Curso breve de VBA orientado a la Climatología Aplicada

(Visual Basic para Aplicaciones Microsoft)

Roberto Rodríguez Barrera

Programa de la asignatura

1. Introducción
    a. Excel y las macros
        i. Importación de datos
        ii. Grabación de macros
    b. Conceptos básicos de VBA
        i. Clase. Instancia. Objeto.
        ii. Propiedades.
        iii. Métodos.
        iv. Eventos.
    c. El entorno de programación VBA
        i. Ventana de proyectos.
        ii. Ventana de propiedades.
        iii. Ventana de diseño.
        iv. Ventana de código.
    d. Conexión entre VBA y las hojas de cálculo
2. Variables
    a. Declaración de variables.
        i. Privadas
        ii. Públicas
    b. Tipos de variables de mayor utilidad
        i. String
        ii. Integer
        iii. Single
        iv. Double
        v. Variant
    c. Conversión a diferentes tipos: Cstr() CInt() CSng CDbl
    d. MsgBox y su variables
    e. Conversión Cstr(), CInt(), CSng(), Cdbl()
    f. Vectores y matrices. Declaración
3. Estructuras del VBA
    a. Módulos y procedimientos
    b. Procedimientos y funciones
    c. Etiquetas
    d. Estructuras de control
        i. For next. While wend Do Loop
        ii. If then else end if
        iii. DoEvents
    e. Objetos especiales
        i. Sheets(“”)
        ii. Cells(x,y)

Cuando nos disponemos a realizar el tratamiento de una serie de datos, nos quedamos normalmente en manos del paquete estadístico con el que trabajemos habitualmente, sin preocuparnos, en principio, de qué es lo que pasa con la información que le pasamos y como la trabaja. Además, en muchas ocasiones, los inputs tienen formatos poco amigables con nuestras apps y la exportación de resultados debe ser también realizada de acuerdo con ellos.

Visual Basic para Aplicaciones Microsoft (VBA®) constituye una herramienta eficaz de series numéricas ya que permite controlar los datos, su lectura, su tratamiento estadístico y la exportación de datos y resultados para otras aplicaciones.

Para ello disponemos de un conjunto de órdenes básicas que nos permiten:

a. Leer (importar) datos de archivos tipo texto
b. Cambiar formatos de archivos de datos
c. Realizar operaciones matemáticas (estadísticas)
d. Presentar resultados
e. Exportar resultados

La ventaja es que el entorno de programación lo tenemos ligado a Microsoft® Office®, el paquete ofimático más popular.

Con unos pocos conocimientos de programación, posteriormente adaptable a todos los lenguajes, podremos realizar aplicaciones que nos ayuden en nuestros análisis de datos.

De entre todas las aplicaciones Office, trabajaremos sobre Excel®, aunque todas y cada una de las aplicaciones llevan incorporado este entorno de programación.

Macros

Una macro es un conjunto de órdenes secuenciales. Las macros pueden ser grabadas o escritas.

- Las grabadas no son más que la transcripción en órdenes VBA de las acciones realizadas en un libro Excel.
- Las escritas corresponden a la secuencia de órdenes en tiempo de programación.

