



Objetivos:

- ❖ Adquirir habilidades en el uso de **sentencias condicionales** (**if**, **select case**).
- ❖ Introducción a los **subprogramas**
- ❖ Identificar las **propiedades** de los objetos gráficos: alineamiento, fuentes, visibilidad, ...
- ❖ **Cambiar dinámicamente** el valor de las **propiedades** gráficas de los objetos
- ❖ Copiado y pegado de código

Programa de resolución de ecuaciones de primero y segundo grado

Interfaz:

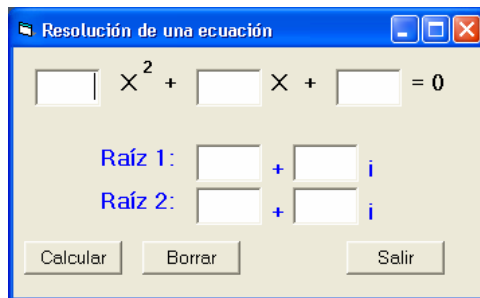


Figura 1: Objetos presentes en la interfaz de la calculadora de ecuaciones

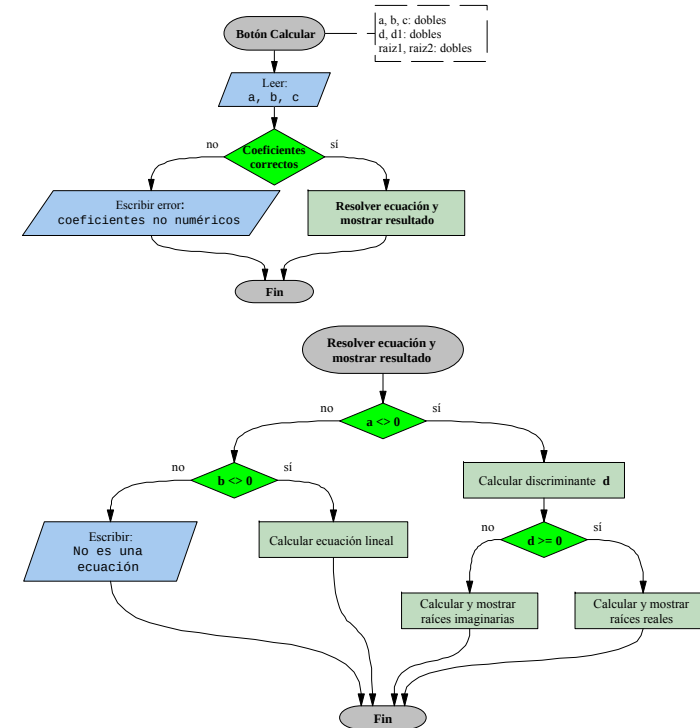
Funcionamiento:

- Al lanzar el programa los objetos del **resultado** (en azul) estarán **invisibles**
- Al hacer **click** en el botón **Calcular**, se debe:
 - **Recoger el valor de los coeficientes a, b y c** de la ecuación
 - Calcular las **raíces**, pudiendo ser (si es $a \neq 0$ y $b \neq 0$):
 - De **primer grado** (ecuación lineal, si $a = 0$)
 - De **segundo grado**, pudiendo ser **real**, cuando el discriminante sea positivo o nulo, o **imaginaria** (números complejos), cuando el discriminante es negativo.
 - Bloquear los campos para que no se modifiquen los valores y se pierda la coherencia
- Al hacer **click** en el botón **Borrar** se debe **borrar el contenido de las cajas de texto volviendo todos los objetos al estado inicial**



- Al hacer **click** en el botón **Salir**, el **programa finalizará**. Para ello usaremos la función **End**.

Algoritmos:



Fórmula general	1º grado	2º grado reales	2º grado imaginarias
$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	$r = \frac{-c}{b}$	$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ $r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$	$r_1 = \frac{-b}{2a} + \frac{\sqrt{4ac - b^2}}{2a}i$ $r_2 = \frac{-b}{2a} - \frac{\sqrt{4ac - b^2}}{2a}i$
Caso particular:	$a = 0$	$b^2 - 4ac \geq 0$	$b^2 - 4ac < 0$



Pasos a seguir:

1. Crearemos los objetos del tipo y forma mostrados en la Figura 1. Sólo daremos un nombre particular a aquéllos que nos interesa leer o modificar sus propiedades en algún momento del programa. En la Figura 2 se muestran en rojo los nombres de estos objetos.

