Lección 4

En esta lección vamos a ver que son y para que sirven: las variables, constantes, tablas y matrices. Todas ellas son estructuras de almacenamiento de datos temporales, pero que nos pueden facilitar la programación en muchas ocasiones.

Variables, constantes y matrices

No podemos decir que estos elementos sean estructuras básicas, ya que no son un grupo de instrucciones, sino que son elementos que nos pueden ayudar a almacenar valores, de forma temporal, para usarlos en nuestra aplicación de la forma que más nos convenga. También tenemos que pensar que en muchas estructuras que veremos en lecciones posteriores utilizaremos variables, constantes, tablas y matrices.

Variables

Las variables se utilizan, en cualquier lenguaje de programación, para almacenar valores de forma temporal (mientras dura la ejecución del programa). A las variables se les pone un nombre para poder trabajar con ellas y se indican de que tipo son. Este tipo nos informa que clase de datos se pueden almacenar dentro de esta variables, (los diferentes tipos de variables los veremos más adelante).

Constantes

Las constantes nos pueden parecer que son exactamente iguales que las variables, pero no es así. Las variables nos sirven para almacenar valores, que podemos modificar durante la ejecución del programa.

Las constantes, en cambio, no cambian de valor durante la ejecución de la aplicación. Se suelen utilizar para sustituir un número o valor, difícil de recordar o que suele salir muchas veces durante la aplicación. Imagina el caso de una aplicación en la que necesites utilizar muchas veces el valor **Pi**. Si siempre que necesitáramos este valor tuviéramos que escribir 3’1415... sería un poco engorroso. Pero, gracias a las constantes nosotros podemos definir una llamada **Pi** con valor 3’1415... y en el momento en el que necesitemos realizar una operación con el valor **Pi** solo tendremos que poner el nombre de la constante y el ordenador se encargará de sustituirlo por su valor.

Matrices

Las matrices son un grupo de valores que tienen un mismo nombre y se diferencian entre ellas por el lugar que ocupan. A este lugar que ocupa se le llama **Indice**. Gracias a este índice podemos crear un código, utilizando estructuras de repetición que nos ayuden a trabajar con estos datos, ahorrando de esta manera código.

Normalmente, las matrices se definen con un límite inferior y uno superior. Con la resta de ambos tenemos el número de elementos que pueden entrar dentro de la matriz. Tenemos que pensar que si nosotros solo definimos el valor superior, el primer objeto tendrá como índice **0** y el último el número que nosotros hayamos definido como límite superior. De esta forma, si nosotros definimos una matriz de una sola dimensión con límete superior 5, en realidad tenemos 6 objetos con índices 0, 1, 2, 3, 4, 5.

Al igual que las variables y constantes, en las tablas también se tiene que definir el tipo de valor que vamos a almacenar dentro.

Las matrices podemos definirlas de solo una dimensión, como si se tratase de una gran fila de datos y en otras muchas ocasiones nos puede interesar utilizar estructuras de dos dimensiones o incluso más, en las que buscaremos los datos por la **fila** y la **columna** que ocupan. En este caso estas matrices tendrán 2 índices.

Por ejemplo en el caso de un tablero de ajedrez nos interesa saber en que **fila** y en que **columna** se encuentra situado una ficha determinada para saber si el movimiento que deseamos realizar está o no permitido.

Más adelante explicaremos como crear, definir y trabajar con variables, constantes y matrices.

Vida de una variable

Nosotros podemos definir una variable para que solo nos sea útil mientras dura el procedimiento en el que se le ha creado. En el momento de finalizar dicho procedimiento, el espacio reservado en memoria para la variable queda liberado. De esta forma, si nosotros queremos utilizar un mismo nombre para diferentes variables que se utilizan en diferentes procedimientos podemos hacerlo sin miedo a que los valores se mezclen o cambien sin que nosotros tengamos control sobre dichos cambios.

Definir una variable

Para definir una variable dentro de un procedimiento utilizaremos la instrucción: **Dim [Nombre Variable] As [Tipo variable]**.