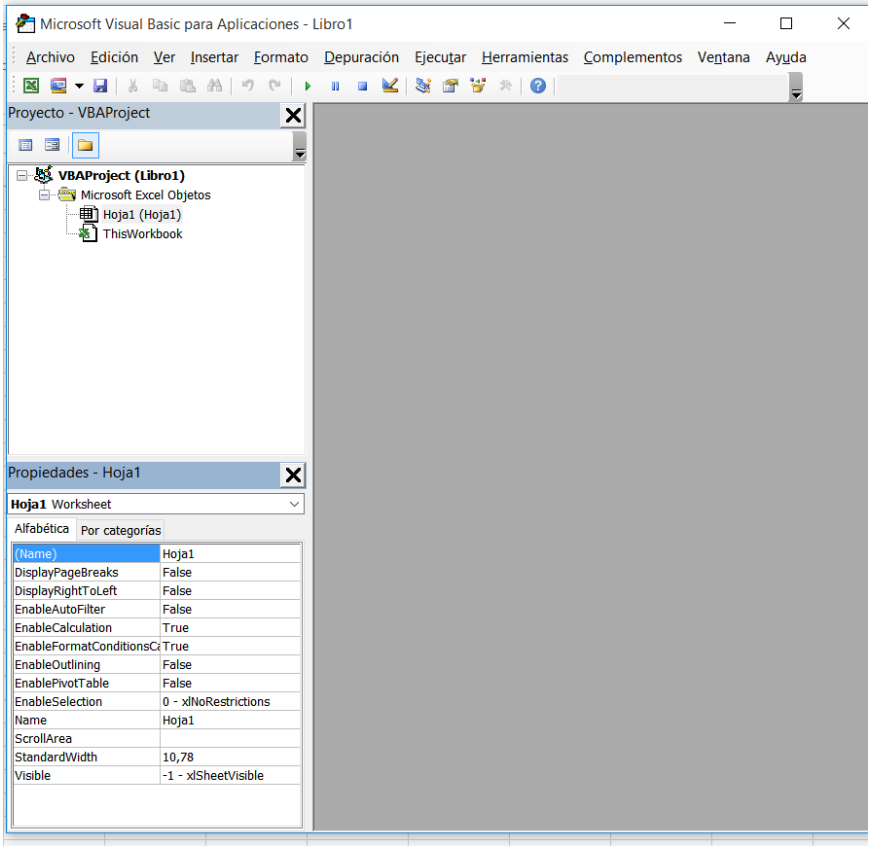
Las macros grabadas pueden ayudarnos a conocer el código necesario para realizar alguna acción determinada.

Las macros se graban como módulos en la ventana de proyectos. La sintaxis es compleja pero puede ayudarnos a realizar acciones que sin ella sería muy costoso (en tiempo de instrucción) realizarlas. A lo largo de este breve curso, realizaremos grabación de macros

Entorno de programación

Si sobre una hoja Excel® pulsamos ALT+F11 se abrirá ante nosotros el entorno de programación VBA. Éste consta principalmente de:

- Barra de menú, horizontal superior.
- Barra de menú gráficos (iconos), horizontal
- Ventana de proyectos, vertical izquierda superior
- Ventana de propiedades, vertical izquierda inferior
- Ventana de diseño donde encontramos al formulario
- Ventana de controles o herramientas
- Ventana de código, a la que se accede haciendo doble click sobre cualquier objeto de la ventana central



En programación hay una serie de conceptos básicos que se deben tener en cuenta para poder entender el lenguaje.

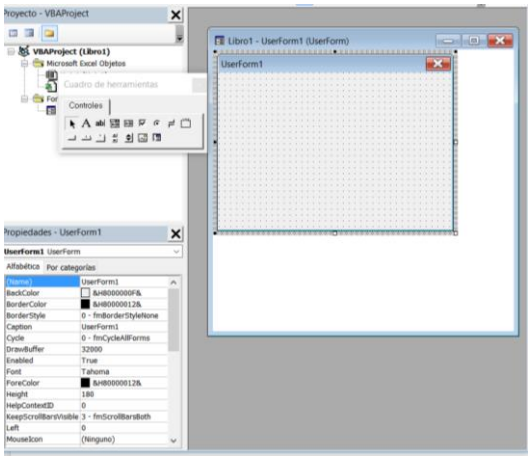
a. Objeto, elemento central del entorno. En sí es un conjunto de “ceros y unos” almacenados en memoria que tendrán características y acciones y sobre los que recaerán sucesos (tipo click, doble click, tecla pulsada, botón izquierdo del ratón pulsado, etc.).
b. Clase, matriz de objetos. Elementos de dónde extraemos los objetos. Por ejemplo, la clase de los botones, de ahí salen los botones que utilizaremos en nuestros programas. Están en la ventana de herramientas o controles y cuando pulsamos allí sobre una clase se dice, en el argot de la programación, que estamos realizando una instancia de la clase button, o sea estamos creando un botón.
c. Propiedades, características (adjetivas) de un objeto. Color, texto, tamaño, estado (activo o no), nombre, etiqueta, etc.
d. Métodos, acciones que realiza (o puede realizar) un objeto.
e. Evento, es una acción que realiza el usuario de la aplicación o suceso a la que el programa debe dar respuesta (tipo click, doble click, tecla pulsada, botón izquierdo del ratón pulsado, etc.).

