Ediciones ENI

VBA Access 2016

Programar en Access

Colección
Recursos Informáticos

Extracto del Libro

Capítulo 5
Los objetos y colecciones en VBA

1. Noción de objeto

VBA es un lenguaje que permite hacer programación orientada a objetos (POO): un objeto representa una idea, concepto o entidad del mundo real, como un avión, un individuo e incluso una película. Tiene una estructura interna y un comportamiento, y se puede comunicar con sus homólogos. Los elementos que permiten describir un objeto forman lo que se llama una clase. Cada objeto que proviene de una clase es una instancia de clase. Las clases tienen propiedades, métodos y eventos.

1.1 Propiedades

El objeto es una entidad que podemos distinguir gracias a sus propiedades (su color o dimensiones, por ejemplo). Si tomamos como ejemplo un libro, este se caracteriza por sus propiedades: número de páginas, título, número de capítulos, editor, contenido, etc. Cada una de sus propiedades se puede especificar en cada libro, pero todos los libros tienen fundamentalmente las mismas propiedades. Algunas propiedades de los objetos se pueden modificar (velocidad de un coche, por ejemplo) y otras no (una marca de coche).

En programación con VBA, la sintaxis general de acceso a las propiedades de un objeto es la siguiente:

```
MiObjeto.SuPropiedad
```

Aquí se accede a la propiedad `SuPropiedad` del objeto `MiObjeto`. La combinación `MiObjeto.SuPropiedad` tendrá el mismo comportamiento que una variable clásica, que puede tomar un valor o devolver un valor con el uso del operador =.

Por ejemplo, es posible leer el contenido SQL de una consulta de Access y visualizarla con el siguiente código:

```
Sub LeerSQLConsulta()
    Dim UnaConsulta As QueryDef
    UnaConsulta.SQL = "SELECT * FROM MiTabla"
    ...
    MsgBox UnaConsulta.SQL
End Sub
```

En el capítulo Los objetos de acceso a los datos DAO y ADO, volveremos al objeto `QueryDef` con más detalle.

Una propiedad de un objeto puede ser un objeto ella misma, por lo que se podría utilizar más adelante en el código.

1.2 Métodos

Además de los elementos que caracterizan un objeto, también es posible realizar acciones con estos objetos, como por ejemplo "despegar", "acelerar", "pasar a otra página", etc. Estas acciones se llaman métodos. Estas acciones se representan en forma de procedimientos y funciones en VBA, según puedan o no devolver valores. Algunas acciones pueden necesitar argumentos para funcionar (número de caracteres de un libro, con o sin espacios).

En programación con VBA, la sintaxis general de acceso a los métodos de un objeto es la siguiente:

```
MiObjeto.SuMetodo [MiArgumento]
```

En el siguiente ejemplo, se trata simplemente de refrescar el vínculo entre una tabla relacionada y la aplicación Access:

```
Sub RefrescarRelacionTabla()
    Dim UnaVariableTemporal As TableDef
    ...
    UnaVariableTemporal.RefreshLink
End Sub
```

Y, en caso de que el método sea una función, podemos almacenar el resultado en una variable, como en el siguiente ejemplo:

```
Sub Número_Campos()
    Dim Nm_Campos As Integer
    Dim MiTabla As TableDef
    ...
Nm = MiTabla.Fields.Count
End Sub
```

De la misma manera que en las llamadas a funciones o procedimientos, los argumentos se separan por comas en las llamadas a métodos del objeto.

1.3 Eventos

Para cada objeto, es posible detectar y tener en cuenta el resultado de acciones particulares externas, que llamamos eventos. Todos los eventos que tienen lugar durante la ejecución de un programa se detectan y gestionan por la máquina, pero, según la opción del desarrollador, solo algunas se pueden tratar de manera particular, como por ejemplo un clic en un botón, una casilla de selección que se marca, la apertura o cierre de una ventana, etc. Estos eventos se gestionan a través de procedimientos que algunas veces tienen argumentos y otras no.

En VBA, la sintaxis más frecuente para los eventos es la siguiente:

```
Sub MiObjeto_MiEventoDestacado()
    'El código que se ejecutará cuando se detecte el evento
End Sub
```

Por ejemplo, cuando se trata de un clic en un botón llamado `MiBoton`, tendrá el código siguiente:

```
Sub MiBoton_Click()
    'código que se ejecutará
End Sub
```

Los eventos en Access se tratarán más adelante en detalle, en el capítulo Los eventos de Access.

