



#### Objetivos:

- ❖ Profundizar en el diseño de **diagramas de flujo** con **estructuras repetitivas**.
- ❖ Profundizar en la codificación de programas VB con **sentencias repetitivas**.

## Programa de demostración del uso de sentencias repetitivas

### Interfaz

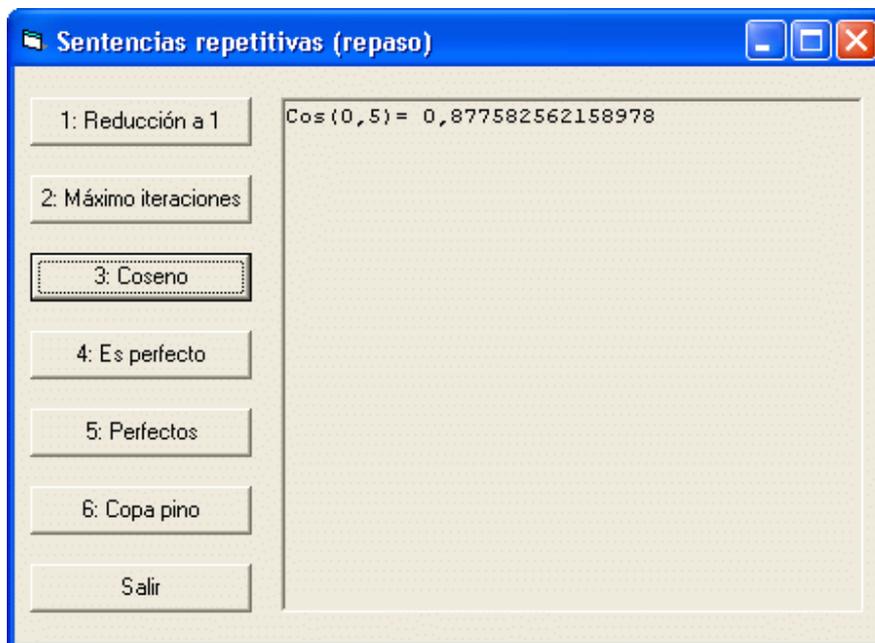


Figura 7.1. Objetos presentes en la interfaz: botones y caja de dibujo

### Funcionamiento general

1. Cada **ejercicio** tendrá un botón de ejecución propio (**cmdEj1**, **cmdEj2**, ..., **cmdEj6**).
2. Para los resultados dispondremos de una **caja de dibujo** (*Picture Box*), **pctRes**. Para poder ver mejor los resultados haremos que la fuente por defecto de este objeto sea "Courier New" de tamaño 8.
3. Al hacer **click** sobre cada botón, borraremos inicialmente el contenido de la **caja de dibujo pctRes**. Utilizaremos para ello el método **Cls** (**pctRes.Cls**).
4. Al hacer **click** en el botón **Salir**, el **programa finalizará**.
5. Se proporciona un modelo de programa **ejecutable** para clarificar los enunciados.

## Ejercicio 7.1: Reducción a 1 (parcialmente resuelto)

### Enunciado

Partiendo de un número **natural positivo**  $n$  y aplicando sucesivamente la **división por dos** cuando es **par** ( $n \text{ Div } 2$ ) y la **multiplicación por tres más uno** ( $n \cdot 3 + 1$ ) cuando es **impar** se obtiene el número 1.

Diseña el **diagrama de flujo** y codifica el **programa** que lea  $n$ , vaya operando de esta manera y nos muestre en la caja de dibujo las operaciones que va a realizar, mostrándonos al final el número de operaciones que ha necesitado mediante un **MsgBox**. Nótese que a partir de un cierto número de operaciones no nos cabrá la secuencia de operaciones en el cuadro de dibujo.