Visual Basic es un lenguaje de programación orientado a eventos, esto quiere decir que nuestro objetivo como programadores son las respuestas que nuestro programa dé a los eventos que sobre los objetos acontezcan.

El objeto que, a su vez, contiene a todos los objetos, es el Userform. En él colocaremos al resto de objetos imprescindibles para nuestra aplicación. Una aplicación puede tener varios Userforms, aunque por el momento podemos considerar una aplicación, un Userform.

Si en la barra de menús clickamos sobre insertar, e insertamos Userfom nos aparecerá una ventana central gris que es eso, un Userform (en este caso de nombre Userform1, ya que es la primera instacia de la clase Userform que hemos realizado para nuestra aplicación), sobre el que añadiremos el resto de objetos de nuestra app o proyecto.

Básicamente el resto de objetos más comunes que trabajaremos en esta aproximaxión al lenguaje serán las cajas de texto (Textbox), las etiquetas (Label) y los botones (Commandbutton),



Ejercicio. Hola Mundo

Al entrar en cada lenguaje existe una primera práctica común a todos por tradición. Ésta es el *Hola Mundo* (Hola Mon, Hello World, etc.) que se suele realizar sobre un cuadro de texto, una etiqueta o un mensaje. Nosotros la podemos realizar sobre un mensaje de pantalla donode, además del texto, podremos mostrar un icono descriptivo.

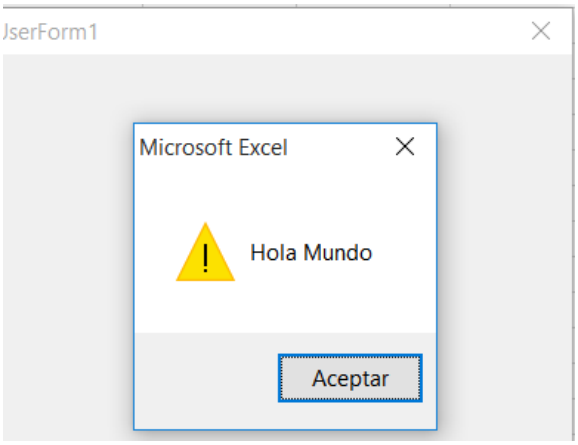
Así pues, para programar el *Hola Mundo*, haremos doble click sobre un Userform previamente insertado. Esto nos abre una ventana que aún no habíamos visitado, la ventana de código. Esta ventana tiene, en la parte superior dos cajas de texto. En la de la izquierda nos muestra el objeto sobre el que escribiremos el código, en este caso Userform, y en la de la derecha el evento al que el código responde, por defecto click. En la ventana central veremos escrito Private Sub Userform_Click y End Sub, lo que indica que, las líneas de código que vamos a escribir responden al evento click realizado sobre el objeto Userform.

Así, vamos pues a dar respuesta a la acción Click sobre el formulario. El objeto mensaje es el objeto MsgBox, que NO está en la ventana de herramientas o controles, por lo que deberemos invocar su presencia manualmente. Así nos situamos en una línea intermedia entre Private Sub Userform1_Click y End Sub y escribiremos la siguiente línea:

Mensaje=MsgBox (“Hola Mundo”,vbExclamation)

Donde mensaje no es más que un lugar de RAM donde se guardará un valor (relacionado con el botón accptar del mensaje) y vbExclamación hace referencia al icono del tipo de mensaje mostrado.

