

**Objetivos:**

- ❖ Diseñar **diagramas de flujo** de programas con **sentencias condicionales**
- ❖ Codificar programas VB con **sentencias condicionales** ([If](#), [If-Else](#)).
- ❖ Expresiones lógicas.
- ❖ Propiedades de los controles: **visible**

## Sumador con comprobación (1)

En el laboratorio anterior, en el ejercicio 2.1 del sumador, si pulsamos directamente sobre el botón “Sumar” o introducimos un texto alfabético cualquiera (ver Figura 3.1) obtendremos un **mensaje de error** como el que se muestra en la Figura 3.2. En la Figura 3.3 se muestra la porción de código que se ve afectada.



**Figura 3.1** Entrada incorrecta de datos en el sumador.



**Figura 3.2** Mensaje de error de tipos

```

Dim s As String
Dim op1 As Integer

s = txtOp1.Text
op1 = CInt (s)
  
```

**Figura 3.3** Porción de código donde se produce el error.

La razón del error, cuyo código puede verse en la Figura 3.3, es que en la asignación Visual Basic intenta convertir el contenido del cuadro de texto recogido en la variable `s` a un número entero (`op1`) y esto no es posible en estas circunstancias, por ej., `CInt ("angel")`.

Las instrucciones condicionales nos permiten comprobar si los datos introducidos son adecuados antes de seguir con los cálculos. Para ello en los diagramas de flujo podremos verificar si una cadena contiene un valor numérico mediante el predicado Numérico (*cadena*) dentro de una cláusula condicional. El diagrama de flujo correspondiente se puede ver en la Figura 3.4.

Para el programa Visual Basic esta verificación se expresará mediante la función IsNumeric que recibe una cadena de caracteres y nos devuelve True si la cadena es convertible a un valor numérico y False en caso contrario.

Tras asegurarnos que se detectan estas circunstancias el código resultante del botón de suma sería el de la Figura 3.5.

