Resolver triángulos en Visual Basic. Parte 3/3

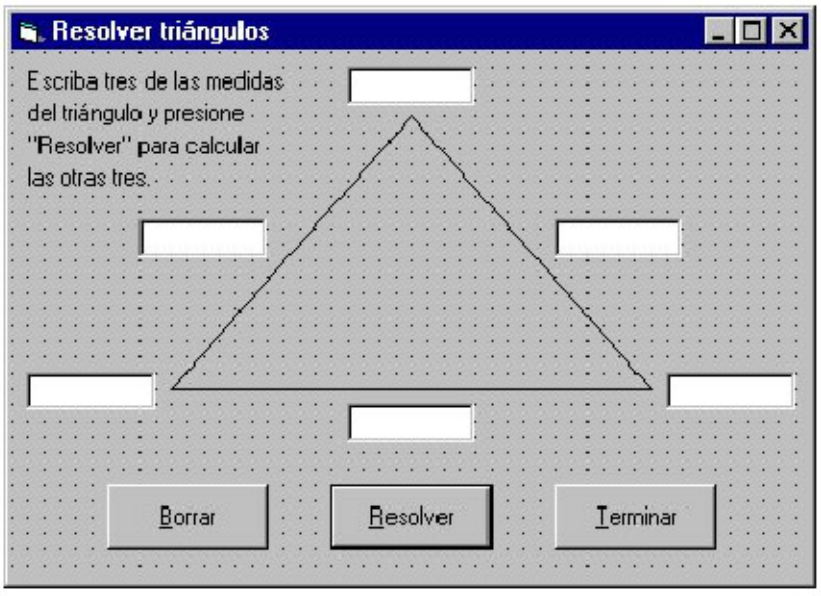
Luis Acuña P.
lacuna@itcr.ac.cr
Escuela de Matemática
Instituto Tecnológico de Costa Rica

Introducción

En las dos columnas anteriores desarrollamos un conjunto de funciones y subrutinas para resolver triángulos usando las leyes de senos y cosenos, así como un formulario para graficar la solución a escala. En esta columna terminaremos el proyecto desarrollando el formulario principal que recibirá los datos del usuario, llamará las funciones apropiadas para resolver el problema y mostrará la solución en forma gráfica.
Pero primero hay una mejora a lo que ya hicimos: En el formulario `frmTringulo` el cuadro `picTringulo` debería tener verdadera la propiedad `AutoRedraw`. Esto hará que el gráfico se redibuje automáticamente luego de que el formulario hubiere sido cubierto por otra ventana de Windows.

1.1 Diseño del formulario principal

Ahora pasemos al trabajo nuevo. El diseño del formulario principal será como en esta figura:



El triángulo está formado por tres controles de tipo `Line`. En sus vértices hay tres cuadros de texto, todos formando un arreglo llamado `txtngulo`. El superior tiene índice 1, el izquierdo 2 y el derecho 3. Para crearlos

se puede colocar un cuadro de texto en el vértice superior, darle nombre `txtngulo` y asignar el valor 1 a la propiedad `Index`. Luego se crea el segundo con el mismo nombre y con `Index` igual a 2. De manera semejante se crea el tercero.
Junto a los lados del triángulo hay un arreglo de cuadros de texto. Ellos comparten el nombre `txtLado` y tienen índices 1 (el inferior), 2 (el derecho) y 3 (el izquierdo).
Por último, hay tres botones de comando llamados `cmdBorrar` (que borrará los datos para empezar de nuevo), `cmdResolver` (que resolverá el problema) y `cmdTerminar` (que terminará el programa). El segundo tiene la propiedad `Default` verdadera para que se active cuando el usuario presiona `[Enter]`.
Ah, hay algo más. Cerca de la esquina superior izquierda hay cuatro etiquetas con el texto que se muestra en la figura. Y ahora sí es todo con respecto al diseño del formulario.