Para ejecutar la aplicación podemos recurrir a F5, al menú Ejecutar o al icono triangular Play. Para para es aspa superior derecha del form en ejecución, o, en la ventana de programación al cuadrado de stop, eso sí, después de clickar sobre Aceptar.

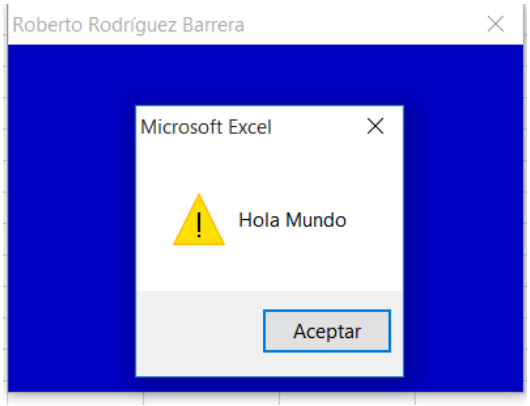


Este *Hola Mundo* nos servirá para ver las propiedades más comunes de los objetos.

Si vamos a la ventana de diseño, clickando en la ventana de proyectos sobre el objeto Userform1, veremos que, en la ventana de propiedades aparece, como propiedad name Userform1. Eso nos indica que estamos en las propiedades del objeto Userform.

- La propiedad name es una propiedad que NO se debe cambiar a la ligera. Indica el nombre a la que la aplicación se dirigirá internamente para mostrar el Userform.
- La propiedad Caption es la etiqueta que se verá en la barra de nombre en programación y ejecución del objeto, en este caso en lugar de Userform1, podemos poner nuestro nombre.
- Backcolor, es la propiedad que determina el color de fondo del objeto. Si escogemos la lengüeta superior de paleta, se nos mostrarán los colores y podremos escoger el que más nos guste.

EL resto de propiedades ya las iremos descubriendo a medida que nos internemos en el lenguaje y utilizando más objetos. Recordad que cada objeto tiene sus propiedades, métodos y eventos.



Variables

Una variable es en realidad, una zona de la memoria que guarda información y que se designa por una dirección binaria o hexadecimal, simbolizada por uno o más caracteres que

utilizaremos siempre para dirigirnos a ella (a su valor en ese momento más concretamente). Se llama variable porque a lo largo de la aplicación variará seguramente su valor en función de las necesidades. Si no debe variar su valor se la conoce como constante.

En Visual Basic hay diferentes tipos de variables pero aquí tan solo hablaremos de los más utilizados.

- De cadena o string (Str)
- De enteros o Integer (Int)
- De reales simples o Singles (Sng)
- De reales largos o Doubles (Dbl)
- Variantes, depende de la asignación o valor que contengan en cada momento (Variant)
- Vectores o Matrices (Arrays) a(x), a(x,y), a(x,y,z), …

Las variables, a su vez, pueden ser públicas (Public) o privadas (Private). Una variable pública es aquella que se puede utilizar con un mismo valor en cualquier parte de la aplicación, esto es, en cualquier respuesta a cualquier evento surgido sobre cualquier objeto. Las privadas solo mantienen su valor en la respuesta al evento surgido en un solo objeto, estando no definidas o con otro valor para el resto.

Las variables públicas se deben declarar (decir que las vamos a utilizar y cómo son) al principio de la aplicación, por ejemplo en el evento Activate del Userform, o también en un módulo-declaraciones (como ya veremos posteriormente).