Nombre Variable: definiremos el nombre que tiene la variable. Este nombre no puede tener más de 255 caracteres, debe comenzar con una letra y no debe contener puntos.

Tipo variables: Especificaremos el tipo de datos que se pueden almacenar dentro de la variable.

Tipos de variables

Vamos a especificar los diferentes tipos de datos que podemos definir al crear una variable, constante o matriz.

Byte: Este tipo de dato lo utilizamos para contener números enteros positivos en el intervalo de 0 a 255.

Boolean: En este tipo de doto sólo tiene dos posibles valores, **True (-1)** o **False (0)**.

Integer: En este tipo de datos contiene variables enteras almacenadas como números enteros de 2 bytes en el intervalo de –32.768 a 32.767.

Long: Almacena números completos entre –2.147.483.648 y 2.147.483.647.

Currency: Es un tipo de datos con un intervalo de -922.337.203.685.477,5808 a 922.337.203.685.477,5807. Es recomendable utilizar este tipo de dato para cálculos de tipo monetario y cálculos de punto fijo donde es muy importante la exactitud.

Single: El intervalo que se puede almacenar en este tipo de datos es de: -3,402823E38 a -1,401298E-45 para valores negativos y de 1.401298E-45 a 3.402823E38 para valores positivos.

Double: El intervalo de almacenamiento de este tipo de dato es de: -1,79769313486232E308 a -4,94065645841247E-324 para valores negativos y de 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos.

Es importante, una vez vistos los intervalos, de cada uno de los diferentes tipos de datos, saber utilizar y definir cada uno de ellos según nuestras necesidades. Debes pensar que si utilizamos un tipo de dato que tenga un intervalo muy pequeño para realizar cálculos complejos puede ser que nos de error en según que casos y si utiliza-

mos un tipo de variable con un intervalo muy grande para una operación muy sencilla estaremos ocupando memoria innecesariamente.

Declaración de una variable

En este apartado veremos como y que métodos podemos utilizar para declarar una nueva variable.

Declaración implícita

En un principio, dentro de un procedimiento, no hace falta que definamos el nombre y el tipo de una nueva variable. Si Visual Basic encuentra un nuevo nombre de una variable, este la define automáticamente. Esto en ocasiones puede ir muy bien pero en otras ocasiones nos puede producir un error que puede ser difícil de detectar. Imagínate que nosotros en una línea de código hacemos referencia a una variable llamada **Contador**. **Visual Basic** como ve que es una nueva variable la crea automáticamente. Pero, si nosotros en otro momento queremos hacer referencia a esta nueva variable y escribimos **Contaidor**, **Visual Basic** cree que es una nueva variable creando así una diferente.

De esta forma si queremos hacer cálculos con la variable **Contador**, puede ser que tengamos algún tipo de error.

Entonces, viendo esto nos tenemos que plantear si es mejor definir nosotros las variables o dejar que sea **Visual Basic** quien defina las nuevas variables que vamos necesitando.

Declaración explícita

Para no tener problemas como el anteriormente citado, nos podemos obligar a definir las variables que utilizamos. Al definirlas, tanto debemos indicar un nombre para la nueva variable, como el tipo de dato que vamos a almacenar. De esta forma, si **Visual Basic** encuentra una nueva variable, en lugar de crearla, nos avisa que no la hemos definido, produciéndose así un error de compilación.

Para hacer que **Visual Basic** utilice la declaración explícita podemos utilizar dos sistemas diferentes.

Podemos hacer que **Visual Basic** active esta opción para todos y cada uno de los módulos que se creen a partir del momento de activar esta opción o activarlo nosotros manualmente.