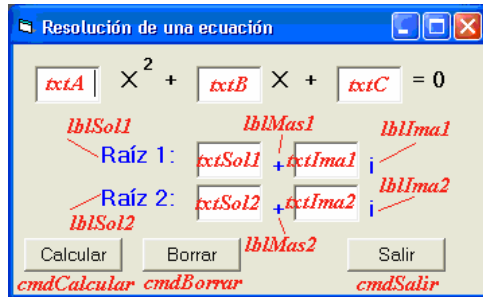


Figura 2: Nombres de los objetos en la interfaz de la calculadora de ecuaciones

2. Añadir el código a los eventos, es decir, la carga del formulario y el clic sobre los botones:
 - **Código de carga del formulario:** para especificar el código de carga del formulario haremos clic dos veces sobre el formulario. Lo que conviene hacer es inhabilitar las cajas de texto de la solución (de una vez por todas) y poner todos los objetos asociados a la solución invisibles (que haremos visibles cuando sea preciso):

```
Private Sub Form_Load()
    ponerSolucionInvisible
    txtSol1.Enabled = False
    txtSol2.Enabled = False
    txtIma1.Enabled = False
    txtIma2.Enabled = False
End Sub
```

Se observa que hemos escrito **ponerSolucionInvisible**, que es un subprograma que escribiremos inmediatamente a continuación.

Lo que tendremos que hacer es especificar uno a uno que los objetos relacionados con la solución van a estar invisibles. Luego, dependiendo del tipo de solución, haremos



visibles de manera selectiva algunos de ellos, e incluso cambiaremos las etiquetas según convenga:

```
Private Sub ponerSolucionInvisible()
    lblSol1.Visible = False
    lblSol2.Visible = False
    txtSol1.Visible = False
    txtSol2.Visible = False
    lblMas1.Visible = False
    lblMas2.Visible = False
    txtIma1.Visible = False
    txtIma2.Visible = False
    lblIma1.Visible = False
    lblIma2.Visible = False
End Sub
```

- **Código del botón Calcular:** se controlará la validez de los coeficientes (han de ser numéricos) y en función de los valores introducidos por el usuario, se determina el tipo de ecuación y se calcularán los resultados. Veamos el código paso a paso.

Primero **declararemos** las variables para los coeficientes **a**, **b** y **c** como **Doubles** (reales), así como otras especificadas en el algoritmo. Luego **controlaremos** que sean **numéricas** y si es así **leeremos** su valor a sus variables correspondientes.

```
Private Sub cmdCalcular_Click()
    Dim a As Double
    Dim b As Double
    Dim c As Double
    Dim d As Double ' Discriminante
    Dim d1 As Double ' Raíz cuadrada del discriminante
    Dim raiz1 As Single
    Dim raiz2 As Single

    ' Obtener los valores de los coeficientes
    If IsNumeric(txtA.Text) And _
        IsNumeric(txtB.Text) And _
        IsNumeric(txtC.Text) Then
        ' Los valores son numéricos
        a = Val(txtA.Text)
        b = Val(txtB.Text)
        c = Val(txtC.Text)

        If (a <> 0) Then
            ' Ecuación de segundo grado

... Aquí irá el código adicional, que veremos a continuación

        Else
            MsgBox ("Error: Coeficientes no numéricos")
        End If
    End Sub ' Fin de cmdCalcular_Click
```



Tras leer el valor de las variables, siguiendo el algoritmo, determinaremos si se trata de una ecuación de **segundo grado**, de **primer grado** o si **no se trata de una ecuación**. Habrá que tener cuidado con la indentación (sangrado) ya que afecta de manera importante a la legibilidad:

```
If (a <> 0) Then
    ' Ecuación de segundo grado
... Aquí irá el código adicional, que veremos a continuación

ElseIf b <> 0 Then
    ' Ecuación de primer grado
... Aquí irá el código adicional, que veremos a continuación

Else
    MsgBox ("Error: No es una ecuación")
End If
Else
    MsgBox ("Error: Coeficientes no numéricos")
```