1.4 Las colecciones

Cuando varios objetos de una misma clase se agrupan, pueden pertenecer a una misma colección de objetos. Para referirse a un elemento en particular de una colección de objetos, hay varias sintaxis, entre las siguientes:

```
Nombre_Colección!Nombre_Objeto
Nombre_Colección![nombreObjeto]
Nombre_Colección("NombreObjeto")
Nombre_Colección(variable_Nombre) 'variable_Nombre es una cadena de
caracteres que contiene el nombre del objeto
Nombre_Colección(variable_Numero) 'variable_Numero es un valor
digital que contiene el número del objeto dentro de la colección.
```

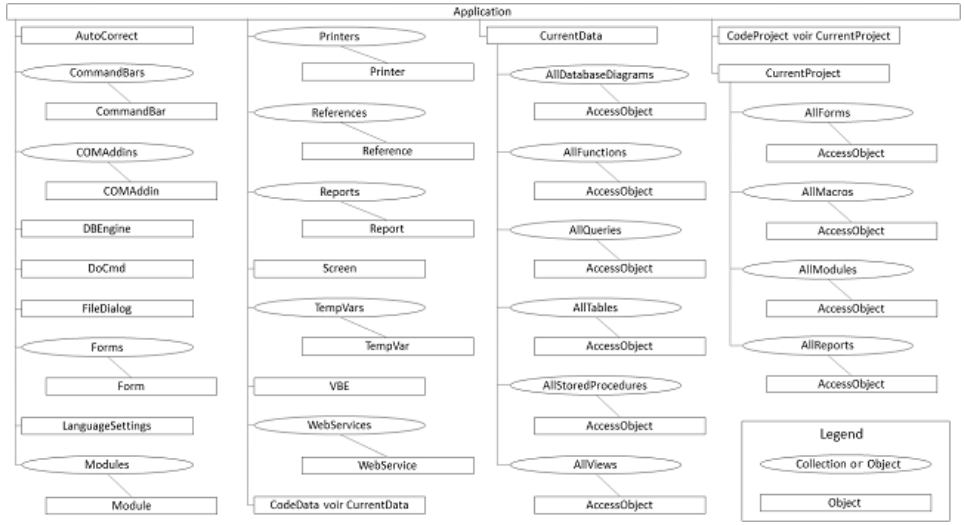
Las dos sintaxis que se utilizan más habitualmente son la tercera y la quinta, ya que permiten el uso de IntelliSense (ver capítulo VBE y seguridad en Access 2016).

Observación

Es habitual en las colecciones comenzar a contar en 0 y no en 1. Además, el número de un objeto dentro de una colección dependerá de la presencia de otros elementos antes o después de él, lo que no garantiza localizar el objeto correcto por este medio.

2. Modelo de objeto de Access

El objetivo de algunas de las secciones siguientes es mostrar el modelo jerárquico utilizado dentro de la aplicación Access, en forma de colecciones de objetos, que se van a explicar a continuación.



3. Colecciones en Access

A continuación se muestran las principales colecciones que podemos manipular con VBA en Access.

Colección	Contiene una colección de	Descripción
`COMAddins`	`COMAddin`	Colección de los **complementos COM**
`CommandBars`	`CommandBar`	Colección de las **barras de comando**
`Forms`	`Form`	Colección de los **formularios abiertos**. Ver también `CurrentProject.AllForms`

Colección	Contiene una colección de	Descripción
`Modules`	`Module`	Colección de los **módulos**
`Printers`	`Printer`	Colección de las **impresoras disponibles**
`References`	`Reference`	Colección de las **referencias a librerías**. Ver **Herramientas - Referencias**
`Reports`	`Report`	Colección de **estados**. Ver también `CurrentProject.AllReports`
`TempVars`	`TempVar`	Colección de las **variables temporales**
`WebServices`	`WebService`	Colección de las conexiones a **servicios web**

4. Objetos de Access

A continuación se muestran los principales objetos que es posible manipular en el modelo Access.

Objeto	Descripción
`Application`	Representa la aplicación Microsoft Access activa.
`AutoCorrect`	Representa las opciones de corrección automática de Access.
`DBEngine`	Representa el motor de base de datos Microsoft Jet. Este objeto permite controlar el resto de los objetos de acceso a los datos.
`DoCmd`	Permite convertir en VBA las acciones de Macros.
`FileDialog`	Permite acceder a las funcionalidades de los cuadros de diálogo (Abrir o Guardar, por ejemplo).