. Práctica 1

1. Abre un nuevo proyecto.

*2. Selecciona **Opciones** dentro del menú **Herramientas**.*

*3. De todas las carpetas selecciona **Editor** y activa la opción **Requerir declaración de variables**.*

4. Acepta el cuadro de diálogo actual.

*5. Mira el código de este proyecto, con el menú: **Ver - Código** o pulsa **F7**.*

6. Observa que no hay ninguna línea de código en nuestro proyecto.

7. Cierra el proyecto actual, sin guardar los cambios.

*8. Abre un nuevo proyecto, con la opción **Nuevo proyecto** de la opción **Abrir**. En el cuadro de diálogo que te aparece a continuación deja la selección actual y pulsa en **Aceptar**.*

9. Mira el código de este proyecto.

Observa como ahora dentro del apartado **(General) (Declaraciones)** aparece la instrucción **Option Explicit**. Esta instrucción nos indica que en este proyecto solo se pueden utilizar variables que se definan dentro de este mismo apartado o dentro de los diferentes procedimientos de los que conste la aplicación.

Si nosotros queremos insertar esta instrucción en una aplicación que ya tengamos en uso o que ya exista, tendremos que escribirla dentro del apartado que hemos especificado anteriormente.

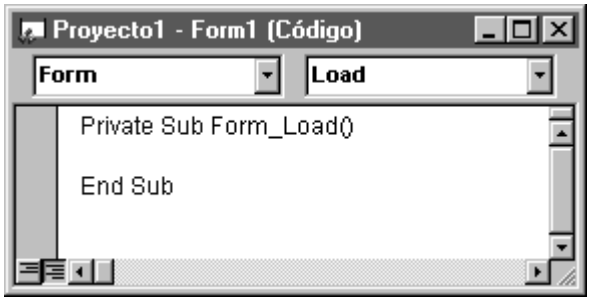
Vamos a ver como podemos llegar hasta este apartado de una forma fácil.

. Práctica 2

*1. Con el último proyecto en pantalla, quita la selección: **Requerir declaración de variables**.*

2. Abre un proyecto que tengas grabado.

3. Accede al código de cualquiera de los objetos que tienes en el formulario.



Observa la pantalla con el código. (En nuestro caso estamos enseñando el código que aparece en el momento de hacer doble clic en el formulario de una aplicación nueva).

Observa como en dicha ventana de código siempre aparece dos listas desplegables. La lista de la **izquierda** es donde se irán situando los **nombres** de los diferentes objetos que están insertados en el formulario actual. Mientras que en la lista de la **derecha** aparecerán los **eventos** del objeto que esté seleccionado en la lista de la izquierda.

*4. Despliega la lista de la **izquierda** y selecciona la opción **(General)** observa como la lista de la **derecha** cambia y aparece **(Declaraciones)**, si no aparece automáticamente despliega la lista y busca dicha opción.*

*5. Cuando estás en este apartado ya puedes escribir **Option Explicit**.*

De aquí en adelante, si continuas trabajando con este proyecto, en el momento de utilizar una nueva variable, si no está creada te aparecerá un mensaje de error.

¿Donde declaramos las variables?

Nosotros podemos declarar una variable en diferentes sitios, según el alcance

que le queramos dar. Nosotros podemos crear variables que solo se utilicen dentro de un procedimiento determinado, variables que se pueda utilizar en cualquier procedimiento de un formulario o en cualquier procedimiento de cualquier formulario de un mismo proyecto.

Es esta lección vamos a ver los dos primeros casos, mientras que el tercero lo veremos en el momento de insertar varios formularios dentro de un mismo proyecto.

De aquí en adelante pensaremos que tenemos escrita la instrucción **Option Explicit** para que no podamos utilizar una variable sin que antes la tengamos definida.

Variables útiles en un procedimiento

Si nosotros queremos crear una variable dentro de un procedimiento lo podemos hacer en cualquier punto dentro de este. (No hace falta crearlo en un lugar determinado siempre y cuando se declare antes de utilizarla).

Siempre, al utilizar este tipo de variables, deberemos pensar que el valor que tenga una variable dentro de este procedimiento no se podrá consultar o variar desde cualquier otro.

