UNIVERSIDAD TECNOLÓGICA DE MORELIA

OFIMATICA

PROGRAMACIÓN

Profesor

Alumno:

Calificación:

INTRODUCCIÓN.

Visual Basic es uno de los tantos lenguajes de programación que podemos encontrar hoy en día. Dicho lenguaje nace del BASIC (Beginner´s All−purpose Symbolic Instruction Code) que fue creado en su versión original en el Dartmouth College, con el propósito de servir a aquellas personas que estaban interesadas en iniciarse en algún lenguaje de programación. Luego de sufrir varias modificaciones, en el año 1978 se estableció el BASIC estándar. La sencillez del lenguaje ganó el desprecio de los programadores avanzados por considerarlo "un lenguaje para principiantes".

VISUAL BASIC

Es un lenguaje de programación que se ha diseñado para facilitar el desarrollo de aplicaciones en un entorno grafico (GUI−GRAPHICAL USER INTERFACE) Como Windows 98, Windows NT o superior.

CARACTERÍSTICAS DE VISUAL BASIC.

Diseñador de entorno de datos: Es posible generar, de manera automática, conectividad entre controles y datos mediante la acción de arrastrar y colocar sobre formularios o informes.

Los Objetos Actives son una nueva tecnología de acceso a datos mediante la acción de arrastrar y colocar sobre formularios o informes.

Asistente para formularios: Sirve para generar de manera automática formularios que administran registros de tablas o consultas pertenecientes a una base de datos, hoja de calculo u objeto (ADO−ACTIVE DATA OBJECT)

Asistente para barras de herramientas es factible incluir barras de herramientas es factible incluir barra de herramientas personalizada, donde el usuario selecciona los botones que desea visualizar durante la ejecución.

En las aplicaciones HTML: Se combinan instrucciones de Visual Basic con código HTML para controlar los eventos que se realizan con frecuencia en una pagina web.

La Ventana de Vista de datos proporciona acceso a la estructura de una base de datos. Desde esta también acceso al Diseñador de Consultas y diseñador de Base de datos para administrar y registros.

DESCRIPCIÓN DE VENTANAS PRINCIPALES

Ejecutamos Visual Basic (VB) y al comenzar podemos ver la siguiente ventana que nos permitirá saber que en que formato deseamos el programa:

Está nos esta mostrando la diferentes formas para comenzar, hay 3 formas, la primera **Nuevo** es iniciando un Nuevo Proyecto, seleccionando uno tipo de proyecto, la segunda **Existente** es abriendo un proyecto ya existente, el archivo tiene que contener la extensión **.vbp*, **.mak* o **.vbg*, y por ultimo la tercera **Recientes** es seleccionando un proyecto de la lista de archivos recientemente abiertos (VB se encarga automáticamente de mostrar una lista de los últimos proyectos abiertos).

Para continuar se elige EXE estándar y luego Abrir, ahora se explicara la Ventana principal de Visual Basic.

LAS PARTES DEL ENTORNO DE VISUAL BASIC.

Barra de titulo: muestra el nombre del proyecto y del formulario q se está diseñando actualmente

Barra de menús: agrupa los menús despegables que contienes todas las operaciones que pueden llevarse a cabo con Visual Basic 6.0.

Barra de herramientas estándar: contienen los botones que se utilizan con mayor frecuencia cuando se trabaja con un proyecto. Simplifica la elección de opciones de los menús Archivo, Edición, Ver y Ejecutar; además, en el área derecha presenta la ubicación (coordenadas) y el tamaño del objeto seleccionado

