1.− Introducción 2

2.− Pasos para realizar un programa 2

3.− Primer contacto 2

3.1 Entorno integrado de desarrollo 2

3.2 Cuadro de herramientas 3

3.3 Agregar controles a un cuadro de herramientas 3

3.4 Añadir paginas al cuadro de herramientas 3

3.5 Explorador de proyecto 3

Controles básicos 9−12

Variables y constantes 13

Declaración de variables 13

Condiciones y bucles 15

Matrices y tipos 15−16

Procedimientos y tipos 16−17

Controles 18

Interfaz de documento múltiple 19

Cuadros de dialogo 19−20

Bases de datos 21−23

Control Dbgrid 23

Metodología de la programación 24−25VISUAL BASIC 6.0

Introducción

El lenguaje de programación BASIC nació en 1964 como una herramienta destinada a principiantes buscando una forma sencilla de realizar programas. La evolución del BASIC en los años 70 fue escasa dado el altísimo auge que tomaron en aquella época programas de alto nivel como FORTRAN y COBOL. En 1978 se definió una norma para unificar los basics existentes creándose la normativa BASIC standard

Características generales

VISUAL BASIC es una herramienta de diseño de aplicaciones para Windows en las que estas se desarrollan

normalmente a partir del diseño de una interfaz grafica. En una aplicación VISUAL BASIC el programa esta formado por una parte de código puro y otras partes asociadas a los objetos que forman la Interfaz Grafica

Pasos a la hora de crear un programa

- Creación de una interfaz de usuario: Esta interfaz será la principal vía de comunicación hombre maquina tanto para la salida de datos como para entrada será necesario de una ventana (la ventana de formularios a la que iremos añadiendo los controles necesarios).
- Definición de las propiedades de los controles objeto que hayamos colocado en este formulario estas propiedades determinan la forma estadística de los controles es decir como son los controles y para que sirven.
- Generación del código asociado a los eventos que ocurran a estos objetos a la respuesta a estos eventos (clic, doble clic, una tecla pulsada) le llamamos procedimientos y deberá generarse de acuerdo a las necesidades del programa.
- Generación del código del programa: Un programa puede hacerse solo con la programación de los distintos procedimientos que acompañan a cada objeto sin embargo VISUAL BASIC ofrece la posibilidad de establecer un código de programa separado de estos eventos. Este código puede introducirse en unos bloques llamados módulos, en otros bloques llamados funciones y en otros bloques llamados procedimientos

Primer contacto

1.− Entorno integrado de desarrollo

- El cuadro de herramientas contiene los iconos que vamos a usar para definir los controles que formaran parte de nuestros proyectos
- La ventana de propiedades contiene la información sobre las propiedades del control activo
- El explorador de proyectos contiene información sobre los formularios y los módulos de los que conste el proyecto

2.− El cuadro de herramientas

El cuadro de herramientas es una de las principales utilidades de VISUAL BASIC ya que se usa para agregar controles a elementos contenedores tales como los formularios o frames mientras nos encontramos en el modo diseño.

3.− Agregar controles a un cuadro de herramientas

Aunque en principio el cuadro de herramientas contenga solo los controles por defecto de VISUAL BASIC podemos añadir mas controles al cuadro para usarlo en nuestras aplicaciones

4.− Añadir paginas al cuadro de herramientas

Inicialmente el cuadro de herramientas tiene una sola pagina general que contiene los controles mas habituales. Al ir añadiendo mas controles aparecen en la pagina activa normalmente general aunque podemos añadir mas paginas y controles a estas. Para añadir una pagina al cuadro de herramientas debemos abrir el menú emergente del cuadro de herramientas y seleccionar la opción agregar ficha. Daremos nombre a la nueva pagina y esta aparecerá en forma de botón en la parte inferior del cuadro

5.− Exploración de proyecto

Un proyecto en VISUAL BASIC es un conjunto de ficheros que contienen el código y los datos necesarios

para construir una aplicación ejecutable una librería de enlace dinámico o un control activex en este curso nos vamos a centrar en el desarrollo de proyectos ejecutables. Los distintos tipos de componentes son: De todos estos tipos de componentes utilizaremos de momento los formularios y los módulos

Si hacemos clic sobre un archivo situado en el explorador de proyectos nos saldrá un menú que contendrá las siguientes opciones:

- Ver objeto
- Ver código
- Propiedades: muestra la ventana de propiedades y los datos del objeto seleccionado
- Agregar
    - ♦ Formulario: Contiene la definición de un formulario de la aplicación describiendo sus características, los controles que contiene, declaraciones locales de variables y constantes y los procedimientos y funciones necesarias para controlar su comportamiento. Se guardan en ficheros de extensión ***.frm**
    - ♦ Formulario MDI: Es un tipo especial de formulario que puede contener a otros formularios . Se guardan en ficheros de extensión ***.frm**
    - ♦ Modulo: En ellos se escriben los procedimientos, funciones y declaraciones que pueden ser accedidas desde otros módulos de la aplicación. Se guardan en ficheros de extensión ***.bas**
    - ♦ Modulo de clase: Se utilizan para describir una nueva clase que nos servirá para definir nuevos objetos en nuestra aplicación. Son la base de la programación orientada al objeto. Se guardan en ficheros de extensión ***.cls**
    - ♦ Control de usuario: En estos ficheros se guardan las descripciones de los controles activex diseñados por el usuario asi como sus procedimientos, funciones y variables. Se guardan en ficheros de extensión ***.cll**
    - ♦ Documento de usuario: Los proyectos de tipo documento activex son descripciones del equivalente a formulario de las aplicaciones EXE estándar. Se guardan en ficheros de extensión ***.dob**
    - ♦ Pagina de propiedades: Crea un formulario para ser utilizado en la edición de las propiedades de un control. Se guardan en ficheros de extensión ***.pag**
- Guardar Form1
- Guardar Form1 como
- Quitar Form1
- Imprimir
- Acoplable
- Ocultar

La opción establecer como inicial que hace que el proyecto seleccionado sea el que se ejecute en primer lugar en caso de que tengamos mas de un proyecto abierto. Todas estas opciones se encuentran en el menú proyectos y en el menú archivo.