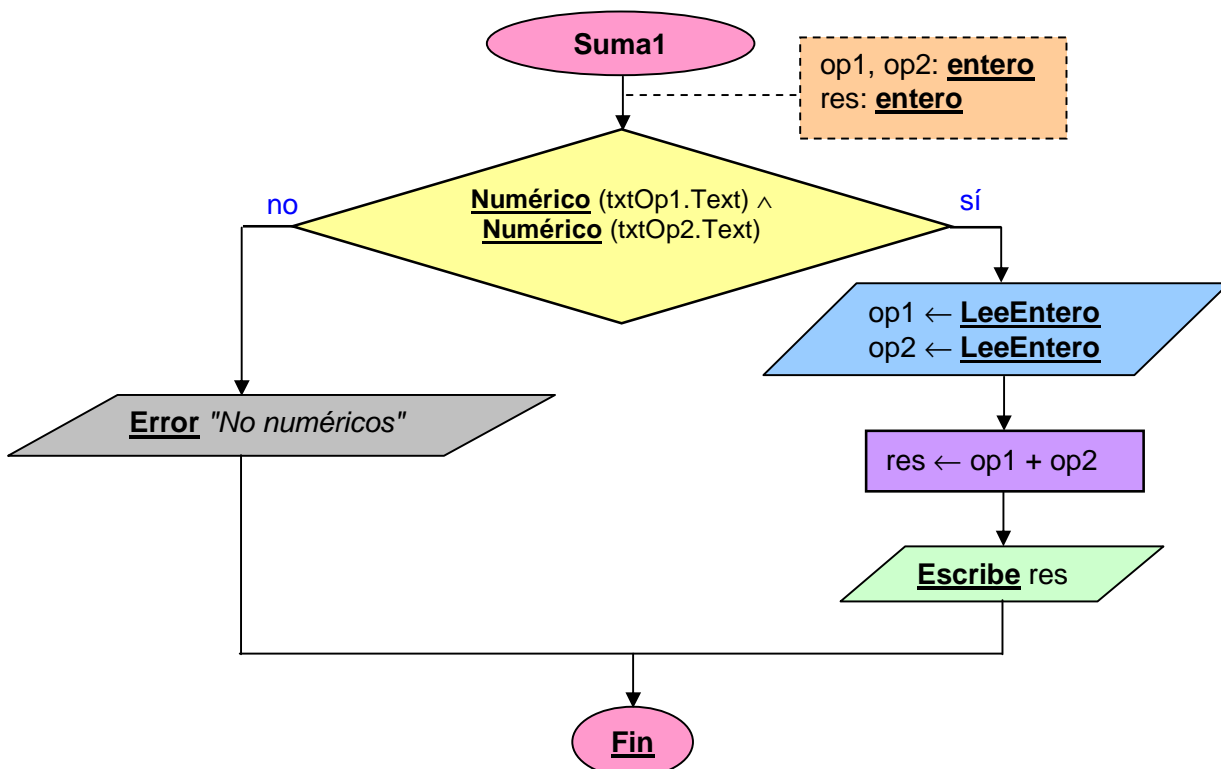


Figura 3.4 Diagrama de flujo de la suma con comprobación.

```

Sub cmdSum1_Click()
    Dim op1 As Integer, op2 As Integer
    Dim res As Integer

    If IsNumeric(txtOp1.Text) And _
       IsNumeric(txtOp2.Text) Then
        op1 = CInt (txtOp1.Text)
        op2 = CInt (txtOp2.Text)
        res = op1 + op2
        txtRes.Text = CStr (res)
    Else
        MsgBox "Los operandos han de ser numéricos"
    End If
End Sub
  
```

Figura 3.5 Subprograma de suma con comprobación.

## Sumador con comprobación (2)

De la manera que hemos resuelto el problema en el apartado anterior del sumador con comprobación (1) el usuario no tiene una información directa de cuál de los operandos es incorrecto.

Para diferenciar los casos necesitamos dos comprobaciones con dos mensajes de error diferentes. El diagrama de flujo que contempla esta posibilidad puede verse en la Figura 3.6.

