Sección
Revista digital Matemática, Educación e Internet (www.cidse.itcr.ac.cr/revistamate/). Vol. 2, Nº 2. Agosto – Diciembre 2002.

# Resolver triángulos en Visual Basic. Parte 2/3

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

## Introducción

En la columna anterior habíamos presentado la teoría necesaria para resolver triángulos con las leyes de senos y cosenos. También vimos tres subrutinas para resolver los casos AAL, ALA, LAL y LLL (la primera subrutina resuelve los dos primeros casos). Dejamos pendientes la solución del caso LLA y el resto del programa. En esta columna resolveremos el caso faltante y desarrollaremos el formulario que graficará las soluciones.

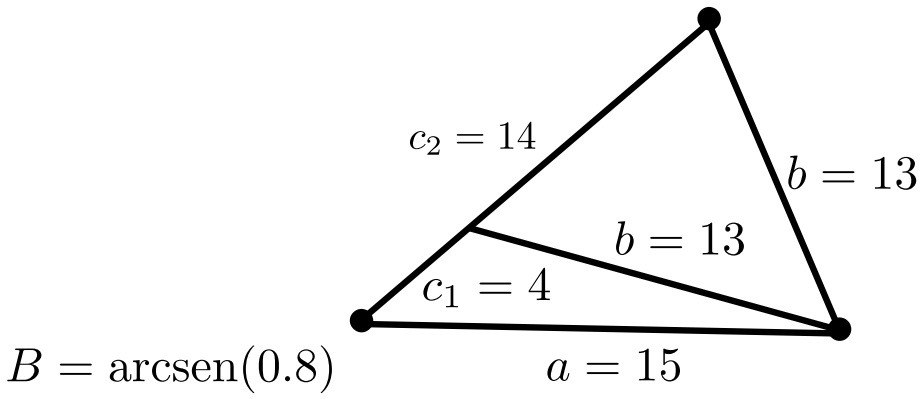
Pero antes de entrar en materia debemos corregir un error en la columna anterior. Los casos AAL y ALA en realidad no pueden resolverse con la misma subrutina com habíamos dicho. La subrutina que dimos en esa columna con nombre `CasoAAL` en realidad resuelve solamente el caso ALA. En vez de esa debemos tener las dos siguientes:

```
Public Sub CasoALA(ByVal angA As Double, ByVal c As Double, ByVal angB As Double, _
    ByRef A As Double, ByRef angC As Double, ByRef b As Double)
'   Resuelve el caso de ngulo-lado-ngulo conocidos.
'   Calcula primero el ngulo desconocido y luego
'   los dos lados por la ley de senos.
'   Recibe los datos angA, angB y c, y calcula
'   los resultados a, b y angC.

    ' validar los datos: debe ser angA+angB<pi y c>0
If angA + angB >= pi Or c <= 0 Then
    A = 0: b = 0: angC = 0
    Exit Sub
End If

    ' calcular el ngulo C por diferencia a pi
angC = pi - angA - angB
    ' calcular el lado a por ley de senos
A = c * Sin(angA) / Sin(angC)
    ' calcular el lado b por ley de senos
b = c * Sin(angB) / Sin(angC)
End Sub

Public Sub CasoAAL(ByVal angA As Double, ByVal angB As Double, ByVal A As Double, _
    ByRef angC As Double, ByRef b As Double, ByRef c As Double)
'   Resuelve el caso de ngulo-ngulo-lado conocidos.
```

```
'   Calcula primero el ngulo desconocido y luego
'   los dos lados por la ley de senos.
'   Recibe los datos angA, angB y c, y calcula
'   los resultados a, b y angC.

    ' validar los datos: debe ser angA+angB<pi y c>0
If angA + angB >= pi Or A <= 0 Then
    A = 0: b = 0: angC = 0
    Exit Sub
End If

    ' calcular el ngulo C por diferencia a pi
angC = pi - angA - angB
    ' calcular el lado b por ley de senos
b = A * Sin(angB) / Sin(angA)
    ' calcular el lado c por ley de senos
c = A * Sin(angC) / Sin(angA)
End Sub
```

## 1.1 El caso faltante: LLA o ALL

Este es el caso más complejo, ya que dos lados y un ángulo no entre ellos no determinan completamente un triángulo. Por ejemplo, si un triángulo tiene lados $a = 15$ y $b = 13$, y ángulo $B = \operatorname{arcsen} 0.8 \approx 53.13°$, entonces el tercer lado puede ser $c = 4$ ó $c = 14$:




La razón es que al buscar el ángulo $A$, la ley de senos da $\operatorname{sen} A = a \operatorname{sen} B/b = 15 \cdot 0.8/13 = 12/13$, pero existen dos ángulos posibles cuyo seno es $12/13$: $\operatorname{arcsen}(12/13) \approx 67.38°$ y $180° - \operatorname{arcsen}(12/13) \approx 112.62°$ (en el gráfico, este es el ángulo en el vértice superior).
Otra posibilidad en el caso LLA es que $\operatorname{sen} A = a \operatorname{sen} B/b > 1$, en cuyo caso no hay solución, o que $\operatorname{sen} A = 90°$, y la única solución sea un triángulo rectángulo. Pero también, aunque $\operatorname{sen} A < 1$ y aparentemente haya dos soluciones, podría ser que cualquiera de las dos haga $A + B + C > 180°$, por lo que no hay solución (por ejemplo, $a = 1$, $b = 1$ y $B = 150°$ dan $\operatorname{sen} A = 0.5 < 1$, pero $a = b$ implica que $A = B = 150°$, lo cual es imposible).
En general, para resolver el caso LLA los cálculos son así:

- $\dfrac{\operatorname{sen} A}{a} = \dfrac{\operatorname{sen} B}{b}$ de donde $A_1 = \operatorname{arcsen}\left(\frac{a \operatorname{sen} B}{b}\right)$ y $A_2 = 180° - A_1$, pero solamente si $a \operatorname{sen} B/b \leq 1$ (si es igual a 1 sólo hay una solución: $A_1 = A_2 = 90°$).

- $C_i = 180^\circ - B - A_i$ para $i = 1,2$, pero solamente si es positivo.
- $c_i = \dfrac{b \operatorname{sen} C_i}{\operatorname{sen} B}$ para $i = 1,2$.

La subrutina `casoLLA` recibirá tres parámetros de entrada: `a`, `b` y `angB`. También tendrá seis de salida: `angA1`, `angC1` y `c1` para la primera solución, y `angA2`, `angC2` y `c2` para la segunda. Como en las subrutinas anteriores, los resultados serán todos 0 si no hay solución. Si sólo hay una solución, los últimos tres resultados serán 0.

En esta subrutina la validación de datos se hará conforme avancen los cálculos.

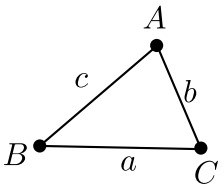
```
Public Sub casoLLA(ByVal a As Double, ByVal b As Double, ByVal angB As Double, _
        ByRef angA1 As Double, ByRef angC1 As Double, ByRef c1 As Double, _
        ByRef angA2 As Double, ByRef angC2 As Double, ByRef c2 As Double)
'   Resuelve el caso de lado-lado-ngulo conocidos.
'   Calcula el ngulo opuesto al primer lado por la ley de senos, el
'   otro ngulo por diferencia a pi y el lado faltante por ley de senos

Dim senA As Double  ' este valor se usa varias veces

    ' primera validacin: deben ser a, b y angB > 0
If a <= 0 Or b <= 0 Or angB <= 0 Then
    angA1 = 0: angC1 = 0: c1 = 0
    angA2 = 0: angC2 = 0: c2 = 0
    Exit Sub
End If

senA = a * Sin(angB) / b

    ' segunda validacin: debe ser senA <= 1
If senA > 1 Then
    angA1 = 0: angC1 = 0: c1 = 0
    angA2 = 0: angC2 = 0: c2 = 0

    ' si senA = 1 slo hay una solucin
ElseIf senA = 1 Then
    angA1 = pi / 2
    angC1 = pi / 2 - angB   ' C1 = pi - A1 - B
    c1 = Sin(angC1) * a     ' c1 = sen(C1)*a/senA
    angA2 = 0: angC2 = 0: c2 = 0    ' no hay segunda solucin

    ' si senA < 1 puede haber dos soluciones
Else
    angA1 = ArcSen(senA)
    angC1 = pi - angA1 - angB
    ' tercera validacin
    If angC1 <= 0 Then   ' no hay solucin
        angA1 = 0: angC1 = 0: c1 = 0
        angA2 = 0: angC2 = 0: c2 = 0
        Exit Sub
    End If
    c1 = Sin(angC1) * a / senA

    angA2 = pi - angA1
```

```
    angC2 = pi - angA2 - angB
    ' cuarta validacin
    If angC2 < 0 Then   ' no hay segunda solucin
        angA2 = 0: angC2 = 0: c2 = 0
        Exit Sub
    End If
    c2 = Sin(angC2) * a / senA
End If

End Sub
```