La ventana de propiedades

El aspecto y muchas de las características de los formularios y controles que podemos encontrar en una aplicación vienen definidos por sus propiedades se modifican en tiempo de diseño aunque algunos puede modificarse en tiempos de ejecución

Formulario

El primer objeto VISUAL BASIC con el que encontramos es el formulario. De hecho cada vez que iniciamos VISUAL BASIC nos presenta en pantalla un nuevo formulario que tiene por defecto el nombre form1.

Propiedades principales:

- Name: Define el objeto durante la ejecución del programa se introduce un tiempo de diseño y no se puede variar
- Caption Titulo: Es el texto que aparece en la barra de titulo cada vez que aparezca en pantalla este formulario
- Control box: Cuadro de control del formulario. **Valor por defecto TRUE propiedad boleana que admite los valores de TRUE o FALSE**
- Max button: Controles maximizar, minimizar. **Valor por defecto TRUE** botones de maximizar y minimizar este formulario son igualmente **propiedades boleanas que admiten el valor TRUE o FALSE** *si están en TRUE aparecerán los controles correspondientes y si están en FALSE no aparecerán dichos controles deben configurarse de una u otra forma*
- Border Style Tipo de borde: Define el tipo de borde que tendrá el formulario durante la ejecución no se puede cambiar el tiempo de ejecución

Valores

0	None	El formulario tiene borde alrededor
1	Fixed Simple	Borde simple no redimensionadle
2	Sizable	Borde grueso redimensionadle
3	Fixed Dialog	
5	Fixed tollwindow	Borde no redimensionadle con área de titulo reducida

- Appearence Apariencia. **Valor por defecto 3D.** Valores 0=flat, plano/ 1=3D
- Autoredraw :**Valor por defecto FALSE propiedad boleana esta propiedad estando en TRUE permite actualizar el contenido del formulario y de sus controles incluso cuando no estén visibles**
- Backcolor Color del fondo
- Drawmode: Establece un valor que determina el aspecto de salida de un metodo grafico o el aspecto de un control Shape o Line
- Drawstyle: **Valor por defecto 0 establece el estilo de linea de salida de metodos gráficos.**

Valores

0	Linea continua
1	Rayas
2	Puntos
3	Raya−punto
4	Raya−punto−punto
5	Transparente
6	Continuo interior

- Enabled Activado. **Valor por defecto TRUE propiedad boleana si esta en TRUE el formulario esta activado y se puede interactuar con el. Si esta en FALSE se desactiva el formulario impidiendo de esta forma que se pueda trabajar con el**
- Forecolor. **Valor por defecto negro** establece el color del primer plano del formulario, es el color que tendrán las letras.
- Fillstyle Tipo de relleno. **Valor por defecto 2** establece el modo de relleno de controles Shape o figuras

Valores

0	Continuo
1	Transparente
2	Linea horizontal
3	Linea vertical
4	Diagonal hacia arriba
5	Diagonal hacia abajo
6	Cruzado
7	Diagonal cruzada

- Fill colour color de relleno: Establece el color del relleno contemplado en fillstyle
- Font Tipo de letra. **Valor por defecto el determinado en la personalización especifica el tipo y el tamaño de letra que se usara en el formulario.**
- Height Altura. **Valor por defecto no existe. Define la altura del formulario.**
- Icon Icono: Podemos asignar a esta propiedad un icono que sera visualizado en la esquina superior
- MDIchild: **Valor por defecto false.** Establece que este formulario es un formulario hijo dentro de un formulario MDI. No se puede cambiar en tiempo de ejecución. Es una propiedad boleana
- Mouselcom: **Valor por defecto ninguno**. Establece un icono personalizado para el puntero del ratón . Cuando este esta encima del formulario. Este icono puede ser un bit−map de los existentes en el directorio icons del VISUAL BASIC o cualquiera que tengamos. Si se pone 99 como valor de la propiedad Mouse pointer cada vez que el puntero del ratón pase por encima del formulario cambiara su forma y adoptara el del icono elegido.
- Mouse pointer: **Valor por defecto flecha**. Determina la forma del puntero del ratón cuando se coloca encima del formulario. Puede elegirse uno de los punteros preseleccionados (15 en total) o el personalizado visto en la propiedad anterior. Para elegir este icono personalizado debemos poner en esta propiedad el valor 99
- Picture Grafico: **Valor por defecto no existe**. Mediante esta propiedad podemos poner un grafico como fondo del formulario. El grafico puede ser un BIT−MAP o un fichero ICO
- TOP posición del borde superior **Valor por defecto no existe.** Esta propiedad establece el borde superior del formulario. Normalmente no se introduce como un valor numérico, sino que lo toma automáticamente de la posición que tenga el formulario.
- Visible: **Valor por defecto TRUE propiedad boleana asignada.** Visible colocado en valor TRUE hace que el formulario sea visible y asignándole el valor FALSE no es visible. Este valor se puede cambiar durante el tiempo de ejecución para ocultar y hacer visible el formulario.
- Width Ancho: **Valor por defecto no existe.** Define la anchura del formulario. Normalmente no se introduce como un valor numérico, sino que lo toma automáticamente del tamaño que tenga el formulario durante el tiempo de diseño.
- Window State: Establece el estado en el que aparecerá el formulario se activa y presenta en pantalla. Admite tres opciones:
    - ♦ 0 Normal: El formulario recupera su posición y tamaño que tenia en el tiempo de diseño.
    - ♦ 1 Minimizado
    - ♦ 2 Maximizado

