VBASIC ACCESS

El lenguaje Basic permite:

- Personalizar una aplicación creando tus propias funciones
- Tratamiento de errores runtime
- Crear o manipular objetos
- Realizar acciones a nivel de sistema
- Manejar registros concretos, uno a uno
- Pasar argumentos que convengan en el momento de la ejecución

Las macros no permiten la mayoría de estas cosas, y la mayoría de las acciones de macros se pueden ejecutar desde código con el objeto (antes instrucción) DoCmd. Las macros son interesantes para crear prototipos de la aplicación con rapidez o hacer acciones sencillas que no entrañen error; también hay algunas cosas que solo se pueden hacer con macros (según versiones).

MODULOS

El código Visual se almacena en los módulos de una base de datos de Access, en un fichero.mdb (en Visual Basic en algo más amplio que llamamos proyecto y que no tiene porque contener una base de datos). Conviene no confundir la Base de datos con el código o aplicación que lo maneja aunque esté contenida en el mismo fichero.

Cada modulo contiene una Sección de Declaraciones y a continuación una serie de procedimientos.

La Sección de declaraciones contiene la Instrucción Option Compare Database y puede contener otras como Option Explicit, Option Base, declaraciones de variables, etc... que afectan al módulo donde se encuentren estas declaraciones.

Cada instrucción ocupa solo una línea aunque una línea puede contener varias siempre que estén separadas por dos puntos (:).

Hay dos tipos de módulos:

- Módulos de formulario o informe llamados **LOCALES**. Son privados de ese objeto y se crean y borran con él ya que forman parte de su diseño; se les puede añadir procedimientos de evento o generales.
- Módulos **GLOBALES**. Son objetos independientes y sus procedimientos pueden ser llamados desde cualquier sitio (expresiones, procedimientos, macros, etc. estén en ese u otro módulo, sea local o global). Los procedimientos de evento no tienen sentido aquí.

Para entrar en un modulo local basta seleccionar el objeto (formulario o informe) y pulsar la opción Código del menú Ver (también con el botón código de la barra de herramientas o de iconos; otra de las formas es Generar evento en el menú de método abreviado (el que se obtiene pulsando el botón derecho del ratón) para ese objeto o para cualquiera de los controles contenidos en él.

PROCEDIMIENTOS

Hay dos tipos de procedimientos:

- **[PRIVATE][PUBLIC] SUB** nombre_procedimiento [(argumentos)]

END SUB

- No devuelven valor por lo que no se pueden usar en expresiones
- Aceptan argumentos o parámetros
- Todos los procedimientos de evento son de éste tipo y se encuentran siempre en módulos locales.

- **[PRIVATE][PUBLIC] FUNCTION** nombre_procedimiento [(argumentos)] [As tipo]

END FUNCTION

- Devuelven siempre un valor por lo que se pueden usar en expresiones
- Aceptan argumentos o parámetros

Los procedimientos son útiles:

- Cuando determinadas operaciones siempre se vayan a hacer de una determinada forma aunque esté parametrizada.
- Si hay que modificar esa operación solo será necesario modificarla en un solo sitio.
- Las operaciones pueden ser complejas (se podrán utilizar estructuras de control, variables, etc.
- Se pueden controlar y recuperar los errores de ejecución (errores runtime)
- Se pueden incluir comentarios en el código para documentarlo, aclararlo.

Los procedimientos de un módulo local son privados (por eso puede haberlos con el mismo nombre en otro módulo local, nunca en el mismo) a no ser que se declaren públicos (PUBLIC) mientras que los procedimientos de un módulo global son públicos (y no puede haber otro procedimiento que se llame igual), a no ser que se declaren en él como privados (PRIVATE).

Los procedimientos privados solo pueden ser llamados desde el correspondiente módulo, mientras que los públicos pueden ser llamados desde cualquier módulo.

Si hubiera dos procedimientos con el mismo nombre en dos módulos distintos, Access primero lo buscaría en el modulo activo (local o global) y si no lo encontrara lo buscaría en los módulos globales (o en el resto de los módulos globales).