La sentencia de definición es la siguiente:

```
Public nombrevariable As String

Private b As Integer
```

Si una variable a, por ejemplo Integer, la necesitamos convertir en cadena utilizaremos b = CStr(a). Así si a = 4 b será la cadena “4”. Estás funciones de conversión de tipos de variable son:

```
CInt(x), CSng(x), CDbl(x), CStr(x)
```

Sobre todo debemos tener en cuenta que la mezcla entre variables de diferente tipo no está permitida. Si queremos mostrar un mensaje

```
mensaje=MsgBox(“La temperatura media es de “+a+” ºC”,vbInformation)
```

si a es una variable Integer , el programa dará un error de Type Mismatch, lo que indica que debemos convertir a a en una String para poder agruparla con “La temperatura media es de “ y con “ ºC”. Para ello en lugar de a escribiremos Cstr(a), quedando la sentencia:

```
mensaje=MsgBox(“La temperatura media es de “+Cstr(a)+” ºC”,vbInformation)
```

De todos modos, excepto los vectores y matrices, si no declaramos las variables quedan automáticamente definidas como Variant. Así, si en cierto lugar de la App aparece t=5.02, t se habrá autodefinido como variable local tipo single. Si después, en otro lugar aparece t=”Hola”,

t se habrá redefinido como una variable local de string (Las comillas se deben a que todos los valores string deben de llevar comillas para designar su valor).

Los vectores y matrices se deben declarar igual que las variables públicas, en Userform – Activate o en un Modulo-Declaration

Inserción de objetos

Ha llegado el momento de utilizar los objetos más comunes de VBA. Así insertaremos una etiqueta en el formulario, que en la ventana de herramientas viene dada por una A. Así arrastraremos sobre el objeto Userform1. Por defecto aparecerá el caption que corresponde al nombre de la clase seguida del número de instancia que se ha hecho de esa clase, en este caso *Label1*.

Los objetos pueden modificar sus propiedades de dos formas:

- En tiempo de programación
- En tiempo de ejecución (RunTime)

Así, si queremos modificar una propiedad en tiempo de programación no tenemos más que con click seleccionar el objeto e ir a la ventana de propiedades y cambiarla. Si, por el contrario, queremos modificar una propiedad en tiempo de ejecución, deberemos escribir una sentencia donde identifiquemos al objeto y su propiedad y le asignemos un determinado valor:

Userform1.Label1.Caption=”Label modificada”

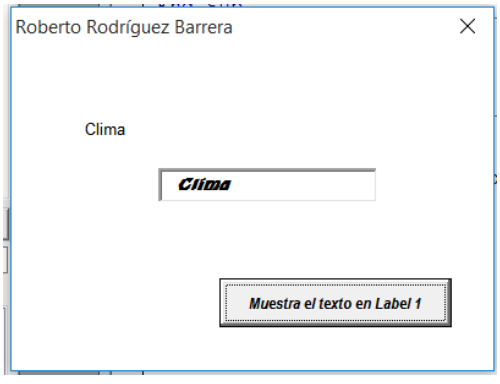
(Si sólo tenemos un Userform no hace falta encabezar con el nombre del formulario al objeto)

Si lo que queremos es asignar en tiempo de ejecución una propiedad cuyo valor ha sido dado en tiempo de ejecución, deberíamos, por ejemplo, escribir la sentencia

Userform1.Label1.Caption=Userform1.Textbox1.Text

Donde Texxtbox1 es un objeto, instancia de la clase Textbox (caja de texto) designada en la barra de herramientas o controles con ab| y Text la propiedad que da el contenido de esa caja.

Pero este conjunto de acciones debería realizarse cuando el usuario, voluntariamente quisiera. Por tanto, una opción podría ser realizar una instancia de la clase CommandButton para crear un objeto, CommandButton1, y cuando suceda el evento click que la etiqueta Label1 tenga como Caption el texto que, previamente, hayamos escrito en la propiedad text del objeto Texbox1.