Eventos

Activate	Activado
Clic	Click
Dblclick	Doble click
Drogdrap	Arrastrar y soltar

Drogover	Arrastrar por encima
Desactivate	Desactivado
Gotfocus	Obtener foco
Paint	Pintar
Key press	Pulsar una tecla
Key up	Soltar una tecla
Key Down	Mantener pulsada una tecla
Link Open	Romper enlace
Link error	Error enlace
Load	Cargar formulario
Lost Focus	Perder foco
Link execute	Ejecución enlace datos
Mouse move	Mover el ratón
Mouse Down	Mantener pulsada una tecla del ratón
Query unload	Confirmación de descarga
Un load	Descarga del formulario
Resize	Cambio de tamaño

Enviar mensajes por pantalla (msgbox)

El sistema por excelencia para enviar mensajes por pantalla del entorno de Windows es mediante msgbox a lo largo de la creación de nuestros aplicaciones msgbox será una de las funciones que mas usemos ya sea para enviar mensajes al usuario o para realizar comprobaciones en nuestro código

Sintaxis

Msgbox{[Mensajeð

Donde mensaje será el mensaje que queremos que nos aparezca en la pantalla.. La segunda opción botones nos sirve para especificar que botones deben aparecer en el cuadro de mensaje mediante la suma de valores que especifican el numero y el tipo de botones que se pretenden mostrar

Los valores para los botones son:

Constantes	Valor	Descripción
Vbokonly	0	Muestra el botón aceptar
Vbokcancel	1	Muestra los botones aceptar y cancelar
Vbabortretryignore	2	Muestra los botones anular reintentar ignorar
Vbyesnocancel	3	Muestra los botones si no cancelar
Vbyesno	4	Muestra los botones si y no
Vbretrycancel	5	Muestra los botones reintentar y cancelar
Vbcritical	16	Muestra el mensaje de mensaje critico
Vbquestion	32	

		Muestra el icono de pregunta advertencia
Vbexclamarion	64	Muestra el icono de mensaje informativo
Vbdefaultbotton1	0	Muestra que el primer botón es predeterminado
Vbdefaultbotton2	256	Muestra que el segundo botón es predeterminado
Vbdefault3	512	Muestra que el tercer botón es predeterminado
Vbdefault4	768	Muestra que el cuarto boton es predeterminado
Vbaplicationmodal	0	Aplicación modal el usuario debe responder al cuadro de mensaje antes de poder seguir trabajando en la aplicación actual
Vbsystemmodal	4096	Sistema modal, se suspenden todas las aplicaciones hasta que el usuario responda al cuadro de mensaje

Introducción de datos mediante inputbox

Mediante la funcion inputbox podemos hacer que el programa envie un cuadro de mensaje al usuario y esperar a que este inserte un texto o valor que al pulsar sobre el boton aceotar lo almacena en una variable del tipo STRING o cadena que ya veremos mas adelante

Sintaxis

inputbox{[mensaje] [titulo][default)) [yposxpos])

[mensaje} Mensaje que nos aparecerá en pantalla

[Titulo} Titulo que nos aparecerá en pantalla

[Default]: Este valor nos aparecerá por defecto en el cuadro de mensaje si se trata de un texto deberá ir entre comillas. Esta opcion no es requerida

Controles básicos

Nada mas empezar a programar en Visual BASIC debemos tener constancia de los controles básicos de este lenguaje de programación para así poder realizar las primeras aplicaciones y ejemplos

Command button (botón de comando): El botón de comandos puede usarse para la entrada de datos a través del ratón

Propiedades

- Name nombre: Es el nombre que define al objeto durante la ejecución del programa.
- Caption Titulo: Determina la forma del botón
- Backcolor: Color de fondo

- Cancel: Establece un valor que indica si un botón de comando es el botón cancelar de un formulario, es una propiedad boleana y admite los valores TRUE/ FALSE.
- Default: Tiene la misma función que CANCEL pero para INTRO.
- Dragmode: Establece un valor que determina si se usa el modo de arrastrar manual o el automático en una operación de arrastrar y colocar. Los valores posibles son:

0 **Manual Predeterminado**

1 **Automático**

- Enabled **Habilitado**: propiedad boleana que habilita o deshabilita el botón . Cuando esta deshabilitado (Enabled=false), el botón no tiene efecto y su apariencia varia, puede variarse en tiempo de ejecución
- HelpcontextID: Establece un numero de contexto asociado para este control. Este numero se aplica para determinar la ayuda interactiva. Puede tener los siguientes valores:

0 **No se especifica numero de contexto**

>0 **Un entero que especifica un contexto valido**

- Index Índice: En caso de que se tengan varios botones que realicen una función similar. Puede utilizarse un array o matriz con estos botones de comando todos tendrán el mismo nombre y se diferenciaran por un índice . Esta propiedad index toma el numero de ese índice.
- Left: Posición del botón
- Mouselcom: Icono para el puntero del ratón. Determina el icono que presenta al puntero del ratón cuando pasa por encima del botón. Cuando se especifica en la propiedad Mousepointer que el puntero del ratón es el definido por el usuario (custom)
- Mousepointer: Puntero del ratón
- Tblindex: nº de orden para el uso del tabulador cuando disponemos de varios controles en un mismo formulario solo uno de ellos tiene el foco. Esta expresión de tener foco significa que esa tecla ENTER haría el mismo efecto que hacer clic con el ratón en ese control. Si usamos el teclado y queremos pasar de un control a otro debemos pulsar repetidas veces la tecla tabulador . El foco ira cambiando de control cada vez que pulsemos la tecla tabulador. Esta propiedad Tblindex marca el orden que seguirá el foco a través de cada control

- Tabstop: Sale del control de la tecla tab. Propiedad boleana cuando esta propiedad esta en FALSE el botón no tomara el foco cuando se pulsa la tecla tabulador. Puede cambiarse en tiempo de ejecución

- TOP: Define la parte superior del control

- Visible: Propiedad boleana. Si es true el botón se hace visible. Si es FALSE el boton no se ve. Puede cambiarse en tiempo de ejecución.