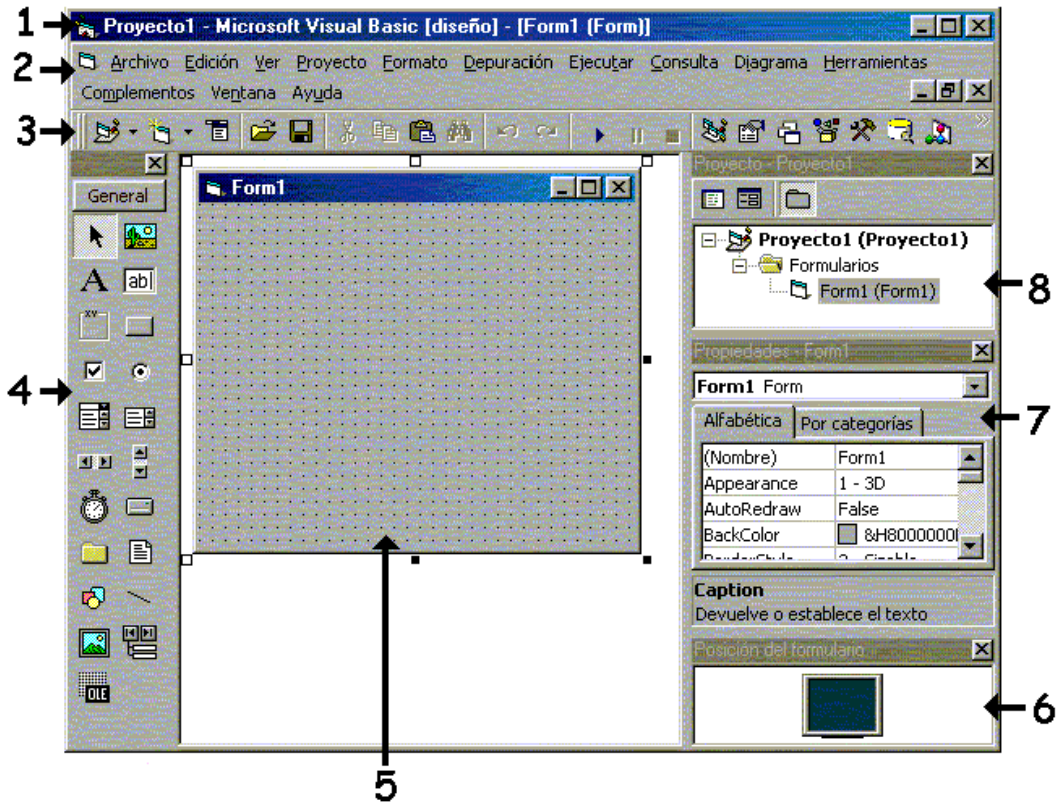
Ventana de formulario: es el área donde se diseña la interfaz gráfica, es decir, es donde se inserta electo gráficos, como botones, imágenes, casilla de verificación, cuadros de listas, etc.

Cuadro de herramientas: presenta todos los controles necesarios para diseñar una aplicación, como cuadros de texto, etiquetas, cuadros de listas, botones de comandos, etc.

Ventana de proyecto: muestra los elementos involucrados en el proyecto, como formularios, módulos, controles oxc, etc. Cada elemento puede seleccionarse en forma independiente para su edición.

Ventana de posición del formulario: muestra la ubicación que tendrá el formulario en la pantalla, cuando ejecute la aplicación. Esta ubicación puede cambiarse si se hace clic con el botón izquierdo del mouse.

La Ventana propiedades muestra todas las propiedades del control actualmente seleccionado, en este caso muestra las propiedades del Form1, luego podemos ver que abajo dice "Form1 Form", lo que está en negrita es el nombre del objeto, y lo que le sigue es el tipo de objeto, en este caso es un Formulario (Form)



1.− Barra de titulo, en está se muestra el nombre del proyecto actual en el que se esta trabajando, luego viene el nombre del programa, continuado a "Microsoft Visual Basic viene el estado en el que está el proyecto, hay 3 tipos de estado en **[Diseño]** cuando se está escribiendo el código, **[Ejecución]** el programa se encuentra en ejecución, o en estado de **[Interrupción]** que es cuando en estado de ejecución se produjo un error, por ultimo en la barra de titulo se puede visualizar el nombre del Form (Formulario) actual en que se está trabajando y como se está trabajando **[Form1 [Form]]** cuando se trabaja en el diseño del Formulario o **[Form1 [Código]]** cuando se trabaja en el código para ese formulario.