1.2 Programación del formulario principal

Resolver el triángulo no será difícil luego de todo el trabajo que ya hicimos en las columnas anteriores. Tedioso sí, porque hay muchos casos que investigar individualmente. De hecho, como son seis las medidas del triángulo y el usuario puede indicar tres cualesquiera de ellas, habrá un total de $\binom{6}{3} = 20$ posibilidades que investigar (sólo 19 son válidas, porque el caso AAA no tiene solución). Pero aparte de considerar cada una de las posibilidades, nuestro trabajo en este formulario no será complicado.
Empecemos por programar los cuadros de texto para que seleccionen todo su contenido cuando reciban el foco.

```
Private Sub txtngulo_GotFocus(Index As Integer)
'   Al recibir el foco
txtngulo(Index).SelStart = 0
txtngulo(Index).SelLength = Len(txtngulo(Index).Text)
End Sub

Private Sub txtLado_GotFocus(Index As Integer)
'   Al recibir el foco
txtLado(Index).SelStart = 0
txtLado(Index).SelLength = Len(txtLado(Index).Text)
End Sub
```

Deshagámonos de los botones de borrar y terminar, que son prácticamente triviales:

```
Private Sub cmdBorrar_Click()
'   Borrar los seis cuadros de texto
Dim i As Byte
For i = 1 To 3
    txtngulo(i).Text = ""
    txtLado(i).Text = ""
Next
End Sub

Private Sub cmdTerminar_Click()
'   Salir del programa
End
End Sub
```

1.2.1 El botón `cmdResolver`

Aquí es donde está el trabajo importante. Vamos a empezar por validar y contar los datos. Si un cuadro de texto está vacío, su valor (según la función `Val`) es cero; por eso contaremos como válidos los datos mayores

que cero y como inválidos los menores que cero, pero no contaremos los que valgan cero. En Visual Basic una expresión lógica tiene el valor 0 si es falsa o el valor -1 si es verdadera. Por eso la fórmula `Numngulos = Numngulos - (A(i) > 0)` incrementa el contador `Numngulos` sólo si `A(i)` > 0. (En la columna anterior convinimos que los tres ángulos serían `A(1)`, `A(2)` y `A(3)`, y que los lados respectivamente opuestos serían `L(1)`, `L(2)` y `L(3)`.)
La siguiente validación consiste en que el número de datos sea exactamente tres. Y si es así entonces el programa considerará los distintos casos según el número de ángulos indicados:

- Si se indican cero ángulos estamos en el caso LLL.
- Si se indica un ángulo estamos en LAL o LLA.
- Si son dos ángulos, el caso es ALA o AAL.
- Y si son tres ángulos estamos en el caso AAA, que no tiene solución.

En cada uno de los casos se investiga más en detalle cuáles exactamente son los datos para llamar la subrutina apropiada con los parámetros en el orden correcto. Cada una de las subrutinas para resolver los distintos casos retornará medidas iguales a cero cuando no haya solución; por eso la condición `A(1) * A(2) * A(3) * L(1) * L(2) * L(3) = 0` (cierta si alguna de las medidas es cero) indicará que el triángulo no tiene solución. En caso contrario, la solución encontrada se grafica en `frmGrfico`.
Bueno, no exactamente. Lo anterior estaría bien si no fuera porque algunos casos tienen dos soluciones. Para acomodar esos casos lo que haremos será crear una copia de `frmGrfico` para cada solución. En Visual Basic es muy fácil crear varias copias de un formulario con la sintaxis `Dim frmCopia As New frmOriginal`, donde `frmOriginal` es un formulario que ya existe (como `frmGrfico` en nuestro caso) y `frmCopia` es el nombre de la copia. Todo lo que necesitamos hacer para graficar la solución es escribir

```
Dim frmGraf As New frmGrfico
frmGraf.Show
```