- WhatthishelpID: Devuelve o establece un numero de contexto asociado a un objeto. Se utiliza para dotar a las aplicaciones de ayuda interactiva con el menú emergente de ayuda de Windows 95 o 98

- Width Ancho: Define el ancho del boton.

Métodos de command button

- Clic
- Key down
- Mouse down

- Dragdrop
- Key press
- Dragover
- Key up
- Mouse Move
- Got focus
- Lost focus
- Mouse up

Link items/ link mode/ link time out/ link topic

Estas propiedades establecen la forma en que debe llevarse a cabo una conexión DDE con otra aplicación.

- Locked: Establece si el texto se puede editar, es decir cambiar cuando esta propiedad se pone en TRUE el texto existente en la caja puede resaltarse con el ratón, pero no puede variarse con el teclado tecleando un nuevo texto. Se puede cambiar por un programa cambiando la propiedad text. Si esta en FALSE puede cambiarse el texto mediante el teclado
- Maxlenght: Indica si se establece la longitud máxima del texto

Paswordchar

Nota: Esta propiedad se ignora cuando la propiedad multiline esta en True

- Scrool bars: Cuando la propiedad multiline esta en TRUE se pueden colocar barras de desplazamiento estas nos permiten tener una caja de texto de tamaño reducido y poder leer en ella un texto mayor que la propia caja . Esta propiedad puede tener los siguientes valores:

0 **No salen barras**

1 **Barras de desplazamiento horizontal**

3 **Ambas barras**

Métodos del textbox

- Clic
- Change
- Dbl clic
- Dragdrop
- Dragover
- Gotfocus
- Keydown
- Keypress
- Keyup
- Linkclose

Label

Una etiqueta es un control que nos permite presentar un texto. La etiqueta debe usarse en aquellos casos en que exista una información estática o dinámica

- Autosize: Propiedad boleana si se pone en TRUE el tamaño de la etiqueta se ajusta al texto que

contiene

- Backcolor
- Backstyle: Opaco o transparente. Cuando se selecciona transparente solamente se ve el texto de la etiqueta. Cuando se selecciona opaco este texto se ve sobre un fondo gris
- Use mneumanic: Devuelve o establece un valor que indica si al incluir el signo & en el texto de la propiedad CAPTION del control LABEL se define una tecla de acceso. Es una propiedad boleana. Los valores que pueden tomar son TRUE/FALSE
    - ♦ TRUE predeterminado los caracteres & que aparezcan en el texto de la propiedad CAPTION definen el carácter siguiente como tecla de acceso. El signo & no aparece en la interfaz del control label.
    - ♦ FALSE: Los caracteres & que aparezcan en el texto de la propiedad caption aparecen como texto en el interfaz del control label
- Wordwrap: Devuelve o establece un valor que indica si un control label con el valor TRUE en su propiedad autosize se expande en vertical u horizontalmente para adaptarse al texto especificado en su propiedad caption . Es una propiedad boleana. Esta propiedad puede cambiarse en tiempo de ejecución .
    - ♦ TRUE: El control label se expande o contrae horizontal o verticalmente para adaptarse al texto y al tamaño de la fuente. Contempla para la expansión horizontal la colocación de los espacios en blanco
    - ♦ FALSE predeterminado: EL texto no se ajusta a la siguiente linea el control label se expande o contrae horizontalmente para adaptarse a la longitud del texto y verticalmente para adaptarse al tamaño de la fuente.

Métodos del label

- Clic
- Drag over
- Change
- Link close
- Dblclic
- Link error
- Drag drop
- Link notify

Check button y option (botones de eleccion y opcion)

El control checkbox o casilla de verificación permite elegir una opción activada/ desactivada, true/ false que el usuario puede establecer o anular haciendo clic. Cada casilla de verificación es independiente de las demás que pueden existir en el formulario pudiendo tomar cada una de ellas el valor true o false, a voluntad del operador

Un control option button muestra una opción que se puede activar o desactivar pero con dependencia del estado de otros controles option button que existen en el formulario

Propiedades

Alignment: Común a ambos controles

0 Left justify

1 Right justify

- Data field Data source: Propiedad de check button solamente. Establecen la base de datos y el campo donde están los datos (true, false) que llevaran la propiedad valué al igual que en los controles label y textbox esta propiedad nos permite visualizar los datos de una forma muy sencilla. En este checkbox solo permite presentar lógicamente los datos de los tipos boléanos
- Valué: Común a ambos controles

Variables

Cuando hablamos de variables nos referimos a posiciones de memoria referenciadas por un nombre donde podemos almacenar el valor de un dato que después podemos modificar o alterar según nos convenga. Esta serie la explicación teórica de lo que es una variable.

Cada vez que vamos a crear un programa necesitamos constantemente el uso de variables ya que necesitamos almacenar multitud de datos para realizar cálculos, contadores para bucles. De ahí la gran importancia de las variables a la hora de programar

Variables de datos

- Byte
- Boleana
- Integer
- long(entero largo)
- Single(coma flotante/precision simple)
- Double(coma flotante/precision doble)
- Currency(entero a escala)
- Date
- Object
- String(longitud variable)
- String(longitud fija)
- Variant(con números)
- Variant(con caracteres)
- Object

Declaración de variables

VISUAL BASIC es uno de los pocos lenguajes de programación que no necesita declarar las variables para poder usarlas este es un método poco recomendable ya que puede producir muchos errores al programa y al programador por ello es mejor la declaración de variables y para ello hacemos uso de la sentencia DIM que declara la variable y le asocia un tipo y reserva el espacio necesario en la memoria para almacenar datos.