## 1.2 Representación gráfica de la solución

Las subrutinas que recién completamos serán el "motor" oculto del programa. El usuario verá dos formularios: uno en el que digitará los datos y otro en el que se mostrarán los resultados. En este segundo formulario se representará la solución gráficamente.

En esta columna vamos a desarrollar el formulario para la solución. Empezamos con repasar la notación que habíamos establecido en la columna anterior:




Esta notación no se respetó estrictamente en el módulo `Funciones.bas` que desarrollamos en esa columna, específicamente porque las posiciones de los lados y ángulos no eran relevantes (excepto que $A$ está opuesto a $a$, etc.). Pero en este formulario sí vamos a atenerlos a las posiciones específicas: El lado $a$ es el inferior, $b$ es el derecho y $c$ el izquierdo; de la misma manera, el ángulo $A$ es el superior, $B$ el inferior izquierdo y $C$ el inferior derecho.

A lo que no vamos a atenernos exactamente es a los nombres de las variables. Vamos a usar un arreglo `L(1 To 3)` para los lados $a$, $b$ y $c$, y un arreglo `A(1 To 3)` para los ángulos $A$, $B$ y $C$. Ellos estarán declarados en un módulo estándar llamado `Globales.pas` en el que de paso definiremos un símbolo para la constante $\pi/180$. El contenido de este módulo será, simplemente:

```
Option Explicit

Public Const pi_180 = 1.74532925199433E-02      ' pi/180

Public A(1 To 3) As Double     ' los ngulos:
                ' A(1) superior, A(2) izquierdo, A(3) derecho
Public L(1 To 3) As Double     ' los lados:
                ' L(1) inferior, L(2) derecho, L(3) izquierdo
```

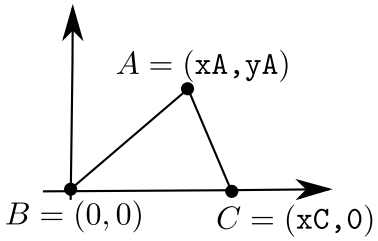
Ahora vamos al formulario `frmGrfico`, donde se graficará la solución. El diseño de este formulario es el siguiente:


En el formulario hay dos controles:

1. Un cuadro de dibujo (`PictureBox`) llamado `picTringulo`. Solamente necesitamos definir tres propiedades en este momento: `Left` $= 0$ y `Top` $= 0$ para ajustar el cuadro a la esquina superior izquierda del formulario, y `Font` $=$ `Arial` para poder mostrar el símbolo de grados. El tamaño del cuadro y la escala para el gráfico no son importantes ahora; éstos se definirán cada vez que el formulario cambie de tamaño.

2. Un botón de comando (`CommandButton`) llamado `cmdCerrar`. Éste tiene `Caption` $=$ "`&Cerrar`", `Default` $=$ `True` (para que se active al presionar la tecla `[Esc]`), y tamaño `Height` $= 495$, `Width` $=$ $1215$. Su posición se definirá también durante la ejecución.

Los tres puntos del triángulo se representarán con las coordenadas $A = (\texttt{xA}, \texttt{yA})$, $B = (\texttt{xB}, \texttt{yB})$ y $C = (\texttt{xC}, \texttt{yC})$. Pero vamos a establecer que $B$ siempre esté en el origen del sistema de coordenadas, así que $\texttt{xB} = \texttt{yB} = \texttt{yC} = 0$. Entonces tres de las coordenadas son constantes y las otras tres (`xA`, `yA` y `xC`) son variables.