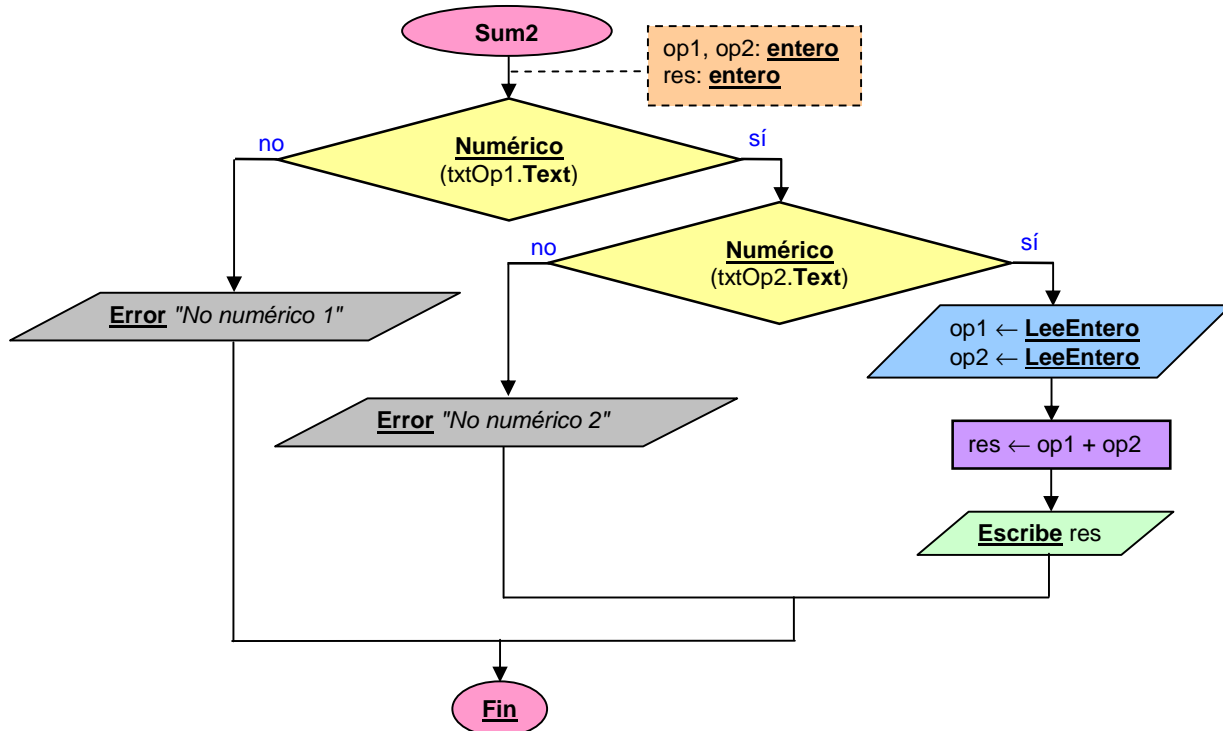


Figura 3.6 Diagrama de flujo de la suma con dos comprobaciones.

```

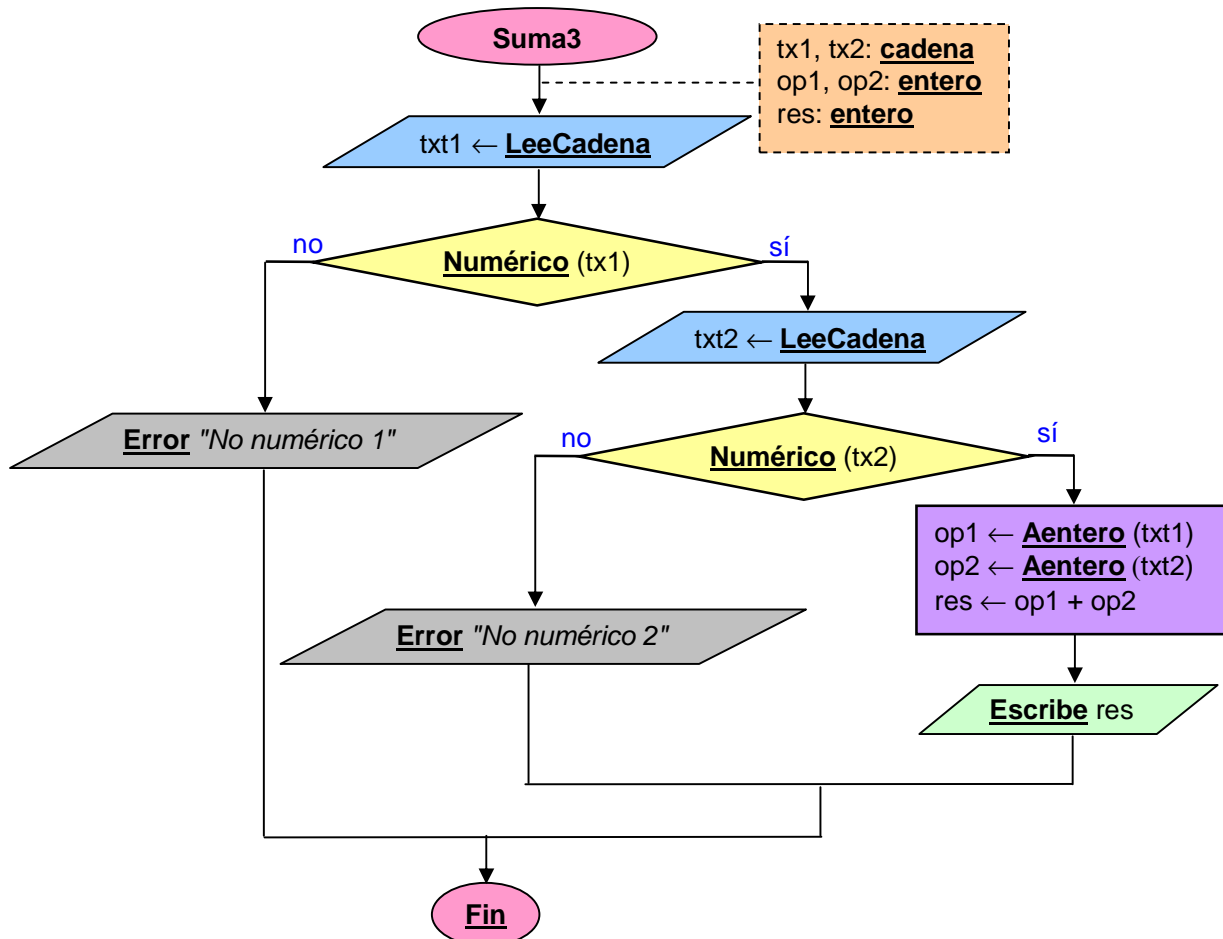
Sub cmdSum2_Click()
    Dim op1 As Integer, op2 As Integer
    Dim res As Integer

    If IsNumeric(txtOp1.Text) Then
        If IsNumeric(txtOp2.Text) Then
            op1 = CInt (txtOp1.Text)
            op2 = CInt (txtOp2.Text)
            res = op1 + op2
            txtRes.Text = CStr (res)
        Else
            MsgBox ("Operador 2 incorrecto")
        End If
    Else
        MsgBox ("Operador 1 incorrecto")
    End If
End Sub
  
```

Figura 3.7 Subprograma de suma con dos comprobaciones.