DIM Edad as integer (numérico)

DIM Nombre as string (carácter

Nombre_de la variable=Valor

Si optamos por no declarar la variable solo nos vale con darle a esta un nombre y asignarle un valor las variables no declaradas por defecto son del tipo *variant*. Una de las cosas a tener en cuenta a la hora de declarar una variable es el ámbito que le vamos a dar y esto depende del lugar donde la definamos.

Declaración de variables multiples

Siempre que vayamos a declarar muchas variables podemos hacerlo todo en la misma línea separadas por comas y tener en cuenta una regla

Supongamos que declaramos tres variables enteras en la misma línea la sintaxis correcta es la siguiente:

DIM A as integer, B as integer, C as integer

Un error muy común es escribirlas de la siguiente manera

DIM A,B,C as integer

El error esta en que si declaramos de la segunda manera A se declara como una variable entera pero B y C toman el tipo variant ya que no se les ha especificado tipo alguno.

Variables globales:

En mas de un programa necesitamos poder usar variables que no solo estén disponibles en un procedimiento, modulo o formulario, sino que nos hará falta que las podamos usar en cualquier parte del programa, estas son las llamadas **variables de ámbito global**

Uso de option explicit

Como ya hemos comentado antes VISUAL BASIC no es necesario declarar las variables antes de usarlas pero esto nos puede llevar a múltiples errores. Los que optan por declarar las variables pueden utilizar la instrucción option explicit que obliga a declarar las variables antes de usarlas y no caemos en la tentación de no declararlas.

Siempre que queramos utilizar option explicit deberemos declararlo en main()

si queremos que tenga ámbito en todo el programa si no vamos a utilizar el procedimiento sub main()

Constantes

Las constantes se forman de un nombre y de un valor como si de una variable se tratase pero con la diferencia de que su valor no puede ser cambiado posteriormente. Su declaración es muy sencilla y solo se tiene que hacer uso de la sentencia const y asignarle un valor

Generar números aleatorios

En mas de una ocasión nos será necesario en muchos programas crear números aleatorios como puede ser para crear programas de lotería para esto contamos con las funciones Randomize y Rnd.

La primera funcion Randomize inicia el generador aleatorio tomando como dato de partida el numero. Devuelve el resultado en una variable llamada Rnd

Sintaxis: **Randomize (numero)**

Pero los números aleatorios generados de esta forma son siempre iguales, eso si dependiendo del nombre que se le introduzca como parámetro. Esta generación de números no produce números aleatorios y se repite el numero que nos ha creado el generador aleatorio por ello la mejor forma de crear números aleatorios es introducir un numero variable, con tiempo para que podamos usar la función Timer.

Sintaxis: **Randomize Timer**

Condiciones y Bucles

Condicion select case

Por ultimo tendremos la condición Select..... Case su uso se basa en evaluar una expresión que puede presentar varios casos asi se ejecutara un caso u otro dependiendo de cual de ellos coincida con el valor de la expresión

Bucle For Next

El bucle for next tambien llamado bucle por contador lo usamos para repetir un grupo de instrucciones un determinado numero de veces que nosotros asignaremos mediante una variable.

Sintaxis: **For valor_inicial To valor_final sep**

Instrucciones

Next

Para empezar declaramos una variable numérica ahora empezamos la sentencia for donde inicializamos numeros a 1

For n=1 to 10

La sentencia step es el valor que nos da el incremento que tomara el bucle que en nuestro caso es la variable números . Si damos step 0 el bucle no tendrá fin

Bucle do loop

El bucle loop consiste en repetir un grupo de instrucciones controlado por una condicion en vez de por un contador para controlar la condicion haremos uso de las sentencias while y until

Matrices y tipos

Hasta ahora conocemos lo que son las variables individuales con un solo valor pero supongamos que vamos a crear un programa en el que guardemos el nombre de los alumnos de una clase. Con los conocimientos que tenemos hasta ahora crearíamos treinta variables una por cada alumno pero esto seria un lió y un gasto importante de memoria importante así que para estos casos que vamos a trabajar con variables del mismo tipo y relacionadas entre si pero con diferentes valores haremos uso de las matrices o array en otros lenguajes

Dim alumnos (29) as string

Siempre que queramos saber cual es el índice mínimo y el máximo de una matriz contamos con las funciones lbound(índice mínimo) y vbound(índice máximo)

Matrices multidimensionales

Antes hemos visto matrices con un sola dimensión con una lista de valores pero habrá veces en que sea necesario que tengamos varias filas y columnas de valores para ello haremos uso de las matrices multidimensionales

Dim notas(1 to 30, 1 to 30) as integer

Matrices dinámicas

Cuando hablemos de matrices dinámicas estamos diciendo que podemos variar el numero de elementos de una matriz en tiempo de ejecución, es decir que si hemos declarado una matriz con cinco elementos mas adelante mediante la instrucción REDIM podemos variar los elementos

Redim[preserve]variable(índice) as tipo

Siempre que vayamos a declarar una matriz dinámica, la declaramos sin ningún elemento y después la dimensionamos al numero deseado

Dim alumnos() as string ` declaramos matriz dinámica

Redim alumnos(30) as string ` Asignamos 30 elementos

Tipos o estructuras

Los tipos definidos nos sirven para crear estructuras de datos en los que podemos agrupar múltiples variables del tipo que deseemos

Procedimientos y tipos

En VISUAL BASIC podemos definir procedimientos y funciones como en la mayoría de los lenguajes de programación. La necesidad de usar procedimientos y funciones ya la vimos en la primera lección por lo que nos queda ver los aspectos específicos de la sintaxis de VISUAL BASIC

Otro asunto a tratar es la terminología utilizada en VISUAL BASIC aquí se denomina procedimientos de forma general a lo que nosotros llamamos :

- Procedimientos
- Funciones
- Procedimientos de eventos
- Procedimientos de acceso a propiedades

Los procedimientos se clasifican :

- Procedimientos sub.: No devuelven ningún valor.
- Las cláusulas public/private definen el ámbito de alcance del procedimiento
- La cláusula **static**: **provoca que todas las variables locales sean estáticas**

Dentro de este tipo de procedimientos se incluyen también los procedimientos asociados a eventos

- Procedimientos función .