Ediciones ENI

VBA Access 2016

Aprenda a crear aplicaciones profesionales: Ejercicios y correcciones

Colección
Prácticas Técnicas

Extracto del Libro

Capítulo 4
Estructuras de control

Duración: 2 horas

Palabras clave

condición, elección, comprobación, alternativa, salto condicional, contador, iteración, bucle, incrementar, IsNumeric, If, ElseIf, Then, Select Case, MsgBox, vbYes, vbNo, vbYesNo, vbOK, Prompt, Do, While, Until, For, Next, Step

Objetivos

Dominar las estructuras de decisión para comprobar condiciones y seguidamente realizar diferentes acciones según el resultado obtenido. Dominar las estructuras de bucle que, asociadas a las instrucciones condicionales, permiten escribir código Visual Basic para la toma de decisiones y la repetición de acciones. Encontrará todas estas estructuras en los capítulos que tratan la programación por eventos y de objeto. En este capítulo, nos limitamos al uso de cuadros de diálogo ya vistos con anterioridad.

Requisitos

Para comprobar los requisitos necesarios antes de iniciar esta práctica técnica, responda a las siguientes preguntas (algunas preguntas admiten más de una respuesta):

1. Para realizar un salto condicional, se utiliza:

- **a.** `If ... Then ... Else ... End If`
- **b.** `Do ... Loop`
- **c.** `Select Case ... Case ... End Select`

2. ¿Qué devuelve la variable **blnCondicion** en el siguiente ejemplo?

```
Dim blnCondicion as Boolean
Dim intA as Integer, Dim intB as Integer
intA=5
intB=9
blnCondicion = IIf(intA=intB, True, False)
```

a. True

b. False

c. Null

3. De entre las instrucciones siguientes, pertenecientes cada una de ellas a una estructura de control distinta, ¿cuáles son correctas?

a. `Case If N1 > N2`

b. `Case A, B, C`

c. `Case 1 to 10`

d. `Case Nombre, Is > 50`

4. La palabra clave `ElseIf`:

a. Puede aparecer a continuación de una cláusula `Else`.

b. Es opcional en una condición.

c. Puede utilizarse varias veces en un bloque `If`.

5. ¿Qué bucle de instrucciones itera mientras que la condición no devuelva el valor True?

a. `For Each . . . Next`

b. `For ... Next`

c. `Do Until ... Loop`

d. `While ... Wend`

e. `With ... End With`

Observación

Para las siguientes tres preguntas, las opciones son idénticas a las de la pregunta cinco.

6. ¿Qué iterador se debería utilizar para ejecutar instrucciones un cierto número de veces conocido de antemano?

7. ¿Con cuál se debería repetir un grupo de instrucciones para cada elemento de una tabla o de una colección?

8. ¿Con cuál se debería ejecutar una serie de instrucciones aplicadas a un solo objeto de un tipo predefinido por el usuario?

Nota

Aplicación **Estructuras de control.accdb**

A continuación puede ver la pantalla de inicio de la aplicación Access que puede descargar de la web de Ediciones ENI.

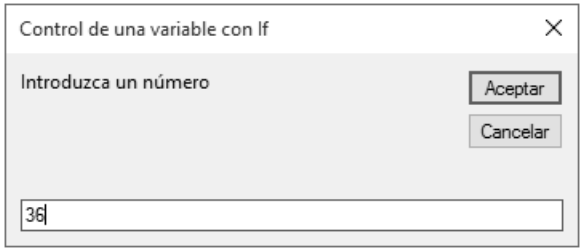
Solución pág. 249

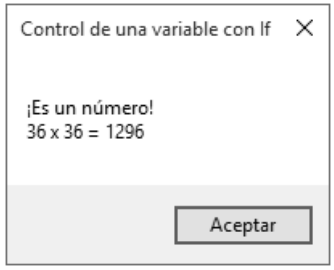
Enunciado 4.1 Controlar la entrada de datos con If

Duración estimada: 10 minutos

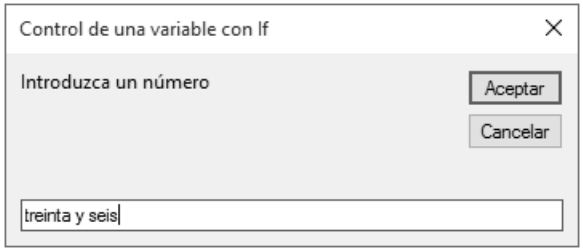
4.1.1 - Comprobar que el dato introducido es un número

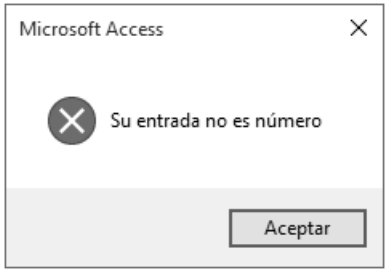
Escriba el procedimiento **ControlarEntrada** que muestre un cuadro de diálogo con un campo de texto y compruebe que se trate de un número. En caso afirmativo, realice la operación de multiplicarlo por sí mismo. En caso contrario, muestre un mensaje.
Ejemplo:





si no:





Observación

Las funciones **Is** permiten comprobar el tipo de una variable o de una entrada.