Cuando hay dos soluciones (solamente en el caso LLA) necesitamos crear dos copias del formulario. Pero también necesitamos tres variables adicionales para la segunda solución. Vamos a crear una subrutina `LLADosSols` que reciba seis parámetros: los tres primeros, por valor, serán los datos, y los tres últimos, por referencia, serán la solución. ‰Pero son dos soluciones! Sí, entonces esta subrutina guardará la segunda solución en tres variables locales, graficará la primera para desentenderse de ella y retornará (en los parámetros por referencia) la segunda para que `cmdResolver_Click` la grafique más tarde.
La subrutina `LLADosSols` es así:

```
Private Sub LLADosSols(ByVal L1 As Double, ByVal L2 As Double, ByVal A2 As Double, _
        ByRef A1 As Double, ByRef A3 As Double, ByRef L3 As Double)
'   Resolver caso LLA con dos soluciones.
'   Guarda la segunda solucin en variables locales,
'   grafica la primera solucin y pasa la segunda a las
'   variables globales correspondientes.

' Para guardar la segunda solucin:
Dim angA2 As Double, angC2 As Double, C2 As Double

' Resolver el tringulo
CasoLLA L1, L2, A2, A1, A3, L3, angA2, angC2, C2

' Si hay segunda solucin debe haber tambin primera:
If angA2 * angC2 * C2 <> 0 Then
    ' Entonces graficar la primera y pasar datos de la segunda:
    Dim frmGraf As New frmGrfico
    frmGraf.Show
```

```
    A1 = angA2: A3 = angC2: L3 = C2
End If
End Sub
```