Estructuras de Control

a. Bucles

Para realizar una acción repetidamente, con un principio y un fin, los conocidos contadores se utilizan

```
For i = x to y Step z
…
Next
```

Donde i es el contador, x, y z son el valor inicial, el final y la cadencia respectivamente

O también el

```
While i <= x
i=i+z
…
Wend
```

Donde i es el contador, z la cadencia

O también

```
Do Until i<6
…
I=i+z
Loop
```

Donde i es el contador y z la cadencia.

b. Condicionales

Para establecer condiciones utilizamos la estructura

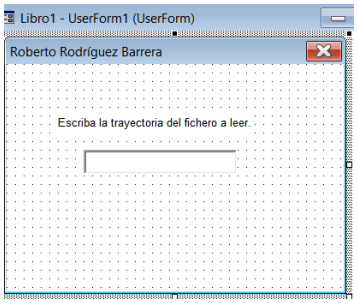
```
If condición then
Si sí
else
si no
End if
```

donde *condición* es una comparativa, por ejemplo a<=5, b=”hola”, etc.; si *sí* es el conjunto de acciones si se cumple la condición, y si *no* es el conjunto de acciones en caso de no cumplirse la condición.

Lectura y escritura en archivos de texto

Para poder trabajar con datos el primer paso es la lectura de ellos desde archivos grabados desde diferentes tipos de soporte y, por tanto, con formatos diversos. Aquí aprenderemos a leer y grabar archivos de datos con el formato de texto estándar, por columnas, y también con alguna variante.

Para leer un archivo de texto, debemos saber en primer lugar la trayectoria que tiene, esto es, si lo tenemos en la carpeta de ejecución del programa o bien en otra distinta. Tanto el nombre del archivo de datos o la trayectoria la podemos ubicar en una instancia de la clase *TextBox*. Para que recordemos dónde debemos situar el archivo o trayectoria, podemos extraer una etiqueta, objeto *Label1*, o sea una instancia de la clase *Label*.



Cuando se abre un archivo (open) debemos especificar si va a ser para leerlo (input), grabarlo (output) o para añadir datos a uno que ya tenía (append). Como quiera que podríamos tener más de uno abierto, deberemos indicar un número o canal de archivo o handle (#n). Así la sentencia de apertura para leer el archivo de datos *temperaturas.dat* sería:

```
Open c:\trabajo\temperaturas.dat For Input As #1
```

o

```
Open Textbox1.text For Input As #1 (si cogemos la trayectoria de la caja de texto).
```

Los archivos de texto estándar o por columnas, constan de filas y columnas de datos, como una hoja Excel® pero sin plantilla:

```
1980    1       10.5
1980    2       11.3
```

Para leerlos debemos leer todas las columnas de cada fila por este orden. Entonces, la mejor opción es hacer un bucle para las filas y en cada fila una lectura de cada columna en variables de tipo vector (array).
Las variables tipo vector se declaran en el mismo lugar que las variables públicas con, por ejemplo

```
Dim v(2,3)
```

Aquí se ha declarado el vector v de 3 posiciones, desde v(0,0) a v(1,2), siendo cada posición una variable en sí misma pero del mismo tipo (string, Int, …)
Los vectores de distinguen por su dimensión. El que hemos utilizado como ejemplo tiene dimensión 2. Los hay de dimensión 1, v(i), y de dimensión n, v(i,j,k,…,m,n). Evidentemente debemos tener cuidado con no definir muchos vectores con dimensión alta porque acaba con los recursos RAM del sistema.