- Procedimientos property: Se utilizan para leer y modificar propiedades de los objetos. Como este estudio de la definición de objetos se sale del ámbito. Solo vamos a indicar que existen tres tipos de procedimientos.

- Property get: Sirven para leer el valor de una propiedad
- Let Utilizado para asignar un valor de tipo distinto al del objeto

- Set Sirve para asignar objetos a propiedades de tipo objeto

Pasar parámetros a una función

Antes dijimos que a una función se le pasan uno o varios parámetros con los que podemos realizar alguna operación . Al declarar la función hay que decirle el nombre de los parámetros que se le van a pasar, de que tipo son (string, integer, boolean)y como se le van a pasar *byval/byref*

Function nombre[optional]byval/byref]

Optional significa que el argumento es opcional y lo podemos omitir en la llamada al procedimiento . Si declaramos un argumento como opcional los demás han de serlo también. En un argumento opcional podemos definir un valor por defecto para en los casos en los que la llamada no se pase ningún valor si el valor de la variable es object solo se admiten el valor nothing. El valor por defecto solo se puede utilizar en los argumentos opcionales

- Byval: Significa que en la llamada al procedimiento se pasa el valor y no la referencia a la variable pasada como parámetro. Los cambios efectuados sobre el argumento no modifican el valor de la variable pasada como parámetro.
- Byref: Significa que la llamada al procedimiento se pasa la referencia a la variable pasada como parámetro. Los cambios efectuados sobre el argumento modifican el valor de la variable pasada como parámetro
- Paramarray: Solo se puede utilizar en el ultimo argumento indica que el argumento es una matriz opcional de elementos de tipo variant. De esta forma podemos pasar al procedimiento un numero variable de procedimientos

Mas de una vez nos será necesario salir de una función antes de que termine por ejemplo si se cumple una condición determinada. Se puede salir de la función mediante la sentencia exit function

Llamada a procedimientos

Tenemos varias formas de llamar a un procedimiento la primera de ellas es con la instrucción call

Call nombre del procedimiento(parametro1,parametro2)

CONTROLES

- Listbox: Permite ofrecer al usuario una serie de opciones para que elija VISUAL BASIC / añadirá barras de desplazamiento al cuadro de lista si esta es demasiado grande para ser vista toda a la vez
- Listcount: No puede modificarse directamente contiene el numero de elementos de una lista
- Sorted: Para mantener la lista alfabéticamente
- Additem: permite insertar una línea de texto en el cuadro losta
    - ♦ **Objeto.additem texto[índice]**
- Clear: Borra los elementos del cuadro lista
    - ♦ **Objeto.clear**
- Removeitem: Permite eliminar una linea del cuadro lista
    - ♦ **Objeto.removeitem(indice)**

Controles de disco

El control drivelistbox nos permite gestionar el acceso a las unidades instaladas en el equipo es decir disquetera/cdroom/disco duro

El control dirlistbox nos permite gestionar el acceso a los arboles de directorios de la unidad seleccionada

Formularios y menus

Cuando se incia una aplicación desarrollada en VISUAL BASIC se ejecuta en primer lugar un formulario o una funcion que se haya establecido como inicial. Si esta establecido que sea un formulario este se cargara automáticamente al iniciarse la aplicación y se oculta cuando se cierra pero si el objeto que se ha establecido como inicial es una funcion o si queremos mostrar otro formulario debemos saber como cargar, mostrar, ocultar y descargar una ventana.

Cuando se muestre una ventana si no esta cargada se hace automáticamente pero muchas veces necesitamos configurar el valor de una variable publica del formulario o una determinada propiedad de este o alguno de sus controles antes de mostrar el formulario. Para poder acceder a estos elementos necesitamos que este cargado y para ello debemos usar el siguiente procedimiento

Load nombre de formulario

Una vez cargado el formulario tenemos que mostrarlo un formulario se puede mostrar de dos formas modal y no modal. La primera de ellas obliga al usuario a trabajar con el formulario mostrado en forma modal no permitiendo cambiar a otro formulario de la aplicación esto es útil cuando queremos obligar al usuario a tomar una decisión antes de continuar trabajando. Por el contrario un formulario mostrado en forma no modal permite cambiar a otro formulario de la aplicación

Menús emergentes

Un menú emergente o un poupmenu es un menú que se despliega en cualquier parte del formulario. Es normal que en los procesadores de texto se obtenga un menú emergente pulsando el botón derecho del ratón .

Para crear un menú emergente es necesario tener en ese formulario un elemento llameémoslo emergente que en la casilla visible deberá estar desactivada y a su vez emergente deberá tener un menú desplegadle con los elementos que queramos que aparezcan en dicho menú. **Una vez hecho esto en el evento mousedown del formulario principal escribiremos esto:**

If button=vbrightbutton then poupmenu nombre

Interfaz de documento múltiple

Hasta este momento hemos estado trabajando con aplicaciones cuya interfaz es la SDI como podía ser la aplicación wordpad incluida en el Windows 98 pero ahora vamos a ver lo que son las aplicaciones MDI

Gracias a la interfaz MDI vamos a poder trabajar con múltiples formularios hijos dentro de un formulario padre o principal . El ejemplo mas común de la aplicación MDI es la aplicación Microsoft Word donde podemos trabajar con múltiples documentos abiertos a la vez. En cualquier aplicación con una interfaz MDI solo puede haber un formulario padre que será llamado formulario MDI y dentro de este podemos agregar tantos formularios hijos como queramos. Estas ventajas hijas tienen diferencias en su comportamiento respecto a las ventajas normales.