NOMBRES EN BASIC ACCESS

- Deben comenzar con una letra
- Solo pueden contener letras, números y el signo de subrayado
- Longitud máxima: 40 caracteres
- No deben contener palabras reservadas (nombres de instrucciones, métodos, funciones de librería, operadores)

VARIABLES

Almacena valores fuera de las tablas. Tiene nombre y tipo y en algún caso se les puede asignar longitud predeterminada.

Declaración: Visual Access no obliga a declarar las variables sino que al usarlas simplemente Visual Access las crearía (declaración implícita), pero con un tipo de datos (variant) que puede no ser conveniente. CONVIENE DECLARAR las variables explícitamente para detectar errores de ejecución y para documentar el código.

Las variables se pueden clasificar de varias maneras:

LOCALES O GLOBALES(por su visibilidad), DINAMICAS O ESTATICAS (vida)

DIM nombre_variable [AS tipo] [, nombre_variable [AS tipo]...]

Las variables declaradas con DIM serán locales y dinámicas, es decir que son visibles en el procedimiento donde se crearon y desaparecen al finalizar el procedimiento donde se crearon (distinto es que se puedan pasar como parámetros a otros procedimientos).

STATIC nombre_variable [AS tipo] [, nombre_variable [AS tipo]...]

Las variables declaradas con STATIC serán locales y estáticas, es decir que son visibles en el procedimiento donde se crearon pero NO desaparecen al finalizar el procedimiento donde se crearon, por lo que conservaran el valor que tenían en la ocasión anterior.

PUBLIC nombre_variable [AS tipo] [, nombre_variable [AS tipo]...]

Las variables declaradas como PUBLIC (antes GLOBAL) permitirá que una variable pueda ser visible desde cualquier procedimiento y no desaparezca hasta que finalice el programa.

NOTA: Todas estas ordenes permiten también crear matrices, determinar el tamaño (según casos) y cláusulas relacionadas con la programación con objetos).

ESTRUCTURAS DE CONTROL DE FLUJO DE PROGRAMA

Estructuras condicionales.

If... End If

Ejecuta condicionalmente un grupo de instrucciones, dependiendo del valor de una expresión.

Sintaxis

If *condición* **Then**
[*instrucciones*]

[**ElseIf** *condición−n* **Then**
[*instrucciones_elseif*] ...

[**Else**
[*instrucciones_else*]]

End If

Select Case... End Select

Ejecuta uno de varios grupos de instrucciones, dependiendo del valor de una expresión.

Sintaxis

Select Case *expresión_prueba*

[**Case** *lista_expresion−n*
[*instrucciones−n*]] ...
[**Case Else**
[*instrucciones_else*]]

End Select

****** Se puede poner: **Case 2 To 5 (como rango)**

****** O incluso una lista: **Case 5,6,8,12,−4**

<u>Estructuras de repetición o bucles.</u>

Do...Loop

Repite un bloque de instrucciones cuando una condición es **True** o hasta que una condición se convierta en **True**.

Sintaxis

Do [{**While** | **Until**} *condición*]
[*instrucciones*]
[**Exit Do**]
[*instrucciones*]

Loop

O bien, puede utilizar esta sintaxis:

Do
[*instrucciones*]
[**Exit Do**]
[*instrucciones*]

Loop [{**While** | **Until**} *condición*]

For...Next

Repite un grupo de instrucciones un número especificado de veces.

Sintaxis

For *contador* = *principio* **To** *fin* [**Step** *incremento*]
[*instrucciones*]
[**Exit For**]
[*instrucciones*]

Next [*contador*]

<u>ESQUEMA DE UNA APLICACIÓN</u>: Ejemplo...