Por tanto, para realizar la lectura podríamos escribir:

```
For i = 0 To 1
    Input #1 v(i,0), v(i,1), v(i,2)
Next
```

Así podríamos dirigirnos posteriormente a cada posición por fila, columna. La v(0,3) será 10.5 y la v(1,1) será 1980
Siempre, una vez leído el archivo se debe cerrar con la orden:

```
Close #1
```

Para grabar un archivo, el proceso es muy similar

```
Open c:\trabajo\temperaturas.dat For Output As #1
o
Open Textbox1.text For Output  As #1 (si cogemos la trayectoria de la caja de texto).
```

Debemos tener cuidado porque esta apertura destruye cualquier otro archivo que se llamara igual. Si ya existe y queremos ampliar el contenido, en lugar del modo Output por Append, quedando de esta manera

```
Open c:\trabajo\temperaturas.dat For Append As #1
o
Open Textbox1.text For Append  As #1 (si cogemos la trayectoria de la caja de texto).
```

Para realizar la grabación de los datos, en lugar de Input que era para lectura, utilizaremos Write #X

Así, siguiendo la estructura ya utilizada en la lectura, podríamos escribir:

```
For i = 0 To 1
   Write #1 v(i,0), v(i,1), v(i,2)
Next

Close #1
```

Uno de los aspectos con los que se debe tener más cuidado en el manejo de ficheros es el fin de fichero, sobre todo en la lectura. Para curarnos en salud y no leer cuando ya no hay más datos en el fichero es trabajar con la sentencia eof (end of file). Podemos incluir la condicional *Si hemos llegado al final del fichero que no lea más*.

```
For i = 0 To n
   If eof (1)  then
         i= n
   Else
        Input #1 v(i,0), v(i,1), v(i,2)
   End if
Next
```

<u>Conexión con la hoja de cálculo</u>

De una manera similar a los vectores, podemos posicionar en la hoja de cálculo los datos leídos en un archivo. La orden para situar datos en la hoja de cálculo es:

```
Hoja1.Cells(i,j) = v(i,j)
```

Así el código

```
For i = 0 To 1
        Input #1 v(i,0), v(i,1), v(i,2)
        For j=0 To 2
            Sheet("Hoja1").Cells(i+1,j+1) = v(i,j)
        Next
Next
```

Colocará el vector v(i,j) en la Hoja1

Ésta sitúa el valor contenido en la posición i,j del vector v en la celda de fila i y columna j

El +1 se debe a que las posiciones de las celdas empiezan en 1 y no en 0 como los vectores.

Si quisiéramos ver, por ejemplo, la evolución de los contadores por pantalla, además de ponerlos en etiquetas o cajas de texto, deberíamos incluir la sentencia DoEvents que realiza un refresco de pantalla actualizando los contadores. Como es obvio, esta acción retarda el final de la aplicación.

Operaciones y funciones

a. Operaciones aritméticas. Con VBA podemos realizar la inmensa mayoría de operaciones y funciones matemáticas. Simplemente debemos indicar la variable donde se guardará el resultado y la variable que contiene el/los operadores.

   Así, pasa sumar, restar, multiplicar, dividir y elevar los comandos son los habituales:

   +, -, *, /, ^x, ^(1/x)

   Las funciones trascendentes tampoco guardan secreto alguno:

   Log(x) (neperiano), Log(x)/Log(base), sin(x), cos(x), tan(x) (x en radianes),

   asin(x), acos(x), (resultado en radianes y -1<=x<=1)

   (atan(x) se debe hacer en la hoja de cálculo)

b. Reasignaciones de variables. Una variable, como por ejemplo t = 5 pueden reasignarse a otro valor que dependa del que tenían hasta ese momento.

   ```
   t = t + 3
   ```

   Matemáticamente es inconsistente pero en programación se suele utilizar con mucha frecuencia. La variable t valía 5 pero al pasar por esa línea de código pasa a valer el valor que tenía más 3, total que t pasará a valer 8.

c. Funciones de string. Las operaciones más habituales de cadena son:
   - Concatenación Si a="Hola " y b="adiós" entonces a + "y " + b será Hola y adiós
   - Longitud string: Si c = len(a), entonces b valdrá 4
   - Subcadena por la izquierda de n caracteres: Si c = left(a,3) entonces c será Hol

- Subcadena por la derecha de n caracteres: Si c = right(a,3) entonces c será ola
- Posición de una subcadena dentro de una cadena: c = inStr(a,"ol") entonces c será 2.

d. Booleanas. Utilizadas en estructuras condicionales como If Then Else End if.
   - And: If a<10 And a>5 Then…. donde se cumplirá la condición si a es mayor que 5 y menor que 10
   - Or: If a<10 or a>15 Then…. donde se cumplirá la condición si a es mayor que 5 o menor que 10, pero no si a =>10 y a<=15.