- Cuando maximizamos una ventana hija ocupa toda el área de la ventana principal y no todo el área de la pantalla.
- Cuando las minimizamos aparece su icono dentro del área de la ventana principal y no en la barra de tareas de Windows 95/98
- Los menús de los hijos activos tienen preferencia sobre los del formulario padre

Propiedades

. Autoshowchildren: Muestra los formularios hijos nada mas cargarlos

. Activeform: Mediante esta propiedad podemos conocer el formulario activo dentro de una aplicación de documentos múltiples

Variable=activeform.caption

Cuadros de dialogo

Para hacer uso de los cuadros de dialogo tendremos que agregar a nuestro proyecto el control Microsoft common dialog 6.0 . Una vez agregado podremos hacer uso de los cuadros de dialogo comunes de windows

- Abrir: commondialog1.showopen
- Guardar: commondialog1.showsave
- Filtros: commondialog1.filter=descripcion de fichero
- Filtros:commandialog1.showopen
- Filtros:commandialog1.showsave
- Seleccionar fuentes:commandialog1.showfont
- Seleccionar fuentes: flags=3
- Imprimir: Commandialog1.showprinter

La impresora elegida podemos conocerla llamando a la propiedad devicename de un objeto que aun no hemos visto el objeto printer. Este objeto es el que se encarga de pasar al administrador de impresión todos los trabajos de impresión que generan las aplicaciones de VISUAL Basic

Nombre de la impresora=printer.devicename

Para que la impresora quede como impresora por defecto de windows debera tener a true la propiedad printerdefault del cuadro de dialogo.

Apertura y cierre de ficheros

Open nombre for modo

- Nombre: Es el nombre del fichero que vamos a abrir puede contener la ruta del fichero.
- Acceso: Explica las operaciones permitidas en el fichero
- Read: Solo se permiten lecturas
- Write: Solo se permiten escrituras
- Readwrite: Se permiten ambas operaciones
- Bloqueo: Indica si otros procedimientos internos o aplicaciones pueden acceder al fichero mientras lo tenemos abierto y que accesos permiten
- Shared: Acceso total

Close#(numero),(#)numero de fichero

Instrucción Print

Print#numero de fichero, expresión

Al guardar los resultados de las expresiones estas se convierten en tipo cadena pero sin formato establecido.

Se traducen de acuerdo a la configuración del idioma que tenga el equipo donde se ejecute el programa

Función put: Escribe el contenido de una variable en un fichero en la posición que indiquemos

Put(#)nº de fichero, nº de registro, variable

EOF: Indica el final del fichero (end of file)

BOF: Indica el principio del fichero (bejín of file)

Loc: Devuelve la posición de lectura o escritura actual en un archivo abierto

Lof: Devuelve la longitud de un fichero.

BOF: Indica el principio del fichero (bejín of file)

Loc: Devuelve la posición de lectura o escritura actual en un archivo abierto

Lof: Devuelve la longitud de un fichero.

BASES DE DATOS

Hoy en día las empresas no paran de desarrollar software de control de datos. Nada mejor que esta lección para aprender el manejo de bases de datos en VISUAL Basic .

Introducción

Las bases de datos es uno de los puntos mas fuertes de VISUAL Basic debido a una fácil programación mediante el control data o código.

Una base de datos es algo mas simple que un almacenamiento de datos una base de datos debe proporcionar medios eficientes para buscar información, ordenarla según nuestras necesidades , combinar información de distintas tablas, sacar extractos, informes. En el trasfondo de todas estas operaciones se encuentra una serie de conceptos importantes

Conceptos:

- Índice
- Relación: Es el vinculo entre dos tablas que se realiza a través de un grupo de campos
- Consultas: Es el resultado de aplicar una restricción a una tabla para obtener un extracto, de realizar un cruce de datos entre varias tablas
- Las consultas se construyen utilizando el SQL que es un lenguaje de consulta a bases de datos estándar

En esta lección vamos a ver el manejo de bases de datos de dos formas. La primera mediante el control data una manera rápida y sencilla de manejo pero que nos privara de todo control de la base de datos. La segunda forma mediante el método dao (data Access object) algo mas compleja pero que a la larga nos será mas útil ya que podemos controlar todas las posibilidades que nos da la base de datos. Aunque sean métodos diferentes en algunos programas nos será necesario el uso de las dos

DAO

El modelo de objetos dao es un conjunto de clases organizadas jerárquicamente que nos facilita el acceso a bases de datos de diversos fabricantes de forma homogénea en realidad es la interfaz del motor de bases de datos jet que realiza el trabajo duro de acceso a las bases de datos

Imagen: Vamos a tratar clases útiles para acceder a bases de datos de Access (*.mdb) desde aplicaciones sencillas.

Antes de entrar en materia debemos agregar a nuestro proyecto la referencia Microsoft Dao 2.5/3.51 compatibility library

Abrir una base de datos

Para abrir una base de datos ya existente lo primero que tenemos que hacer es declarar una variable como un objeto database y después usaremos el método opendatabase para abrir la base de datos . Ahora solo nos será necesario poder manejar la base de datos para ello debemos aprender el uso del objeto Recordset.

Un objeto Recordset representa los registros de una base de datos o los registros resultantes de la ejecución de una consulta. Al utilizar objetos de acceso a datos toda la interacción con los datos se produce a través de objetos recordset . Todos estos objetos están formados por registros y campos. Así pues si queremos declarar un objeto recordset para después asignarlo a una tabla de la base de datos primero debemos declarar el objeto recordset y después abrir la tabla mediante el método openrecordset. Con el método openrecorset podremos manejar las tablas de las bases de datos pero debemos tener en cuenta que para tabla que queramos abrir debemos crear un objeto recordset.