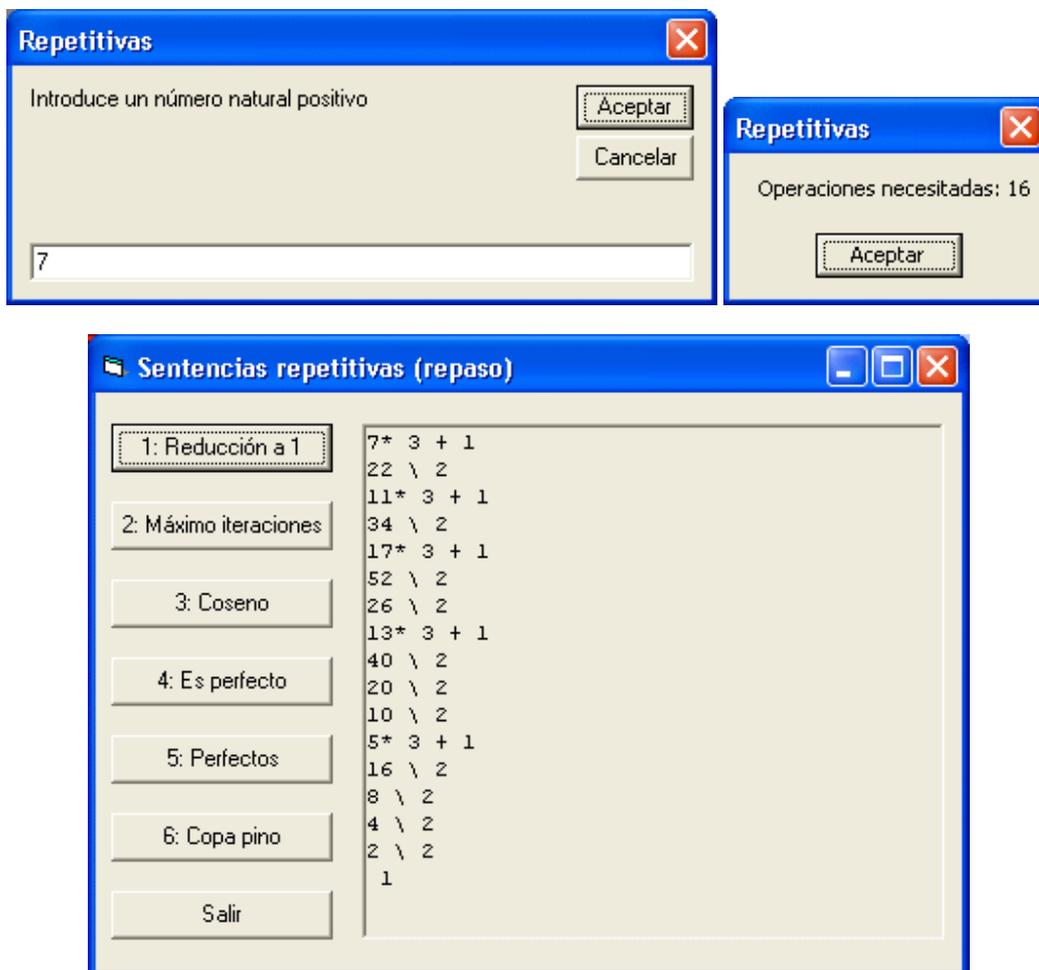


Figura 7.2. Ejemplo de reducción a 1 del número 7.

### Diagrama de flujo

En este ejercicio se propone la obtención del diagrama de flujo a partir del código VB.

## Código Visual Basic (resolución)

```

Sub cmdEj1_Click()
    Dim s As String
    Dim n As Integer
    Dim i As Integer
    pctRes.Cls
    s = InputBox ("Introduce un número natural positivo")
    n = CInt (s)
    i = 0
    While n <> 1
        i = i + 1
        If n Mod 2 = 0 Then
            pctRes.Print CStr (n) & " \ 2"
            n = n \ 2
        Else
            pctRes.Print CStr (n) & "* 3 + 1"
            n = n * 3 + 1
        End If
    Wend
    pctRes.Print CStr (n)
    MsgBox "Operaciones necesitadas: " & CStr (i)
End Sub

```

Figura 7.3 Código VB para la reducción a 1.

1. **Diagrama de flujo** del programa de reducción a uno, parcialmente resuelto.
2. **Diseña** el diagrama de flujo y **codifica** un programa VB que lea un número **tope** y verifique cuál de los números de uno a **tope** requiere la mayor cantidad de iteraciones para reducirse a 1, como en el ejercicio 7.1.
3. Diseña el **diagrama de flujo** y **codifica** el programa VB que lea un número positivo **n** dibuje la **copa de un pino** de dimensión **n**. En la Figura 7.4 se muestran varios ejemplos. Nota: se puede evitar el saldo de línea de la orden **Print** de un cuadro de dibujo añadiendo un **punto y coma** tras el texto a imprimir.

*	* ***	* *** *****	* *** ***** *****
n = 1	n = 2	n = 3	n = 4

Figura 7.4. Ejemplos de copas de pinos de dimensiones 1, 2, 3 y 4.