. Práctica 3

1. Crea un nuevo proyecto.

*2. Inserta dos **CommandButton** a los que llamaremos **Boton1** y **Boton2**.*

*3. Inserta un **Label** al que llamaremos **Valor**.*

*4. Escribe dentro del **Boton1**, haciendo doble clic, estas líneas de código.*

```
Private Sub Boton1_Click()
   Dim Contador As Integer
   Valor.Caption = Contador
End Sub
```

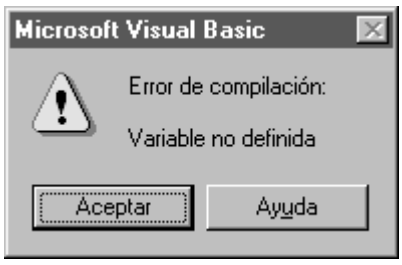
*5. Y dentro del **Boton2** estas otras:*

```
Private Sub Boton2_Click()
   Valor.Caption = Contador
End Sub
```

Observa como en el primer botón hemos definido una variable llamada **Contador**, mientras que en el segundo **Botón** no.

*6. Realiza una ejecución de prueba. Pulsa en el primer **botón**. Observa como el valor de la variable a pasado a nuestro **Label**.*

*7. Pulsa ahora en el segundo **botón**.*



Se produce un error, apareciendo una ventana como la que mostramos en esta imagen. Este error nos avisa que existe una variable que no está definida.

Aunque parezca que la tenemos definida no es así. La definición de dicha variable está en otro procedimiento.

*8. Pulsa el botón **Aceptar** y observa donde se ha producido el error.*

9. Detén la ejecución de la aplicación.

Si deseas utilizar una variable con el mismo nombre en otro procedimiento deberías volverla a definir. Piensa que aunque se llamen exactamente igual, son variables diferentes ya que están en procedimientos diferentes.

Si nosotros creamos las variables con **Dim** al volver a entrar dentro del evento donde se ha creado la variable, esta se vuelve a iniciar. Si queremos que dentro de un procedimiento el valor de una variable se conserve deberás definirla poniendo **Static** en lugar de **Dim**.

10. Modifica el código de nuestra aplicación para que quede de la siguiente forma:

```
Private Sub Boton1_Click()
   Static Contador As Integer
   Contador = Contador + 1
   Valor.Caption = Contador
End Sub
```

```
Private Sub Boton2_Click()
   Dim Contador As Integer
   Contador = Contador + 1
   Valor.Caption = Contador
End Sub
```

Lo que pretendemos con este ejemplo es que veas como utilizando una variable definida como **Static** se puede mantener el valor dentro de un procedimiento, mientras que la misma variable definida como **Dim** en otro procedimiento actúa completamente diferente.

11. Realiza una ejecución de prueba.

*12. Pulsa repetidamente el primer **botón**.*

Aunque cada vez que pulsamos en botón estamos entrando en el procedimiento la variable guarda el valor y se le sigue sumando **1** gracias a la línea **Contador = Contador + 1**.

*13. Pulsa repetidamente el segundo **botón**.*

Observa como cada vez que se entra en este botón el valor vuelve a ser el mismo, ya que no se guarda el valor de las veces anteriores.

*14. Vuelve a pulsar el primer **botón**.*

El valor que teníamos almacenados en este procedimiento vuelve a surgir.

15. Detén la ejecución.

Observa bien las diferencias entre estos dos tipos de asignación de variables.

Variables útiles en todos los procedimientos

Si lo que nosotros deseamos es poder utilizar una variable en cualquiera de los procedimientos que forman parte de un módulo (formulario) deberemos definir dicha variable en el apartado **General - Declaraciones**.

En un principio cuando queramos definir variables de este tipo lo haremos de la siguiente forma:
Private [Nombre Variable] As [Tipo Variable].

16. Quita la declaración de las variables que hemos hecho anteriormente en cada uno de los botones y declara una sola variable para todo el formulario.

Constantes

Una constante posee las mismas características que una variable, ya que se puede definir en los mismo sitios, pero no se puede modificar su contenido.

Si deseamos utilizar la constante solo dentro de un procedimiento la definiremos allí, sin que esta puedas ser usada por otro procedimiento. Si lo que deseamos es que se pueda utilizar en cualquiera de los procedimientos definidos dentro de un formulario, deberemos definirla en el apartado **General - Declaraciones**.