Para las ecuaciones de segundo grado (coeficiente **a** $\neq$ 0) calcularemos el **discriminante** (variable **d**). Si éste es **nulo** tendremos **dos raíces reales idénticas**, si es **positivo** tendremos **dos raíces reales** y si es **negativo** tendremos **dos raíces imaginarias**. Tendremos dos modos distintos de visualización que se traducirán en los subprogramas **ponerEcuacionRealSegundoGrado**, para visualizar solamente los campos de la solución real, y **ponerEcuacionImagSegundoGrado**, para visualizar también la parte imaginaria de las raíces complejas.

```
' Ecuación de segundo grado
d = b ^ 2 - 4 * a * c ' Cálculo del discriminante
If d >= 0 Then
    ' Soluciones reales
    d1 = Sqr(d)
    txtSol1.Text = (-b + d1) / (2 * a)
    txtSol2.Text = (-b - d1) / (2 * a)
    ponerEcuacionRealSegundoGrado
Else
    ' Soluciones imaginarias
    d1 = Sqr(-d)
    txtSol1.Text = -b / (2 * a)
    txtSol2.Text = -b / (2 * a)
    txtIma1.Text = d1 / (2 * a)
    txtIma2.Text = -d1 / (2 * a)
    ponerEcuacionImagSegundoGrado
End If
ElseIf b <> 0 Then
```



El código asociado al subprograma **ponerEcuacionRealSegundoGrado** se encarga de hacer visibles los objetos necesarios para mostrar la solución de las raíces reales, y es el siguiente:

```
Private Sub ponerEcuacionRealSegundoGrado()
    bloquearABC
    lblSol1.Caption = "Raíz 1:"
    lblSol1.Visible = True
    txtSol1.Visible = True
    lblSol2.Caption = "Raíz 2:"
    lblSol2.Visible = True
    txtSol2.Visible = True
End Sub
```

Siempre que vayamos a mostrar la solución a una ecuación vamos a **bloquear** los objetos de **entrada de los coeficientes** de la ecuación, ya que si permitimos modificarlos no serán consistentes con las soluciones mostradas. Este bloqueo lo haremos con el subprograma **bloquearABC**. El código para esto será:

```
Private Sub bloquearABC()
    txtA.Enabled = False
    txtB.Enabled = False
    txtC.Enabled = False
End Sub
```

Cuando se trata de una ecuación de segundo grado con soluciones imaginarias tendremos que hacer visibles los mismos objetos que los de la solución anterior, lo cual haremos mediante un nuevo subprograma al que llamaremos **ponerEcuacionImagSegundoGrado** y luego llamaremos al subprograma anterior **ponerEcuacionRealSegundoGrado**, además de visualizar los objetos correspondientes a la parte imaginaria:

```
Private Sub ponerEcuacionImagSegundoGrado()
    ponerEcuacionRealSegundoGrado
    lblMas1.Visible = True
    lblMas2.Visible = True
    txtIma1.Visible = True
    txtIma2.Visible = True
    lblIma1.Visible = True
    lblIma2.Visible = True
End Sub
```

Finalmente, cuando el coeficiente **a** es nulo nos encontramos ante una ecuación lineal, que sólo tendrá una raíz. Con esto finalizaremos el código asociado al botón **Calcular** y añadiremos un subprograma para visualizar una única raíz:



```
ElseIf b <> 0 Then
    ' Ecuación de primer grado
    txtSol1.Text = -c / b
    ponerEcuacionPrimerGrado
Else
    MsgBox ("Error: No es una ecuación")
```

El código del subprograma **ponerEcuacionPrimerGrado** es más sencillo que los anteriores:

```
Private Sub ponerEcuacionPrimerGrado()
    bloquearABC
    lblSol1.Caption = "Raíz:"
    lblSol1.Visible = True
    txtSol1.Visible = True
End Sub
```

