Comencemos a programar con

VBA - Access

Entrega 03

Primeros conceptos

Eduardo Olaz

Primeros conceptos

Ya hemos visto que en la ventana Inmediato podemos efectuar cálculos, con valores directos o con unos elementos a los que les asignamos valores.

Todo esto es interesante, pero todavía no podemos hacer grandes cosas, de hecho con una simple calculadora seríamos más eficientes.

Un poco de paciencia; todo a su tiempo…

Si no tenemos abierto un fichero Access, es el momento para hacerlo, por ejemplo crea un nuevo fichero con el nombre Entrega03.mdb.

Abrimos el apartado [Módulos] y presionamos el botón [Nuevo].

Pulsamos con el cursor en la ventana de Código y escribimos lo siguiente:

```
Option Compare Database
Option Explicit


Const Pi As Double = 3.14159265358979
```

¿Qué significa esta última línea?

A falta de explicar otros detalles, declaramos una Constante llamada Pi del tipo Double (un valor numérico de coma flotante de 8 bytes) y le asignamos el valor del famoso Pi.

Os recuerdo que `Pi` es algo más que el simple `3,1416` que nos enseñaron en el colegio…

Al declarar `Pi` como Constante nos impide que en otro punto del módulo podamos hacer algo así como

```
Pi = 25.36
```

-¿Por qué?-

-Porque el valor de una constante es en definitiva Constante, y no se puede cambiar-.

-¿Y cómo podemos declarar algo que se pueda cambiar?-

-Ese algo, en contraposición a una constante, se llama Variable.

Para declarar variables se podría hacer algo así como

```
Dim Precio As Currency
Dim Nombre As String
```

Enseguida veremos qué es eso de `Currency` y `String`.

Ahora vamos a intentar usar el valor que hemos declarado de `Pi`.

Pulsamos en la ventana Inmediato y escribimos

`? Pi` y presionamos [Enter]

¡Y no pasa nada!

Se suponía que seríamos muy felices si nos imprimiera el valor `3,14159265358979`

¿Por qué no lo ha hecho?

Ámbito ó Alcance de las Constantes y variables

Antes que nada aclarar que se suelen utilizar indistintamente ambos términos

¿Qué entendemos por ámbito o por Alcance?

Algo tan simple, y a la vez tan complejo, como los lugares en los que VBA puede "Ver", a una variable ó constante lo que equivale a que se entera que existen, las constantes y variables que declaramos. Más adelante veremos que todo esto aún se puede generalizar para otros elementos de la programación.

En el punto anterior hemos escrito **`? Pi`** en la ventana Inmediato y no ha pasado nada.

Al declarar en la cabecera de un módulo una variable con `Dim`, o una constante con `Const` hacemos que esa variable, y esa constante sólo sean visibles desde dentro del código del módulo.

Para que puedan verse desde fuera es necesario ponerles la palabra **`Public`**.

Esto se hace de forma diferente para las Constantes y Variables.

En el caso de las constantes se pone `Public` delante de la palabra `Const`.

En el caso de las variables `Public` sustituye a `Dim`.

El código del módulo quedaría así:

```
Option Compare Database
Option Explicit

Public Const Pi As Double = 3.14159265358979
Public Precio As Currency
Public Nombre As String
```

Si ahora pulsamos con el ratón en la ventana Inmediato, escribimos **`? Pi`** y [Enter]

Nos escribirá `3,14159265358979.`

Vamos a hacer otra prueba

Escribe en la ventana Inmediato **`Pi = 3.1416`** y pulsa [Enter]. ¿Qué ha pasado?


```
Pi = 3.1416
```

Simplemente que estamos intentando asignar un valor a una constante que ya lo tenía.

En definitiva una constante sólo puede tomar el valor una vez, "y a ti te encontré en la calle".

Por definición una constante no puede cambiar.

Sólo mantiene relaciones y se casa una vez.

En cambio las variables son mucho más promiscuas.

Todo lo anterior, entendedlo como una introducción a la declaración de variables y Constantes, que en realidad presenta más posibilidades que las aquí vistas.