Vamos a definir otras seis variables: `Izquierda` y `Derecha` serán las coordenadas de los bordes izquierdo y derecho del gráfico, y `Ancho` será la distancia entre ambas. Hay algo de redundancia aquí, porque obviamente `Ancho` $=$ `Derecha` $-$ `Izquierda`, pero así el programa es más fácil de leer. Además, no es cierto lo que dijimos que era obvio: en realidad `Ancho` podría tener que ajustarse para mantener las proporciones correctas entre las longitudes horizontal y vertical, como veremos abajo. Por ahora, las otras variables serán `Arriba` y `Abajo`, las coordenadas de los extremos superior e inferior, y `Alto`, la altura. La sección `(General)` completa es así:

```
Option Explicit
```

```
'   Este formulario grafica el tringulo a escala

'   El vrtice A (superior) tiene coordenadas (xA,yA);
'   el vrtice B (inferior izquierdo) es (0,0), y
'   el vrtice C (inferior derecho) es (xC,0)
Dim xA As Single, yA As Single, xC As Single
Const xB = 0: Const yB = 0: Const yC = 0

'   Dimensiones del cuadro de dibujo
Dim Izquierda As Single, Derecha As Single
Dim Arriba As Single, Abajo As Single
Dim Ancho As Single, Alto As Single
```

Lo más fácil de programar es el botón `cmdCerrar`, así que quitémoslo de en medio ya:

```
Private Sub cmdCerrar_Click()
'   Cerrar el formulario
Unload Me
End Sub
```

Ahora trabajemos en serio. Al cargar el formulario deben calcularse las tres coordenadas no constantes: $\texttt{xA} = c\cos B$, $\texttt{yA} = c\,\mathrm{sen}\,B$ y $\texttt{xC} = a$. Luego, la coordenada izquierda del gráfico será el mínimo entre `xA` y `xB` ($A$ podría estar a la izquierda de $B$); la derecha será el máximo entre `xA` y `xC` (o a la derecha de $C$). Por último, la coordenada superior será `yA` y la inferior `yB`:

```
Private Sub Form_Load()
'   Calcular coordenadas del grfico

xA = L(3) * Cos(A(2))           ' c cos(B)
yA = L(3) * Sin(A(2))           ' c sen(B)
xC = L(1)                       ' a

Izquierda = IIf(xA > xB, xB, xA) ' extremo izquierdo: min(xB,xA)
Derecha = IIf(xA < xC, xC, xA)  ' extremo derecho: max(xA,xC)
Arriba = yA                     ' extremo superior: siempre yA
Abajo = yB                      ' extremo inferior: siempre yB
End Sub
```

Eso tampoco fue difícil. El verdadero trabajo viene luego, al graficar el triángulo. Eso se hará cada vez que el formulario cambie de tamaño (en particular, al cargarse el formulario). La agenda es ésta:

1. Darle la posición correcta al botón `cmdCerrar`, en el extremo inferior derecho del formulario.
2. Darle el tamaño correcto al cuadro de dibujo, que llegue hasta el extremo derecho del formulario y hasta un poco antes del extremo inferior (dejando espacio para el botón).
3. Ajustar la escala del cuadro de dibujo para que quepa el triángulo completo. Esto implica dejar un margen sin usar, bien a la derecha o bien arriba del triángulo, para que la escala sea real (es decir, que los ángulos y las proporciones entre los lados sean correctos).
4. Indicar las longitudes de los lados.
5. Indicar las medidas de los ángulos.

Es un trabajo complicado, pero hay que hacerlo. La siguiente subrutina se encarga de todo:

```
Private Sub Form_Resize()
'   Cada vez que el formulario cambia de tamao (y al abrirlo)

cmdCerrar.Top = Height - 920    ' posicin del botn de Cerrar
cmdCerrar.Left = Width - 1400

Ancho = Derecha - Izquierda     ' ancho y alto del grfico
Alto = Arriba - Abajo

With picTringulo
    .Height = Height - 980      ' ajustar el cuadro de dibujo...
    .Width = Width - 100        ' al nuevo tamao de formulario

    ' Si una dimensin (horizontal o vertical) del tringulo
    ' no cabe, reducir la escala disminuyendo la otra dimensin
    If Ancho * .Height < Alto * .Width Then
        Ancho = Alto * .Width / .Height
    Else
        Alto = Ancho * .Height / .Width
    End If

    .Cls                        ' limpiar el cuadro
End With

    ' Definir la escala y dibujar los tres segmentos
picTringulo.ForeColor = vbBlack
picTringulo.Scale (Izquierda - Ancho / 10, Abajo + Alto + Alto / 10)- _
    (Izquierda + Ancho + Ancho / 10, Abajo - Alto / 10)
picTringulo.Line (xA, yA)-(xB, yB)
picTringulo.Line (xB, yB)-(xC, yC)
picTringulo.Line (xC, yC)-(xA, yA)

    ' Indicar las longitudes
picTringulo.ForeColor = vbBlue
EscribirLado (xB + xC) / 2, (yB + yC) / 2, 1
EscribirLado (xA + xC) / 2, (yA + yC) / 2, 2
EscribirLado (xA + xB) / 2, (yA + yB) / 2, 3

    ' Indicar los ngulos
picTringulo.ForeColor = vbRed
Centrarngulo xA, yA - AltoRengln, 1
Centrarngulo xB, yB, 2
Centrarngulo xC, yC, 3
End Sub
```