2.− En la barra de menús se encuentran todas las opciones para manejar Visual Basic, a medida que se vayan utilizando se irán explicando.

3.− La barra de herramientas, contiene varios accesos directos a los menús, para agilizar el manejo de Visual Basic.

4.− Se le llama Cuadro de herramientas y sirve para colocar objetos en la Ventana de Diseño, como Cuadros

de textos, Botones, Imágenes, y otros. Esto funciona muy fácil ya que seleccionamos por ejemplo un CommandButton (botón), y vamos al ventana diseño y lo creamos haciendo Clic con el Mouse (sin soltarlo!) y lo arrastramos hasta el tamaño que deseemos, en ese momento soltamos el botón. Y listo ya tienes un botón estilo Windows para usar!.

5.− La Ventana diseño, es aquí donde se le da la apariencia al programa, es decir lo que el usuario va a ver. Los puntillos del fondo son utilizados con el fin de alinear más fácil los objetos en el Formulario(Ventana).

6.− En la ventana Posición del Formulario, se puede ver un monitor y dentro de él una ventanita, esto sirve para definir la posición de la ventana cuando se inicie el programa, es decir donde se va a mostrar. Pruebe hacer clic sobre la ventana (sin soltar el botón!) y arrastre la ventana dentro del monitor, luego suelte el botón del Mouse. Y cuando se ejecute la aplicación la ventana aparecerá en esa posición.

7.− La Ventana propiedades muestra todas las propiedades del control actualmente seleccionado, en este caso muestra las propiedades del Form1, luego podemos ver que abajo dice "**Form1** Form", lo que está en negrita es el nombre del objeto, y lo que le sigue es el tipo de objeto, en este caso es un Formulario (Form).

8.− En la ventana Posición del Formulario, se puede ver un monitor y dentro de él una ventanita, esto sirve para definir la posición de la ventana cuando se inicie el programa, es decir donde se va a mostrar. Pruebe hacer clic sobre la ventana (sin soltar el botón!) y arrastre la ventana dentro del monitor, luego suelte el botón del Mouse. Y cuando se ejecute la aplicación la ventana aparecerá en esa posición.

8.− El Explorador de proyectos es donde visualizamos todas los Formularios(Ventanas), Módulos, Clases, entre otros del proyecto actualmente abierto, es decir de la aplicación que se está creando, en este caso hay un solo Formulario llamado Form1, lo que aparece dentro de paréntesis es el nombre de archivo. Y este Formulario se encuentra dentro de una carpeta llamada "Formularios", por ser que VB clasifica los archivos por tipos, cuando use otros tipos de archivos además de Formularios podrá visualizar otras carpetas.

VARIABLE

Basic, desde siempre, al contrario de otros sistemas de programación, no exigió la definición previa de una variable. Una variable, como Vd. seguro que conoce, es un nombre que en el programa le asignamos a un dato. Ese dato podrá cambiar.

Dim: Al declarar una variable con esta palabra estamos diciendo que la variable sea local al ámbito en que se declara. Puede ser dentro de un procedimiento o dentro de un formulario, de esta forma no sería accesible desde los demás procedimientos o formularios.

Public: Las variables declaradas serán publicas y podrán estar accesibles desde todos los formularios de la aplicación. Para conseguirlo tendremos que declararlas en un módulo de código, no en la sección declarations de cualquier formulario de los que conste la aplicación. Para crear un módulo de código en el menú principal de Visual Basic marcamos en INSERT/MODULE y aparecerá junto a los demás formularios de la ventana de proyecto aunque con un icono distinto indicando que se trata de un módulo de código.

Static: Con esta forma de declarar variables conseguiremos que las variables locales no se creen y se destruyan al entrar y salir de los procedimientos donde fueron declaradas sino que se mantenga su valor durante todo el periodo de ejecución de la aplicación. De esta forma a entrar en algún procedimiento las variables recuerdan el valor que tenían cuando se salió de él.