En puntos posteriores, estudiaremos el tema con más profundidad.

Aquí han aparecido cosas nuevas; por ejemplo la palabra `Currency` y la palabra `String`.

En el ejemplo, con `Currency`, hacemos que la variable `Precio` sea del tipo Moneda; un tipo de datos "aparentemente" de coma flotante creado para manejar datos monetarios sin errores de "Redondeo".

Podéis obtener ayuda sobre estos tipos de datos poniendo el cursor, por ejemplo en cualquiera de sus nombres, por ejemplo `Currency` y pulsando [F1].

Con `String` hacemos que la variable `Nombre` sea tratada como una variable de tipo Cadena, que permite manejar cadenas de Caracteres.

En realidad podríamos haber hecho las declaraciones sin especificar su tipo pero, por el momento sin más explicaciones, os diré que esto genera código menos optimizado e incluso puede dar problemas a la hora de encontrar posibles errores.

Por ejemplo podríamos haber hecho:

```
Public Const Pi = 3.14159265358979
Public Precio
Public Nombre
```

Y la cosa funcionaría…

En realidad el declarar así una variable ó constante, equivale a haber hecho esta otra declaración

```
Public Const Pi As Variant = 3.14159265358979
Public Precio As Variant
Public Nombre As Variant
```

-¿Y qué es eso de `Variant`?-

El `Variant` es un tipo de dato al que se le puede asignar casi cualquier cosa.

Esto lo hace a costa de consumir más recursos que otros tipos de variables y por consiguiente una velocidad más lenta en la ejecución.

Ya tenemos las bases para el siguiente paso:

- Procedimientos `Sub`
- Procedimientos `Function` ó Funciones

Procedimientos Sub

Un procedimiento `Sub` llamado también Procedimiento a secas es un conjunto de código que realiza determinadas tareas.

Suele estar contenido entre las expresiones `Sub` y `End Sub`

El término Sub puede ir precedido de otras expresiones, por ejemplo para delimitar el ámbito en el que puede ser llamado el procedimiento.

Al procedimiento se le pueden pasar una serie de datos para que los use internamente.

A estos datos se les llama Parámetros ó Argumentos.

Si en la ventana de código escribimos la palabra `Sub`, le ponemos encima el cursor y presionamos la tecla [**F1**], Access nos mostrará la ayuda aplicable a **Sub**.

En la ayuda podemos entre otras cosas podemos ver:

[**Private** | **Public** | **Friend**] [**Static**] **Sub** *nombre* [(*lista_argumentos*)]
[*instrucciones*]
[**Exit Sub**]
[*instrucciones*]

End Sub

Por cierto, estas líneas indican cómo es la estructura de un procedimiento `Sub`.

Los elementos que están entre Paréntesis Cuadrados **[ ]** son opcionales.

Cuando hay varias palabras separadas por Barras Verticales **|**, nos está indicando que podemos seleccionar una de ellas.

Según esto serían igual de válidas las sentencias:

```
Sub Procedimiento_01()

End Sub


Public Sub Procedimiento_02()

End Sub


Private Sub Procedimiento_03()

End Sub


Public Sub Procedimiento_04(Argumento1 As Double)

End Sub
```

Por supuesto que cada opción genera un código que se comportará de forma diferente.

Es igualmente válido el que ya hemos usado en la primera entrega.

```
Private Sub cmdSaludo_Click()
    Me.lblSaludo.Caption = "¡¡¡Hola Mundo!!!"
    Me.Caption = "¡Aquí estoy!"
End Sub
```

Con una salvedad, este último es un tipo especial de Sub. Y es especial porque captura el evento que se produce cuando de presiona el botón cmdSaludo de su formulario. Recordemos que un Evento es un mensaje que envía Windows y en este caso es capturado por Access.

En el módulo que hemos creado vamos a escribir el siguiente código:

```
Public Sub Circunferencia()
    Dim Radio As Single
    Radio = 2.5
    Debug.Print "Circunferencia = " & 2 * Pi * Radio
    Debug.Print "Círculo = " & Pi * Radio ^ 2 & " m2"
End Sub
```

Ahora, en la ventana Inmediato escribe `Circunferencia y presiona [Enter]`

Efectivamente, en esa ventana se escribirá:

```
Circunferencia = 15,7079632679489
Círculo = 19,6349540849362 m2
```

Vamos a analizar un poco esto.