## Sumador con comprobación (3)

Para los dos apartados anteriores hemos utilizado un modelo de lectura de cajas de texto (*Text Boxes*). Para realizar una comprobación similar utilizando un modelo **InputBox** tendremos que leer los números a variables de tipo cadena, realizar la comprobación sobre éstas y en caso de corresponderse adecuadamente con valores numéricos obtener los números correspondientes y realizar el cálculo. El diagrama de flujo de este modelo puede verse en la Figura 3.8.



**Figura 3.8** Diagrama de flujo de la suma con lecturas **InputBox**.

```

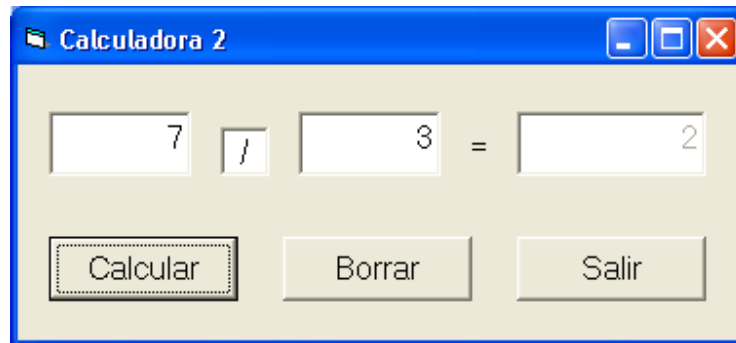
Sub cmdSum3_Click()
    Dim str1 As String, str2 As String
    Dim op1 As Integer, op2 As Integer
    Dim res As Integer
    str1 = InputBox ("Introduce primer sumando")
    If IsNumeric(str1) Then
        str2 = InputBox ("Introduce segundo sumando")
        If IsNumeric(str2) Then
            op1 = CInt (str1)
            op2 = CInt (str2)
            ...
  
```

**Figura 3.9** Porción del subprograma de suma utilizando **InputBox**.

## Ejercicio 3.1: Programa calculadora 2

Diseña el diagrama de flujo, la **interfaz** y **codifica** la calculadora de la Figura 3.10.