4.1.2 - Multiplicar dos números introducidos por el usuario

Escriba un procedimiento que permita al usuario multiplicar dos números.

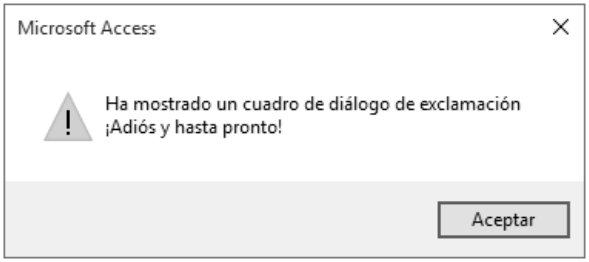
Pista

Compruebe los valores introducidos por el usuario.

Solución pág. 250

Enunciado 4.2 Gestionar condiciones alternativas con ElseIf

Duración estimada: 10 minutos
Complete el procedimiento **DecirAdios**. Este muestra una sucesión de cuadros de diálogo con el objetivo de mostrar el mensaje "¡Adiós, y hasta pronto!" mostrado a continuación en función de las elecciones del usuario.



Observación

La ayuda en línea permite conocer mejor los distintos valores que pueden adquirir los botones de los cuadros de diálogo.

Pista

```
Sub DecirAdios()
  Dim strEstilo
  Dim intEleccion As Integer
  Dim varError As Variant
  On Error GoTo GestionDeErrores
  strEstilo = vbOKCancel + vbDefaultButton1
  ... = MsgBox("Pulse Aceptar para ver el siguiente cuadro" & _
               "de diálogo", strEstilo)
  If ... = vbOK Then
     ... = vbYesNoCancel + vbDefaultButton1 + vbQuestion _
     ... = MsgBox("Ha mostrado un cuadro de diálogo de interrogación," & _
                  ¿Quiere continuar?", strEstilo)
     If intEleccion = ... Then
        ... = vbOKOnly + vbExclamation
        ...
     ElseIf intEleccion = ... Then
        strEstilo = vbOKOnly + vbCritical
        intEleccion = MsgBox("Peligro", strEstilo)
     Else
        strEstilo = vbOKOnly + vbCritical
        intEleccion = MsgBox("Ha pulsado cancelar", strEstilo)
     End If
  Else
```

```
      strEstilo = ...
      intEleccion = ...
   End If
Exit Sub
GestionDeErrores:
   varError = MsgBox("Error N° " & Err.Number & " - " & Err.Description)
End Sub
```

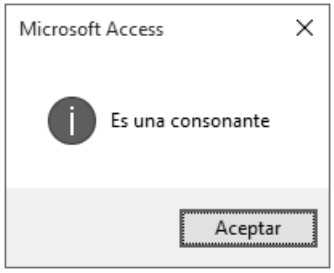
Solución pág. 252

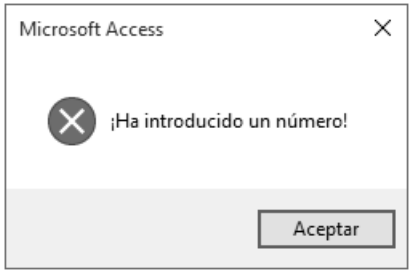
Enunciado 4.3 Evaluar varios casos con Select Case

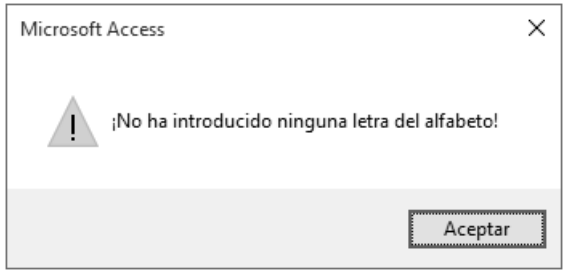
4.3.1 - Identificar una letra del alfabeto

Duración estimada: 10 minutos

Escriba el procedimiento **ConsonanteVocal** que solicita la introducción de una vocal o una consonante. Muestre los siguientes mensajes en función de lo que se introduzca:







Pista

Compruebe la letra en mayúscula.