Hemos creado un procedimiento llamado `Circunferencia`.

Este procedimiento es de los de tipo Sub.
La primera línea es el encabezado.

Al hacerlo Public, dentro de un módulo estándar, permitimos que se pueda llamar desde cualquier parte de la base de datos. Es un procedimiento público.

En la segunda línea declaramos la variable Radio, de tipo Single.

El tipo Single es un tipo de coma flotante, que ocupa 4 bytes, o lo que es lo mismo, 32 bits.

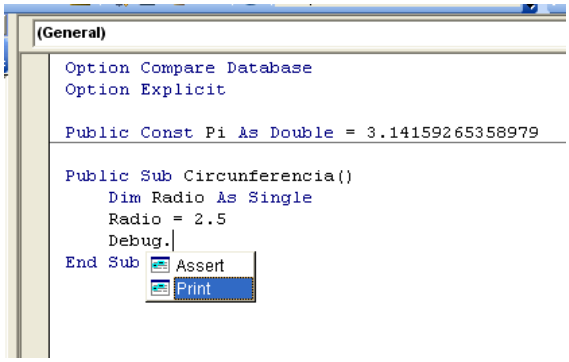
En la siguiente línea asignamos a la variable el valor `2.5`

Recuerdo que dentro del código, las comas separadores de decimales se ponen como puntos.

La siguiente línea utiliza el método Print del objeto Debug.

La forma de llamarlo es poniendo primero el objeto, a continuación un punto y después el método.

Fijaros que mientras lo escribís se os muestra una “Ayuda en línea” que os permite seleccionar lo que queréis escribir.



El objeto Debug es la propia ventana Inmediato.

El método Print imprime lo que se ponga a continuación.

Lo primero que imprime es una cadena, en nuestro caso **"Circunferencia = "**

A continuación vemos el operador **&** seguido de la expresión **2 * Pi * Radio**

Este operador hace de "Pegamento" entre dos expresiones.

En este caso enlaza la cadena primera con el resultado de la operación **2 * Pi * Radio**

Dando como resultado **Circunferencia = 15,7079632679489**

Lo mismo es aplicable a la siguiente línea.

Con `End Sub` se acaba el código del procedimiento.

-Esto está bien, pero sólo puede imprimir los datos de la circunferencia de radio 2,5, lo que no es gran cosa-.

-Bueno, sí. Es un inicio-.

-Ahora, me voy a sacar un conejo de la chistera y modifico el procedimiento Sub, de la siguiente forma:

```
Public Sub Circunferencia(Radio As Single)
    Debug.Print "Circunferencia = " & 2 * Pi * Radio
    Debug.Print "Círculo = " & Pi * Radio ^ 2 & " m2"
End Sub
```

Fijaros que la declaración de la variable Radio ya no se hace en el cuerpo del procedimiento, sino en su cabecera.

Además ha desaparecido la línea en la que asignábamos el valor del radio.

-¿Y cómo se utiliza esto?-.

No tiene ninguna complicación especial

Vamos a modificar la forma como llamamos desde la ventana Inmediato al procedimiento, y escribimos

```
Circunferencia 2.5
```

El resultado es el mismo que en el caso anterior.

Pero si escribimos

```
Circunferencia 7.8
```

Nos dará los valores correspondientes a una circunferencia de radio `7.8`.

En este caso hemos utilizado un procedimiento con un único argumento.

El argumento es precisamente `7.8`.

Si tuviera, por ejemplo 2, se escribirían separados por una coma

```
NombreProcedimiento Argumento1, Argumento2
```

Hay otro tipo de procedimientos de los que hemos hablado.

Se llaman procedimientos **`Function`**, las famosas **Funciones**.

Tienen la particularidad de que pueden devolver un valor, pero de eso hablaremos en la próxima entrega.