Set variable=objeto.openrecordset(origen)

Modificar registros

Una vez abierto el objeto recordset podemos proceder a realizar las modificaciones de los registros, podemos agregar nuevos registros modificarlos, borrarlos etc. Todos estos metodos vamos a verlos a continuación

Método addnew

Utilizaremos el método addnew para crear y agregar nuevos registros en el objeto recordset **recordset.addnew** una vez se hayan introducido los datos en el nuevo registro debemos utilizar el método update para guardar los cambios y agregarlo al conjunto de registros no se modificara la base de datos hasta que se utilice esta opcion

Ejemplo

Clientes.addnew

Clientes.update

Método edit

Copia el registro actual en un objeto recordset en el buffer de copia para así poder modificar el registro. Utilizaremos este método para modificar un registro ya existente **recordset.edit**

Una vez hemos llamado al método edit los cambios efectuados en los cambios del registro actual se crearan en el buffer de copia. Al terminar de realizar los cambios deseados usaremos el método update para guardarlos en la tabla

Método update

Guarda el contenido del buffer de copia en un objeto recordset, es decir actualiza la base de datos el contenido del registro situado en la memoria

Recordset.update

Método delete

Este método elimina el registro actual de un objeto recordset. Para eliminar un registro debe haber un registro actual en el recordset antes de utilizar delete pues de lo contrario se producira un error interceptable una vez eliminado el registro este sigue siendo actual.

Recordset.delete

Control Dbgrid

El control DBgrid una cuadricula de celdas que podemos utilizar para editar una tabla o una consulta enlazado con un control data

- Permite modificar el contenido de las celdas y actualizar los cambios de la base de datos
- Permite borrar o añadir registros a una tabla

Una vez tenemos enlazado el control DBgrid tenemos que configurar las columnas para hacerlo de forma automática hacemos clic en el botón derecho sobre el DBgrid y escogemos la opción **recuperar campos** del menú emergente después de esto tendremos una columna por cada campo

SQL

Es un lenguaje desarrollado especialmente para las bases de datos y preparado especialmente para hacer consultas y cálculos de una o varias tablas. Gracias al SQL podemos conseguir en nuestras aplicaciones orientadas a bases de datos unos resultados impresionantes su uso no es muy complicado y si ya conocemos el uso de SQL en las consultas de Microsoft Access en VISUAL Basic nos será mucho mas sencillo

Antes de que veamos la utilización del SQL en nuestras aplicaciones vamos a explicar cuales serian las normas para su uso. Siempre que declaremos un SQL debemos tener en cuenta que tendremos que hacerlo en un string

Los nombres de los campos y tablas que contengan mas de una palabra debemos encerrarlos entre corchetes [tabla clientes]

Cuando nos referimos a un campo de una tabla debemos poner primero el nombre de la tabla y después el nombre del campo separados ambos por un punto por ejemplos clientes.codigo

Metodología de la programación

Hablando en lenguaje de metodología las variables son lugares donde se introducen diferentes tipos de datos para guardarlos y usarlos mas tarde existen diferentes tipos de variables y nos acogemos a una u otra dependiendo de lo que queramos almacenar en ellas. Para poder disponer de ellas en un programa previamente hay que crearlas aportándoles una denominación por regla general significativa de la función que vamos a aplicar y de que tipo va a ser. Técnicamente diremos que las variables son zonas de memoria identificadas por un nombre donde almacenamos la información sean valores o datos y que podemos

modificar y alterar cuando deseemos.

Las instrucciones de entrada y de salida sirven de comunicador entre el programa y el exterior. Las instrucciones de entrada sirven para que el usuario envié mensajes hacia nuestro programa usando el teclado u otros dispositivos. En resumen las instrucciones de entrada y de salida son el medio por el que se puede establecer un entendimiento entre el programa y el usuario de este

Sintaxis

Instrucción de salida: escribir(cadena explicativa)

Instrucción de entrada: leer(cadena explicativa)

En ambos casos la cadena explicativa es opcional pero recomendable también lo es la expresión en la instrucción de salida pero al menos debe aparecer en una de las dos.

En la instrucción de entrada el valor leido del teclado se guarda en la variable indicada.

Procedimientos y funciones

Cuando tenemos un conjunto de instrucciones que realiza una labor especifica y se necesita repetir esa labor en mas de una ocasión conviene agruparlas en un procedimiento o función llamados también subprogramas. También resulta útil para reutilizar código.

Dentro de un subprograma podemos definir también variables y constantes que solo existan en el ámbito de este, es decir antes y después de ejecutar el subprograma no existirán y no las podemos utilizar. Sin embargo si podemos utilizar las variables definidas en el programa principal siempre que no se llamen de la misma forma que una definida dentro de un subprograma. Pero utilizar las variables fuera de un subprograma no es recomendable porque si cometemos un error en su programación y modificamos el contenido de una variable importante erróneamente podemos hacer que otras partes del programa también fallen el terrible fallo colateral.

Para evitar esto están los parámetros son variables del subprograma a las que se les asigna un valor inicial desde fuera del subprograma.

Ficheros

Abrir y cerrar ficheros

Al abrir un fichero hay que guardar la referencia para poder distinguirlos de otros ficheros que pudiéramos tener abiertos. Esta referencia es de un tipo especial que se llama por supuesto fichero y se guarda en variables de este tipo. También tendremos que indicar si abrimos el fichero para leer o escribir, es decir for read o for write

La primera vez que usamos la instrucción de entrada leemos el primer dato del fichero. Cada vez que volvamos a usar dicha instrucción pasamos a leer el siguiente dato del fichero hasta llegar al ultimo registro.

.