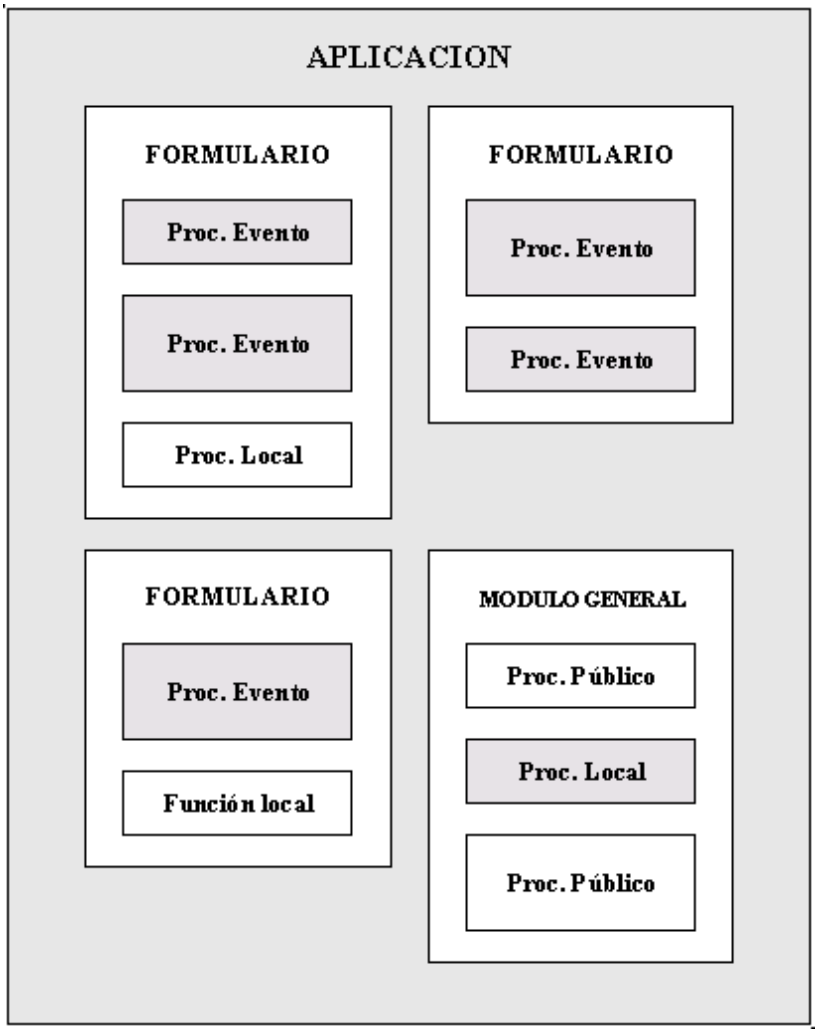

TABLA DE TIPOS DE VARIABLES:

Tipo de datos Tamaño de almacenamiento Intervalo

Byte 1 byte 0 a 255

Boolean 2 bytes True o False

Integer 2 bytes −32.768 a 32.767

Long (entero largo) 4 bytes −2.147.483.648 a 2.147.483.647

Single (coma flotante/ precisión simple) 4 bytes −3,402823E38 a −1,401298E−45 para

valores negativos; 1,401298E−45 a 3,402823E38 para valores positivos

Double (coma flotante/ precisión doble) 8 bytes −1,79769313486232E308 a

−4,94065645841247E−324 para valores negativos; 4,94065645841247E−324 a 1,79769313486232E308 para valores positivos

Currency (entero a escala) 8 bytes −922.337.203.685.477,5808 a

922.337.203.685.477,5807

Decimal 14 bytes +/−9.228.162.514.264.337.593.543.950

sin punto decimal; +/−7,9228162514264337593543950335 con 28 posiciones a la derecha del signo decimal; el número más pequeño distinto de cero es +/−0,0000000000000000000000000001

Date 8 bytes 1 de enero de 100 a 31 de diciembre de 9999

Object 4 bytes Cualquier referencia a tipo Object

String (longitud variable) 10 bytes + longitud Desde 0 a 2.000 millones de la cadena

String (longitud fija) Longitud de la cadena Desde 1 a 65.400 aproximadamente

Variant (con números) 16 bytes Cualquier valor numérico hasta el intervalo de

un tipo Double

Variant (con caracteres) 22 bytes + longitud El mismo intervalo que para un tipo String de

longitud variable de cadena

Definido por el usuario (utilizando Type) Número requerido por los elementos. El

intervalo de cada elemento es el mismo que el intervalo de su tipo de datos.