Código del botón Borrar: antes hemos creado un subprograma para borrar todos los objetos de la solución al cargar el formulario llamada **ponerSolucionInvisible**. Además de eso hemos de desbloquear los campos de los coeficientes a, b y c, poniéndolos vacíos:

```
Private Sub cmdBorrar_Click()
    ponerSolucionInvisible
    desbloquearABC
End Sub
```

El subprograma para desbloquear los coeficientes es fácilmente adivinable:

```
Private Sub desbloquearABC()
    txtA.Enabled = True
    txtB.Enabled = True
    txtC.Enabled = True
    txtA.Text = ""
    txtB.Text = ""
    txtC.Text = ""
End Sub
```

Código del botón Salir. Al igual que en ejemplos anteriores, terminaremos la ejecución del programa mediante la orden **End**:

```
Private Sub cmdSalir_Click()
    End
End Sub
```

Nota: Crea un nuevo proyecto por cada ejercicio que resuelvas



Programa rodillo

Interfaz:

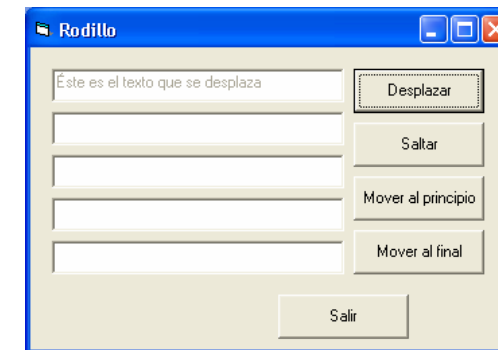


Figura 3: Objetos presentes en la interfaz del rodillo

Funcionamiento:

El programa contiene 5 cuadros de texto **no editables** (propiedad *Enabled* a *False*). El primero de ellos contiene el texto “*Este es el texto que se desplaza*”. Los botones de la ventana permiten desplazar el texto.

1. **Botón Desplazar:** El texto se desplaza a la **siguiente posición** de modo circular. Si el texto se encuentra en la última posición, éste pasa a la primera.
2. **Botón Saltar 1:** El texto se desplaza de manera circular **dejando un hueco** entre la posición que ocupaba y la nueva posición. Cuando llega al final vuelve a desplazar por arriba.
3. **Botón Mover al principio:** Coloca el texto en la **primera casilla**.
4. **Botón Mover al final:** Coloca el texto en la **última casilla**.

Recomendaciones: Utiliza las sentencias condicionales para determinar en qué casilla se encuentra el texto y así decidir la nueva ubicación de éste, según el botón.

Copia el texto de la casilla origen a la casilla destino y luego pon la casilla origen a cadena vacía.



Programa calculadora 2

Interfaz:

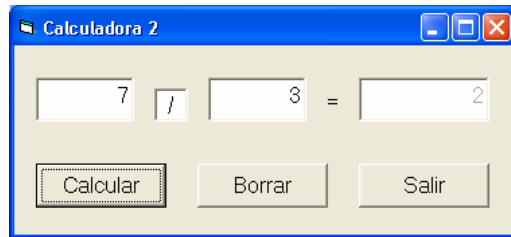


Figura 4: Interfaz del segundo modelo de calculadora

Funcionamiento:

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**.

Se **controlará** que los valores introducidos en las cajas de texto son correctos (numéricos) así como que no se intente dividir por cero.

También se **controlará** que el operador sea uno de los cinco especificados: +, -, *, / o \.

El cuadro de texto donde se presentan los resultados **no es editable**.

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.