¿Cómo se declara una contante?

Tanto dentro de un procedimiento como en **General - Declaraciones** las constantes se pueden declarar de la misma forma. En un principio existen dos formas diferentes de definir las constantes: indicando que tipo de valor se guardará en su interior o sin indicarlo, con lo que **Visual Basic** adaptará el contenido de la constante a las utilidades que nosotros le demos. Este tipo de dato, que se adapta a todo, se le llama **Variant**. Con este tipo de datos tenemos la ventaja que no nos tenemos que preocupar de que tipo es el dato que guardaremos dentro de la constante, pero tiene el pequeño inconveniente que ocupa un poco más de memoria que muchos otros tipos de datos que veremos más adelante.

Definición de la constante sin indicar el tipo de dato:
Const [Nombre Constante] = [Valor Constante]

Definición de la constante indicando el tipo de dato:
Const [Nombre Constante] As [Tipo de dato] = [Valor Constante]

. Práctica 4

Vamos a imaginar que queremos realizar una aplicación en la que partiendo de un número inicial de alumnos, cada vez que pulsemos un botón el número de alumnos aumente en 1.

1. Borra las líneas de código que hemos escrito en las prácticas anteriores y escribe el siguiente código allí donde corresponda. (Ten presente no estamos utilizando el segundo botón).

```
Option Explicit
Public Contador As Integer
Const Alumnos = 45
```

```
Private Sub Boton1_Click()
```

```
    Contador = Contador + 1
    Valor.Caption = Contador + Alumnos
End Sub
```

2. Realiza una ejecución de prueba.

3. Finaliza dicha aplicación.

Podríamos pensar que no hace falta crear una constante llamada **Alumnos** donde introdujéramos el número de alumnos que tenemos. Pero piensa que una constante es de suma utilidad en el momento que estamos realizando una gran aplicación en la que surge muchas veces una cantidad con la que tenemos que trabajar.

Ejemplo: imagina que tienes una aplicación con cientos de líneas en la que calculas el promedio de notas de la clase, el promedio de faltas en un trimestre, etc. Bien, pues en todos estos cálculos necesitas saber el número de alumnos que tienes. Si utilizaras este misma aplicación otros años deberías cambiar el número de alumnos. Entonces tendrías que buscar línea a línea allí donde realizas dichos cálculos, para cambiar el número de alumnos. En cambio si utilizas una constante, con solo cambiar el valor de la constante, todos los cambio ya están hechos.

*4. Modifica el valor de la constante **Alumnos**.*

5. Realiza otra ejecución del programa.

Observa que funciona exactamente igual, pero con valores diferentes.

6. Detén la ejecución del programa.

En prácticas posteriores trabajaremos con constantes.

Matrices

Como ya hemos explicado en la introducción las matrices nos permiten trabajar con un grupo de datos, los cuales se almacenan bajo un mismo nombre, pero se diferencian unos de otros por el lugar que ocupan, a este lugar le llamamos **Indice**.

Un peligro que deberemos tener siempre presente al trabajar con matrices, es no hacer referencia NUNCA a un índice que no esté definido. Con esto quiero decir que siempre debemos tener presente el tamaño de nuestra matriz. De esta forma si nosotros hemos definido una matriz de una dimensión con 5 elementos y hacemos referencia al elemento 10, **Visual Basic** nos dará un error de desbordamiento. Él a intentado acceder a un lugar que está fuera del tamaño al tamaño real de la matriz.

Como definir una matriz

Podemos decir que una matriz se define exactamente igual que una variable y una constante, pero con la diferencia que deberemos especificar el tamaño que deseamos que tenga. Esto lo hacemos de la siguiente forma, según si la matriz es de una o más dimensiones:

Matriz de una dimensión

Dim [Nombre Matriz] ([Tamaño]) As [Tipo de datos]

Tamaño: aquí definiremos, siempre entre paréntesis, el tamaño de nuestra matriz. Debemos recordar que si nosotros definimos una matriz de 5 elementos, el primer

elemento está ocupando la posición 0 y el último la 5, por lo tanto en realidad estamos trabajando con 6 elementos.