4. Diseña el **diagrama de flujo** y **codifica** el programa VB que lea un número positivo **n** y nos diga si es **perfecto**. Un número **n** es perfecto cuando la suma de sus divisores (exceptuando **n**) es igual al número **n**. Por ejemplo, el 6 es perfecto porque  $1+2+3 = 6$ .
5. Diseña el **diagrama de flujo** y **codifica** el programa VB que lea un número **tope** y verifique cuáles de los números de 1 a **tope** son perfectos.
6. Diseña el **diagrama de flujo** y **codifica** un programa VB que lea un ángulo en **radianes** y calcule su **coseno**, utilizando el desarrollo de **Taylor** con un **error (valor absoluto de la diferencia** entre dos aproximaciones) inferior a 0,000001.

$$\cos(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

**Datos de prueba**

Ángulo (radianes)	Coseno
1,0	0,540302303791887
1,5	7,07372049851851E-02
2,0	-0,416146839638903

**Algoritmo coseno (resolución)**

En la Figura 7.5 se proporciona una posible resolución del diagrama de flujo del programa propuesto.

La variable **fact2i** es de tipo **real** para permitir valores superiores. Si no, para un valor de  $i=7$  se produce un sobrepasamiento (*overflow*).

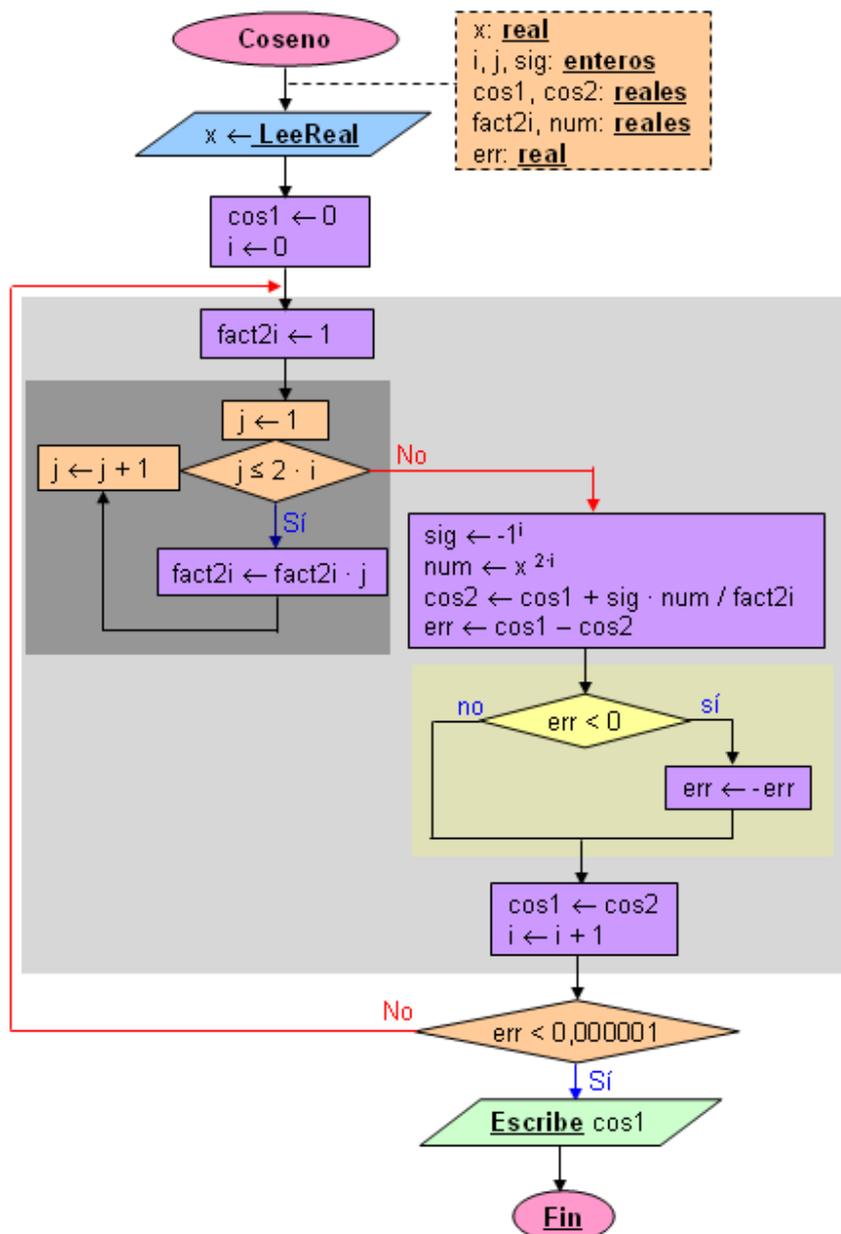


Figura 7.5. Diagrama de flujo del cálculo del coseno.