n = n + 1
                v(p) = num
            End If
        Else
            n = n + 1
            ' PosOrd ya ha introducido num en esta posición
        End If
    Loop Until n = 6
    Call MuestraVectorInt(v)
End Sub

```

La Figura 12.8 representa el diagrama de flujo de la función `PosOrd` con el comportamiento anteriormente descrito.

Nótese que se utiliza un bucle `While` ya que no sabemos de antemano cuántas comparaciones hemos de realizar.

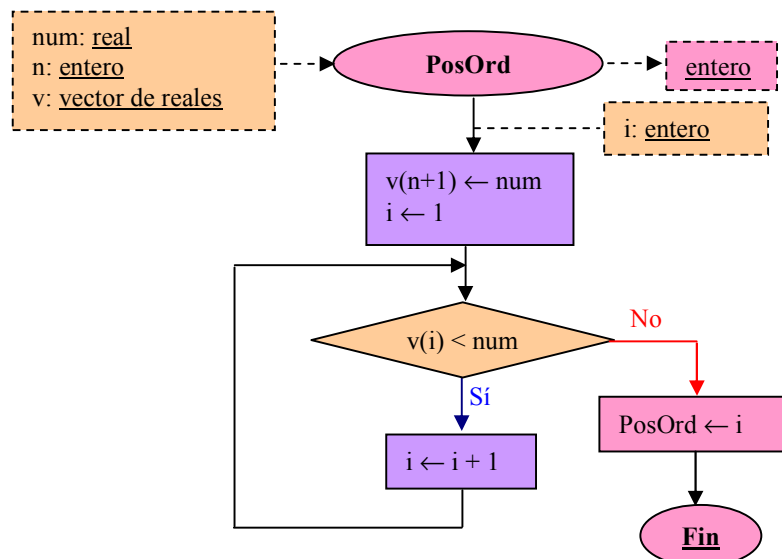


Figura 12.8. Diagrama de flujo de la función `PosOrd`.