Si nosotros deseamos empezar a trabajar desde un número determinado de índice hasta otro deberemos definir la matriz de la siguiente manera:

Dim [Nombre Matriz] ([Índice inicial] To [Índice final]) As [Tipo de datos]

Una matriz definida de esta forma tendrá como primer índice el Índice inicial y como último el Índice final.

Matriz de más dimensiones

Dim [Nombre Matriz] ([Tamaño fila], [Tamaño columna]) As [Tipo de datos]

Una matriz de más de una dimensión podemos pensar que es como una cuadricula en la que necesitaremos dos o más índices para hacer referencia a alguno de los objetos que tenemos en su interior.

Tamaño fila: aquí definiremos el número de filas que deseamos tenga nuestra matriz.

Tamaño columna: aquí definiremos el número de columnas que deseamos tenga nuestra matriz.

Vamos ha hacer una representación gráfica de una matriz con **5 filas** y **10 columnas**:

		Columnas								
Filas	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1										
2										
3										
4										
5										

Observa como las **filas** están dispuestas **horizontalmente**, mientras que las **columnas** lo hacen **verticalmente**.

Así de esta forma, en el momento en el que nosotros queremos hacer referencia a uno de los elementos introducidos en la matriz, deberemos indicar el número de **fila** y el de **columna** que ocupa dicho elemento.

Al igual que en las tablas de una sola dimensión podemos especificar donde queremos que empiece el índice de nuestra matriz, pero en este caso tendremos que especificar cada uno de los índices que utilizamos (fila y columna).

Esta podría ser la definición de una matriz de dos dimensiones definiendo tanto el inicio como el final del índice.

Dim [Nombre Matriz] ([Índice inicial fila] To [Índice final fila], [Índice inicial columna] To [Índice final columna]) As [Tipo de datos]

Asignar valores

En este apartado vamos a ver como podemos introducir valores tanto en variables como en matrices.

Variables

Para la asignación de valores tenemos que pensar, siempre, el tipo de datos que podemos almacenar dentro de estas. Si en algún momento asignamos algún tipo de dato diferente al que hemos definido al crear la variable se producirá un error.

Para almacenar un valor en una variable solo deberemos escribir una instrucción como esta:

[Nombre Variable] = [Valor]

Valor: será el valor que queremos almacenar en nuestra variable.

Recuerda que si no está definida y no tienes la opción **Option Explicit, Visual Basic** definirá automáticamente la variable como tipo **Variant**.

Podemos decir, para facilitar el entendimiento de la asignación de valores en una variable, que esta se realiza de derecha hacia izquierda.

Matrices

Para asignar un valor a una posición de una matriz, lo haremos exactamente igual que en la asignación de valores en una variables, pero con la diferencia que aquí tenemos que especificar el índice (posición) donde deseamos se almacene el valor.

En el caso de una matriz de una sola dimensión lo deberemos hacer de esta forma:

[Nombre Matriz] ([Indice]) = [Valor]

En caso de una matriz de dos dimensiones lo deberemos hacer de esta otra forma:

[Nombre Matriz] ([Indice fila], [Indice columna]) = [Valor]

En el caso de las matrices deberás tener mucho cuidado en no asignar valores a índices que están fuera de la tabla.

Más adelante, cuando veamos las estructuras de repetición, veremos algunas formas de poder movernos por toda la matriz de forma sencilla para poder realizar cálculos, ordenaciones de datos y búsquedas de datos en una matriz.

Matrices de controles

En este capitulo vamos ha ver un sistema con el cual nos podemos ahorrar algunas líneas de código. Sobre todo en el momento en que tenemos muchos objetos que son exactamente iguales y queremos que todos actúen de una misma forma.

Esto lo vamos a explicar mejor con un pequeño ejemplo.