### Interfaz



**Figura 3.10** Interfaz del segundo modelo de calculadora.

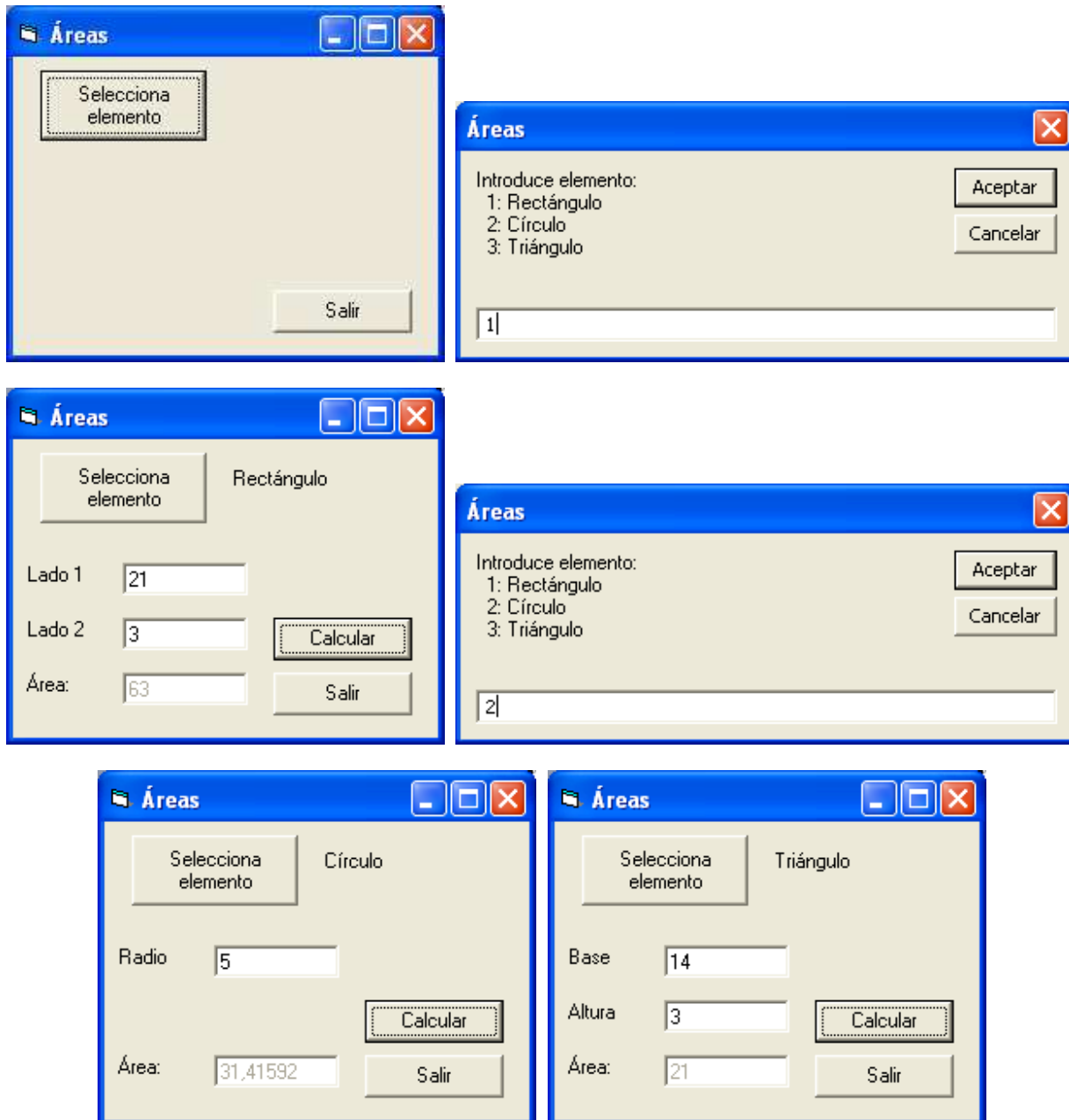
### Funcionamiento

1. Esta aplicación permite calcular los resultados de las operaciones **suma** (+), **resta** (−), **multiplicación** (\*), **cociente** de la división entera (/) y el **resto** de la división entera (%). Nótese que en Visual Basic el cociente de la división entera se representa mediante la **barra invertida** (\) y el resto de la división entera mediante la palabra reservada [Mod](#).
2. Se **controlará** que los valores introducidos en las cajas de texto son correctos (numéricos) así como que no se intente dividir por cero.
3. También se **controlará** que el operador sea uno de los cinco especificados: '+', '−', '\*', '/' ó '%'.
4. El cuadro de texto donde se presentan los resultados **no es editable**.
5. Los botones del programa actuarán como sigue:
  - **Botón Calcular:** Realiza la operación.
  - **Botón Borrar:** Limpia el contenido de las cajas de texto.
  - **Botón Salir:** Finaliza la ejecución del programa.

## Ejercicio 3.2: Programa de cálculo de áreas

**Diseña la interfaz** y **codifica** la calculadora de áreas de la Figura 3.11. No es necesario que dibujes el diagrama de flujo. Todos los datos serán reales.

### Interfaces



**Figura 3.11** Colección de interfaces del programa de cálculo de área.

## Funcionamiento

1. Esta aplicación permite calcular el área de un **rectángulo**, un **círculo** o un **triángulo**. En cada uno de los casos pedirá los datos necesarios, ocultando para ello los objetos que no sean necesarios (poniendo a **False** la propiedad **Visible**) y modificando las etiquetas (propiedad **Caption**) en caso de que sea necesario. El cuadro de texto donde se presentan los resultados **no es editable**.

Nota: Como en el ejercicio anterior controla que los valores introducidos en las cajas de texto son correctos (son numéricos).

2. Los botones del programa actuarán como sigue:
  - **Botón “Selecciona elemento”**: lanza un **MsgBox** proponiendo los elementos a calcular el área. Para poder partir en líneas concatenaremos **vbCrLf** a la cadena como fin de línea. Verificará que la opción introducida sea una opción válida (1, 2 ó 3). Visualiza los elementos apropiados con las etiquetas pertinentes, ocultando los objetos que no sean adecuados.
  - **Botón “Calcular”**: Realiza la operación.
  - **Botón “Salir”**: Finaliza la ejecución del programa.