Bueno, casi todo. Falta definir cómo se indican los lados (con `EscribirLado`) y los ángulos (con `Centrarngulo`). Pero primero, la instrucción que calcula la escala del cuadro merece algo de comentario:

```
picTringulo.Scale (Izquierda - Ancho / 10, Abajo + Alto + Alto / 10)- _
    (Izquierda + Ancho + Ancho / 10, Abajo - Alto / 10)
```

El método `Scale (x1,y1)-(x2,y2)` define las coordenadas de las esquina superior izquierda como `(x1,y1)` y de la superior derecha como `(x2,y2)`. Aquí estamos definiendo que la esquina superior izquierda del cuadro tendrá coordenada $x$ igual a `Izquierda - Ancho / 10`, que es la coordenada izquierda del triángulo menos un 10% del ancho (para rodear el triángulo por un margen que dé espacio para anotaciones). La coordenada $y$ de esa esquina es `Abajo + Alto + Alto / 10`: la coordenada inferior más la altura, más un 10% de la altura como margen. Por otra parte, la coordenada inferior derecha tiene coordenada $x$ igual a `Izquierda + Ancho + Ancho / 10` y coordenada $y$ igual a `Abajo - Alto / 10`, con explicaciones semejantes.

Ahora bien, la subrutina `EscribirLado` recibe un par de coordenadas $(x,y)$ y un índice $i$, y escribe el valor de `L(`$i$`)` a partir de la posición dada. Como vemos en el evento `Form_Resize`, cada longitud se escribe a partir del punto medio del lado respectivo.

```
Private Sub EscribirLado(x As Single, y As Single, i As Byte)
'   Escribir L(i) en la posicin (x,y) de picTringulo

picTringulo.CurrentX = x
picTringulo.CurrentY = y
picTringulo.Print ValorAHilera(L(i));
End Sub
```

(Luego veremos qué es `ValorAHilera`.) Los ángulos, en cambio se escriben centrados horizontalmente en el vértice, por encima en el caso de $A$ y por debajo en el caso de $B$ y $C$. La subrutina `Centrarngulo` recibe una posición $(x,y)$ y un índice $i$, y escribe el valor de `A(i)` centrado en la posición indicada.

Hay varios detalles que cuidar aquí: Al escribir en una posición $(x,y)$, el texto aparecerá hacia la derecha y abajo de $(x,y)$, lo cual era apropiado para los lados. Pero aquí queremos dos variaciones: que el valor de $A$ se escriba encima del vértice, y que cada medida se centre en la posición dada. Para lo primero debe conocerse la altura de un renglón; para lo segundo se necesita el ancho de la hilera por escribir. Como estos valores dependen del tamaño y tipo de letra, no podemos programarlos como constantes sino que deberán calcularse durante la ejecución.

Para calcular la altura de un renglón usamos la función siguiente. Ella pone el "cursor" en la línea 0, escribe una línea en blanco y retorna la nueva posición vertical del cursor. Esta función ya había sido usada en la instrucción `Centrarngulo xA, yA - AltoRengln, 1`, que escribe el valor de $A$ en la posición con coordenada $x$ igual a `xA` y coordenada $y$ igual a `yA` desplazada una altura `AltoRengln` hacia arriba.

```
Private Function AltoRengln() As Single
'   Calcula la altura de un rengln al escribir
picTringulo.CurrentY = 0
picTringulo.Print
AltoRengln = picTringulo.CurrentY
End Function
```