TIPOS DE VARIABLES

TIPO	**COMENTARIO**

BOOLEAN	Sólo admite 2 valores TRUE o FALSE
BYTE	admite valores entre 0 y 255
INTEGER	admite valores entre −32768 y 32767
LONG	admite valores entre −2.147.483.648 y 2.147.483.647
SINGLE	admite valores decimales con precisión simple
DOUBLE	admite valores decimales de doble precisión
CURRENCY	válido para valores de tipo moneda
STRING	cadenas de caracteres
DATE	fechas, permite operar con ellas

Constante: Declaración de constantes que pueden ser usadas en cualquier punto en lugar de su valor, permitiendo cambiarlo cuando sea necesario, sin tener que cambiarlo en todos los sitios en que se utiliza. La expresión no puede utilizar llamadas a funciones, pues la constante se calcula en tiempo de compilación, no en tiempo de ejecución.

EXPRESIONES LÓGICAS

Existen los llamados **operadores lógicos**, que nos permiten establecer condiciones que dependadan de más de un criterio de selección. Todos los operadores hacen que la expresión en la que se encuentren se evalúa a verdadero o falso, sin posibilidad de cualquier otro valor.
Estas expresiones son:

And: exp1 And exp2, donde se evalúa a verdadero s&oacutelo en el caso que el exp1 como exp2 se evalúen a verdadero. En cualquier otro caso se evalúa a falso.

Or: exp1 Or exp2, donde se evalúa a verdadero cuando alguna de las expresiones exp1 o exp2 se evalúa a verdadero.
Not: Not exp1. aquí se evalúa a verdadero si exp1 es falso y se evalúa a falso si exp1 es verdadero.
Xor: exp1 Xor exp2, se evalúa a verdadero sólo en el caso de que una y sólo una de las expresiones exp1 o exp2 se evalúa a verdadero.

OPERADORES DE VISUAL BASIC

En Visual Basic existe un gran número de operadores que se pueden utilizar para crear fórmulas.
Los operadores más utilizados en una aplicación de Visual Basic son los siguientes:

Operador	**Operación que realiza**
+	Suma / Concatenación de cadenas de caracteres
−	Resta
*	Multiplicación
/	División
\	División entera
Mod	Resto de la división entera
^	Exponenciación
&	Concatenación de cadena de caracteres

OPERADORES RELACIONALES

Los operadores relaciónales que reconoce vcpp son:

Operador	**Significado**
==	Igual que
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
!=	No es igual que o es diferente que

SENTENCIAS

SENTENCIA DIM

Es la forma mas común de declarar una variable como

Privada. Puede emplearse en un Procedimiento, Función, Formulario o Módulo. La sintaxis es de la siguiente forma:

Dim nombrevariable **As Integer** (o el tipo que sea)

Declarando una variable con la sentencia DIM, en un ***formulario***, ***Función***, ***procedimiento*** o ***módulo***, el entorno de la variable será el explicado anteriormente para una variable declarada como Privada. Es decir, esa variable no sale del formulario, procedimiento ó módulo donde se declaró. Cada vez que entremos al formulario, procedimiento o módulo, esa variable tomará el valor cero (si es numérica) o nulo (si es string).

Resumimos la forma de declarar una variable :

En un Procedimiento (La variable no puede usarse fuera de esta Procedimiento)

Dim Variable **As** Tipovariable

En un Formulario (En su sección de declaraciones)

SENTENCIAS CONDICIONALES.

Llamamos sentencias condicionales a aquellas que se realizan si se cumple una determinada condición. Son las sentencias por las que empieza cualquier texto de Basic, y este no va ser menos.

La sentencia condicional mas usada es:

Si se cumple una condición **Entonces**

Realiza estas instrucciones

Si no se cumple

Realiza estas otras instrucciones

Fin de la sentencia.