Como vemos, aquí se crea y se muestra una copia de `frmGrfico` en una variable local llamada `frmGraf`. Normalmente las variables locales desaparecen al terminar una subrutina, pero ahora el formulario `frmGraf` permanece mientras sea visible, aunque la subrutina haya terminado. El usuario decidirá cuándo lo cierra con el botón "Cerrar" del gráfico o el botón "Terminar" del formulario principal.
Al retornar el control a `cmdResolver_Click`, la subrutina `LLADosSols` habrá dejado en las variables globales correspondientes la segunda solución, la cual será graficada al final de `cmdResolver_Click`. Esto es lo que falta en el programa:

```
Private Sub cmdResolver_Click()
'   Calcular las tres medidas restantes
Dim i As Byte
Dim Numngulos As Byte, NumLados As Byte    ' nmeros de datos

' Leer, validar y contar los datos:
For i = 1 To 3
    ' ngulo i-simo:
    A(i) = Val(txtngulo(i).Text) * pi_180
    If A(i) < 0 Then
        MsgBox "El ngulo " & i & " no puede ser negativo.", _
            vbCritical, "Error en los datos"
        txtngulo(i).SetFocus
        Exit Sub
    End If
    Numngulos = Numngulos - (A(i) > 0)    ' contarlo

    ' Lado i-simo:
    L(i) = Val(txtLado(i).Text)
    If L(i) < 0 Then
        MsgBox "El lado " & i & " no puede ser negativo.", _
            vbCritical, "Error en los datos"
        txtLado(i).SetFocus
        Exit Sub
    End If
    NumLados = NumLados - (L(i) > 0)        ' contarlo
Next

' Ver que sean exactamente tres datos:
If Numngulos + NumLados <> 3 Then
    MsgBox "El nmero de datos debe ser igual a 3.", _
        vbCritical, "Error en los datos"
    Exit Sub
End If

' Proceder segn el nmero de ngulos conocidos
Select Case Numngulos

    Case 0      ' tres lados conocidos: LLL
        CasoLLL L(1), L(2), L(3), A(1), A(2), A(3)

    Case 1      ' un ngulo y dos lados: LAL o LLA
        ' El ngulo conocido puede ser A(1), A(2) o A(3)
```

```
    If A(1) > 0 Then
        ' El lado desconocido puede ser L(1), L(2) o L(3)
        If L(1) = 0 Then        ' LAL: L2,A1,L3
            CasoLAL L(2), A(1), L(3), A(2), L(1), A(3)
        ElseIf L(2) = 0 Then    ' LLA: L3,L1,A1
            LLADosSols L(3), L(1), A(1), A(3), A(2), L(2)
        Else                    ' LLA: L2,L1,A1
            LLADosSols L(2), L(1), A(1), A(2), A(3), L(3)
        End If

    ElseIf A(2) > 0 Then
        If L(2) = 0 Then        ' LAL: L1,A2,L3
            CasoLAL L(1), A(2), L(3), A(1), L(2), A(3)
        ElseIf L(1) = 0 Then    ' LLA: L3,L2,A2
            LLADosSols L(3), L(2), A(2), A(3), A(1), L(1)
        Else                    ' LLA: L1,L2,A2
            LLADosSols L(1), L(2), A(2), A(1), A(3), L(3)
        End If

    ElseIf A(3) > 0 Then
        If L(3) = 0 Then        ' LAL: L1,A3,L2
            CasoLAL L(1), A(3), L(2), A(1), L(3), A(2)
        ElseIf L(2) = 0 Then    ' LLA: L1,L3,A3
            LLADosSols L(1), L(3), A(3), A(1), A(2), L(2)
        Else                    ' LLA: L2,L3,A3
            LLADosSols L(2), L(3), A(3), A(2), A(1), L(1)
        End If
    End If

Case 2      ' dos ngulos y un lado: ALA o AAL
    ' El lado conocido puede ser L(1), L(2) o L(3)
    If L(1) > 0 Then
        ' El ngulo desconocido puede ser A(1), A(2) o A(3)
        If A(1) = 0 Then        ' ALA: A2,L1,A3
            CasoALA A(2), L(1), A(3), L(2), A(1), L(3)
        ElseIf A(2) = 0 Then    ' AAL: A1,A3,L1
            CasoAAL A(1), A(3), L(1), A(2), L(3), L(2)
        Else                    ' AAL: A1,A2,L1
            CasoAAL A(1), A(2), L(1), A(3), L(2), L(3)
        End If

    ElseIf L(2) > 0 Then
        If A(2) = 0 Then        ' ALA: A1,L2,A3
            CasoALA A(1), L(2), A(3), L(1), A(2), L(3)
        ElseIf A(1) = 0 Then    ' AAL: A2,A3,L2
            CasoAAL A(2), A(3), L(2), A(1), L(3), L(1)
        Else                    ' AAL: A2,A1,L2
            CasoAAL A(2), A(1), L(2), A(3), L(1), L(3)
        End If

    ElseIf L(3) > 0 Then
        If A(3) = 0 Then        ' ALA: A1,L3,A2
            CasoALA A(1), L(3), A(2), L(1), A(3), L(2)
        ElseIf A(2) = 0 Then    ' AAL: A3,A1,L3
```

```
                CasoAAL A(3), A(1), L(3), A(2), L(1), L(2)
            Else                    ' AAL: A3,A2,L3
                CasoAAL A(3), A(2), L(3), A(1), L(2), L(1)
            End If
        End If

    Case 3      ' tres ngulos: AAA
        MsgBox "No hay solucin si slo se conocen los ngulos.", _
            vbCritical, "Error en los datos"
        Exit Sub
End Select

' Hay solucin?
If A(1) * A(2) * A(3) * L(1) * L(2) * L(3) = 0 Then
    MsgBox "No hay solucin para esos datos.", _
        vbCritical, "Datos invlidos"
Else
    ' Crear nuevo formulario para esta solucin
    Dim frmGraf As New frmGrfico
    frmGraf.Show
End If

End Sub
```

1.3 Conclusión

El programa ahora calcula la o las soluciones para cualquier conjunto de datos, o indica si el conjunto es inválido. Los formularios con las soluciones se conservan hasta que el usuario los cierre individualmente (con el botón “Cerrar” de cada gráfico) o termine el programa (con el botón “Terminar” del formulario principal, que cerrará todos los formularios).
Y así terminamos un gran proyecto, con muy buenos resultados. En la siguiente figura vemos las soluciones del ejemplo que mencionamos en la columna anterior: $a = 15$, $b = 13$ y $B = 53.13^\circ$.