Programa de cálculo de áreas

Interfaces:

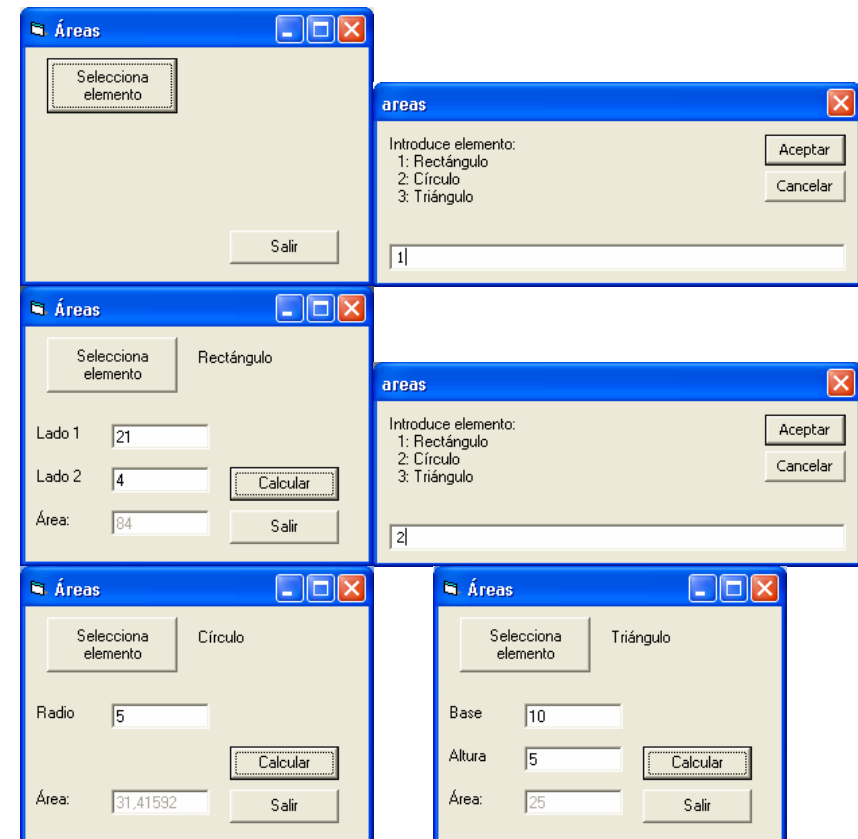


Figura 5: Colección de interfaces del programa de cálculo de área



Funcionamiento:

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 y modificando las etiquetas 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).

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. 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.

Referencia rápida – Sintaxis		
If Condición Then Instr1 [Else Instr 2]	Select Case Expresión	Ejemplo
If Condición 1 Then Lista sentencias 1	Case Opción 1 Lista sentencias 1	Select Case X+2
ElseIf Condición 2 Then Lista sentencias 2	Case Opción 2 Lista sentencias 2	Case 0
...	...	Case Is >0
Case Opción n Lista sentencias n	Case Opción n Lista sentencias n	Case "A", "a"
Else Lista sentencias n	Case Else Lista sentencias n+1	Case 0 To 10
End	End Select	Case Else
		End Select



Programa de cálculo de la fecha siguiente

Interfaz:

Figura 6: Interfaz del programa de cálculo de fechas

Funcionamiento:

El programa trabaja con fechas (probar ejecutable).

1. **Botón +:** Controla que la fecha introducida es correcta (años positivos de cuatro dígitos) y calcula la fecha del día siguiente, escribiéndola en los cuadros de texto correspondientes.
2. **Botón Borrar:** Elimina el contenido de las cajas de texto.
3. **Botón Salir:** Sale del programa.

Controles:

Contenido del campo día, mes y año numéricos
Día correcto: mayor o igual que 1 y menor o igual que 31
Mes correcto: mayor o igual que 1 y menor o igual que 12
Año correcto: mayor o igual que 1000 y menor o igual que 9999
Fecha correcta: no hay día 31 en febrero, abril, junio, septiembre o noviembre
Fecha correcta: no hay 30 de febrero
Fecha correcta: no hay 29 de febrero los años que no son bisiestos ¹

Recomendación:

Nótese que al incrementar el día podemos llegar a una fecha incorrecta, en cuyo caso habrá que pasar al primer día del mes siguiente y posiblemente al primer día del año siguiente. Esta verificación será similar a la del control de corrección de la fecha.

¹ Son bisiestos los años **divisibles por 4 que no lo son por 100 salvo los que son divisibles por 400**. Por ejemplo, 1996, 2000 y 2004 fueron bisiestos, pero no lo serán el 2100, 2200 y 2300, y sí 2400. A ver quiénes lo contamos.