Ejercicio:

Del archivo de temperaturas medias mensuales de un determinado año, codificado.dat, extraigan, en la Hoja1, columna A el año, columna B el mes y columna C el valor de la temperatura media en ese mes.

Ejercicio

Rutina

Transposición. Es habitual que, para análisis de datos, nos interese, sobre todo en climatología, pasar de datos en columnas a datos de fila, por ejemplo de archivos *año, mes, dato* a *año, dato de enero, ……, dato de diciembre* y viceversa.

```
Private Sub CommandButton1_Click()
Dim a(3000, 3)
' lectura de las tres primeras columnas del archivo
Open "bellol.dat" For Input As #1
For i = 1 To 3000
        If EOF(1) Then
                tope = i - 1
                i = 3000
        Else
                Input #1, a(i, 0), a(i, 1), a(i, 2), g, h
        End If
Next
Close #1
Men = MsgBox("Número de filas"+CStr(tope), vbInformation)

' escritura en 13 columnas del archivo
Open "Bello.mes" For Output As #1
For i = 1 To tope Step 12
        Write #1, a(i, 0), a(i, 2), a(i + 1, 2), a(i + 2, 2), a(i + 3, 2), a(i + 4,
2), a(i + 5, 2), a(i + 6, 2), a(i + 7, 2), a(i + 8, 2), a(i + 9, 2), a(i + 10, 2), a(i +
11, 2)
Next
Close #1
End Sub
```

Aquí hemos declarado el vector en un procedimiento o subrutina local el archivo a leer está en la misma carpetas que el libro Excel®. El archivo de salida se grabará en la misma carpeta.

Ejercicio III: Transponer

a. Leer el archivo *.dat y volcarlo sobre una hoja de cálculo.
b. Transponer en ficheros de 13 columnas en otra hoja.
c. Grabarlos en un archivo *.mes

d. Leer un archivo *.mes y volcarlo sobre una hoja.
e. Transponer en ficheros de 3 columnas, año, mes y dato en otra hoja.
f. Grabarlos en un archivo *.dat

Ejercicio IV: Cálculos sobre archivos climáticos

- El archivo de tipo secuencial ic.dat tiene los siguientes campos: año, fracción mensual anual, temperatura media, contador, mes. Empieza en enero de 1657 y acaba en diciembre de 1977. Sobre él debemos realizar las siguientes acciones:

  a. Hacer una lectura del archivo.
  b. Leerlo sobre una matriz
  c. Calcular el valor medio mensual
  d. Calcular el valor medio anual
  e. Calcular la desviación estándar anual
  f. Mostrar los resultados en VBA
  g. Volcar datos y resultados en una hoja Excel
  h. Crear el archivo mensual ic.mes donde figuren los campos año, enero, febrero, …, diciembre

- Del archivo de tipo mensual barp.mes, presión media mensual de Barcelona (dPa), de enero de 1780 hasta diciembre de 1989. con los campos año, enero, …, diciembre, debemos realizar las siguientes acciones:

  a. Hacer la lectura del archivo
  b. Volcarlo sobre una matriz
  c. Calcular el valor medio mensual
  d. Calcular el valor medio anual
  e. Calcular la desviación estándar anual
  f. Mostrar los resultados en un mensaje por pantalla
  g. Volcar datos y resultados en una hoja Excel