Así de fácil es programar en Basic. Lo que ocurre es que esta herramienta habla inglés, y lo descrito anteriormente toma la forma:

If condición **Then**

Instrucciones

Else

Otras instrucciones

End If

LA SENTENCIA SELECT

La sentencia SELECT "selecciona" los campos que conformarán la consulta, es decir, que establece los campos que se visualizarán o compondrán la consulta. El parámetro 'lista_campo' está compuesto por uno o más nombres de campos, separados por comas, pudiéndose especificar también el nombre de la tabla a la cual pertenecen, seguido de un punto y del nombre del campo correspondiente. Si el nombre del campo o de la tabla está compuesto de más de una palabra, este nombre ha de escribirse entre corchetes ([nombre]). Si se desea seleccionar todos los campos de una tabla, se puede utilizar el asterisco (*) para indicarlo.

Una sentencia SELECT no puede escribirse sin la cláusula FROM. Una cláusula es una extensión de un mandato que complementa a una sentencia o instrucción, pudiendo complementar también a otras sentencias. Es, por decirlo así, un accesorio imprescindible en una determinada máquina, que puede también acoplarse a otras máquinas. En este caso, la cláusula FROM permite indicar en qué tablas o en qué consultas (queries) se encuentran los campos especificados en la sentencias SELECT. Estas tablas o consultas se separan por medio de comas (,), y, si sus nombres están compuestos por más de una palabra, éstos se escriben entre corchetes ([nombre]).

Tenemos aquí algunos ejemplos de mandatos SQL en la estructura SELECT...FROM...:
SELECT nombre,apellidos FROM clientes;
Selecciona los campos 'nombre' y 'apellidos' de la tabla 'clientes'.
SELECT clientes.nombre, producto FROM clientes, productos;
Selecciona el campo 'nombre' de la tabla 'clientes', y el campo 'producto' de la tabla productos.
Hay que tener en cuenta que si dos tablas poseen el mismo nombre de campo (un 'nombre' de cliente y un 'nombre' de producto, hay que especificar también la tabla a la cual pertenece dicho campo, ya, que de lo contrario, seleccionaría ambos nombres).
SELECT pedidos.* FROM pedidos;
Selecciona todos los campos de la tabla 'pedidos'.
SELECT * FROM pedidos;
Selecciona todos los campos de la tabla 'pedidos'.
SELECT nombre, apellidos, telefono FROM clientes;
Selecciona los campos 'nombre', 'apellidos' y 'telefono' de la tabla 'clientes'. De esta manera obtenemos una agenda telefónica de nuestros clientes.
SELECT [codigo postal] FROM [tabla morosos];

Selecciona el campo 'codigo postal' de la tabla 'tabla morosos'.

BUCLE POR CONTADOR

Realiza el bucle tantas veces como le indiquemos. Por ejemplo, en este bucle nos va a presentar las 26 letras mayúsculas del alfabeto inglés

```
For N=65 To 90
  Label1.caption = Chr ( N )
  Next N
```

Este "programa" nos presentará en una caja (Label) los caracteres cuyo número ASCII vaya desde el 65 (A) al 90 (Z) Comenzará presentando el correspondiente al número 65, e irá presentando sucesivamente el 66, el 67, etc., hasta llegar al 90, donde se parará.

BUCLES POR CONDICIÓN

Ejecuta las instrucciones del bucle mientras se cumple una condición

```
  X = 0
Do While X < 1000
X = X + 1
Loop
```

El programa toma una variable (X) que previamente tuvimos la curiosidad de ponerla a cero, e incrementa su valor una unidad. Analiza si el valor de X es menor que 1000, y si es cierto, vuelve a realizar el bucle. Así hasta que X ya no sea menor que 1000. Al dejar de cumplirse que X sea menor que 1000, sale del bucle. Acabamos de realizar un temporizador, y también de exponer las sentencias condicionales y los bucles, inicio de cualquier curso de Basic. Como final de lección, se propone un problema. Con el primer bucle, donde visualizábamos los caracteres A a la Z, posiblemente no nos diese tiempo de ver cada una de las letras que iban apareciendo en la pantalla, en la etiqueta Label1, dado que cambiaría con mucha velocidad, y solamente veríamos la Z, que es donde se detuvo el programa. Podemos poner un temporizador cada vez que presente una letra, y así nos dará tiempo a verla. Con los dos bucles vistos anteriormente ya podemos hacerlo. Si lo que queremos es que el programa se ejecute mientras no se cumpla una determinada condición, la sentencia será:

```
  X = 0
Do Until X > 1000
X = X + 1
```

Loop

Observe que la diferencia entre una y otra es la condición, While para indicar Mientras se cumpla que ... y Until para indicar Mientras no se cumpla que ....