Para calcular el ancho de la hilera por escribir, el truco se semejantemente sucio: Se escribe la hilera y se toma nota del cambio en la posición horizontal del cursor. Pero ahora hay que escribir la hilera realmente; no basta con escribir un renglón vacío como antes. Para no ensuciar el gráfico con un ensayo de la hilera, la solución es escribirla fuera del rango visible del cuadro, como en la posición `-yA` (muy por debajo del borde inferior). En fin, la subrutina es así:

```
Private Sub Centrarngulo(x As Single, y As Single, i As Byte)
'   Centrar A(i) en la posicin (x,y) de picTringulo
```

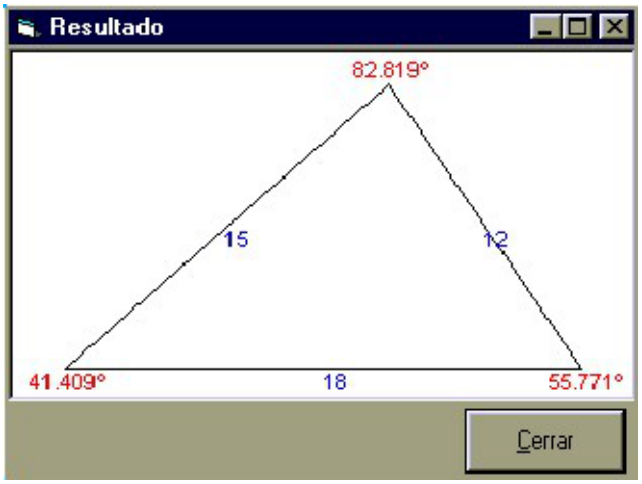
```
Dim txt As String, AnchoHilera As Single
txt = ValorAHilera(A(i) / pi_180)   ' convertir a grados

    ' medir el ancho de la hilera al escribirla
With picTringulo
    .CurrentX = 0: .CurrentY = -yA  ' fuera de la vista
    picTringulo.Print txt;
    AnchoHilera = .CurrentX         ' cunto midi la hilera
    .CurrentX = x - AnchoHilera / 2: .CurrentY = y  ' centrar
    picTringulo.Print txt; "o"     ' escribir
End With
End Sub
```

El símbolo `"°"` en la última instrucción, `picTringulo.Print txt; "°"`, parece una letra "o". En realidad es el símbolo de grados, y puede digitarse con la combinación de teclas `[Alt+0186]`. Para que se muestre correctamente en el cuadro de gráfico, recuérdese darle al cuadro la propiedad `Font = Arial`.

El último detalle pendiente es la función `ValorAHilera`. Ésta recibe un valor real $x$ y retorna una hilera que lo aproxima con cinco dígitos significativos. Aquí no sirve la función `Round(x,d)` de Visual Basic, porque ésta redondea $x$ a $d$ decimales. Como no conocemos la magnitud de los valores preferimos redondear a cinco dígitos significativos en vez de redondear a un número fijo de decimales. Pero acabamos de mentir otra vez: la función `ValorAHilera` en realidad no redondea sino que trunca. La manera de hacerlo es directa: el valor de $x$ se convierte en hilera con la función `Str`; del resultado se ignora el primer carácter, que con seguridad es un espacio, y se toman los siguientes seis caracteres (si $x$ tuviera seis o más digitos antes del punto decimal, el resultado tendrá seis, no cinco dígitos significativos, porque el punto no estará ocupando un lugar).

```
Private Function ValorAHilera(Valor As Double) As String
'   Convierte Valor en una hilera
ValorAHilera = Mid(Str(Valor), 2, 6)
End Function
```

Como dijimos, era un trabajo difícil. Pero ya terminamos, por ahora. Aquí tenemos una muestra del funcionamiento de este formulario, suponiendo que de alguna forma definimos las variables globales `L(1 To 3)` y `A(1 To 3)` con los valores indicados.




Parte importante de la belleza del resultado es que al cambiar el tamaño del formulario el gráfico se redibujará aprovechando al máximo el espacio disponible, pero siempre dejando márgenes de al menos 10% alrededor

del triángulo y respetando los verdaderos ángulos y las proporciones entre las longitudes.

El siguiente paso será diseñar el formulario principal: Leer los tres datos del usuario, llamar la función apropiada en el módulo `Funciones.bas` y pasar la solución a este formulario que acabamos de diseñar. Ése será el objetivo de la siguiente columna.