. Práctica 5

Vamos a realizar una práctica en la que tendremos 5 botones, cada uno con una vocal. Nosotros lo que queremos es que cada vez que el usuario pulse un botón se acumulen las letras en un **Label**. Por ejemplo si el usuario pulsa el botón con la **A**, después la **O** y por último el botón de la **E**. En el **Label** deberá aparecer la cadena **AOE**.

Es un ejemplo sencillo pero fácil para comprender el funcionamiento de las matrices de controles.

1. Abre un nuevo proyecto.

*2. Inserta un **Label**, ponle como nombre **Cadena**, borra el contenido de dicha etiqueta.*

*3. Inserta un **botón**. Cambia su nombre, poniéndole **Letra**.*

*4. Escribe en su interior la letra **A**.*

5. Ajusta el tamaño del botón, el tamaño de la letra, su apariencia, etc. realiza los cambios como desees.

Ahora vamos a pasar a copiar este elemento 4 veces más para tener los demás botones, los cuales representarán al resto de vocales.

*6. Pulsa un clic sobre el **botón** que tenemos en nuestro formulario para seleccionarlo.*

*7. Accede a la opción **Copiar** dentro del menú **Edición**.*

Aparentemente no ha pasado nada. Pero el ordenador ahora sabe que deseamos copiar este objeto.

*8. Vuelve a acceder al menú **Edición**, pero esta vez selecciona la opción **Pegar**.*

Acto seguido te aparecerá un cuadro de diálogo en el que te avisa que ya existe un objeto que tiene el mismo nombre y si deseas crear una matriz de controles.

9. Contesta afirmativamente.

*10. Repite la acción de copiar hasta que tengas **5 botones**.*

Las copias de los botones irán apareciendo en la esquina superior izquierda de nuestro formulario.

Observa que en el momento que ya copias uno de los botones, contestado afirmativamente a la creación de una matriz de controles, el resto de copias las realiza sin hacerte ningún tipo de pregunta.

11. Sitúa uno debajo del otro todos los botones que hemos creado.

*12. Selecciona el primero de los botones y observa la propiedad **(Nombre)**.*

Podrás observar como el nombre de dicho botón ya no es **Letra**, sino que pasa a ser **Letra(0)**. Este número, que aparece entre paréntesis, es un índice el cual nos sirve para distinguir cada uno de los botones que hemos creado. Nuestra matriz de botones irá desde el botón con índice **0** al que tiene el índice **4**. En total **5** elementos.

A partir de este momento si queremos modificar la propiedad de uno de los botones en concreto, deberemos especificar el número de índice de dicho botón. De este forma podremos utilizar estructuras de repetición para modificar las propiedades de muchos objetos en pocas líneas. Más adelante veremos ejemplos en los que utilizaremos está característica.

Otra característica que tienen estas estructuras de datos es que todo el código que escribamos dentro de uno de ellos afectará igualmente a cualquiera de las copias.

13. Pulsa doble clic en cualquiera de los botones.

Observa que en el momento de entrar en un evento, en la línea de código donde se especifica en que tipo de evento nos encontramos, aparecen las palabras **(Index As Integer)**. Este es el índice de los elementos que hemos insertado.

14. Escribe la siguiente instrucción:

```
Cadena.Caption = Cadena.Caption & Letra(Index).Caption
```

Esta instrucción lo que nos hace es "sumar" el contenido actual del **Label** llamado **Cadena** con el **Caption** del botón que hemos pulsado.

Antes de realizar una ejecución de prueba deberemos hacer una última modificación.

Si te fijas todos los botones tienen como propiedad **Caption** una **A**. Vamos a modificar cada uno de los botones para que contengan las diferentes vocales.

*15. Accede a cada uno de los botones y modifica la propiedad **Caption** escribiendo dentro de cada uno las diferentes vocales. No hace falta que estén en orden según el índice de cada botón.*

16. Realiza una ejecución de prueba.

*17. Pulsa los diferentes botones y observa como se van añadiendo las diferentes letras en nuestro **Label**.*

18. Detén la ejecución y graba nuestra pequeña práctica.

Fin lección 4