Utilizaremos la condición de While para: (Mientras se cumpla la condición)

For N=65 To 90

Label1.caption = Chr (N)

Label1.RefreshN ' Refresca la etiqueta

X = 0

Do While X < 1000

X = X + 1

Loop

Next N

La instrucción de la cláusula Else, sólo se ejecuta cuando el resto de condiciones han fracasado. Cualquiera de los bloques de instrucción puede contener un número arbitrario de instrucciones, incluida la posibilidad de que existan otras instrucciones IF−THEN−ELSE. Se pueden colocar todas las sentencias de código que queramos en cada uno de los bloques de sentencias, siempre que cada sentencia vaya en una línea distinta..

Un ciclo similar a el 'Do While' es el 'Loop While'. La principal diferencia es que el ciclo Do While la condición para poder continuar esta al principio (primero ceha la condición y luego ejecuta el código que se desea repetir), mientras que en el Loop While esta al final, esto es después de que ejecuta el código checa si debe continuar. Esto significa que un ciclo Do While, puede nunca ser ejecutado, mientras que un ciclo Loop While siempre al menos se ejecutará una vez.

Cuando existe un gran número de instrucciones a evaluar es aconsejable utilizar otra estructura de decisión, como puede ser las instrucción **Select Case**. Esta instrucción no dá más potencia al lenguaje, pero hace que el código sea más legible y eficiente. Esta sentencia permite realizar operaciones diferentes dependiendo del valor de una variable.

La estructura de repetición **For...Next** es adecuada cuando conocemos el número de veces que debe repetirse un conjunto de instrucciones y deseamos reducir la cantidad de código escrito.

Otra estructura de repetición es **Do...Loop**. Esta estructura se utiliza cuando desconocemos cuántas veces se ha de ejecutar el bucle.

Si utilizamos **While** el bucle se repite mientras la condición se cumpla y si utilizamos **Until** el bucle se repetirá hasta que la condición dé valor verdadero. Con While, la condición es comprobada al principio del bucle, por lo que si no se cumple al iniciarse, el cuerpo del bucle no se ejecutará. Con Until se comprueba a la salida del bucle, por lo que por lo menos, una vez se ejecutará el bucle.

Los siguientes comandos pueden ser usados para repetir varias veces la ejecución de una sección de código.

Ciclo Do (Sintaxis):

Do While *condición*

líneas de código

Loop

Ciclo While (Sintaxis):

Do

líneas de código

Loop While *condición*

Los ciclos For si los programadores saben cuantas veces un cliclo debe ser ejecutado.

Ciclo For (Sintaxis):

For *contador* = *valor_inicial* To *valor_final* [Step *incremento*]

líneas de código

Next [*contador*]

If estructura de decisisón

If en Línea simple

End If

El If de multi−línea necesita una expresión 'End If' para definir el final del bloque *If*.

Si cualquiera de las dos condiciones es falsa, el programa ejecutará el código que está en el bloque 'Else'.

El comando 'ElseIf' puede usarse para checar una segunda condición en caso de que la primera sea falsa.

Select' define el final del bloque de casos. Si deseamos que presente un mensaje si se introduce un valor no definido en el bloque de casos,

El incremento en Step como parte del ciclo, es usado para decirle a VB cuanto debe aumentar en cada ciclo la variable de control. Introduce el siguiente código y observa que valores se imprimen.:

For test = 1 To 10 Step 2

Print test

Next test

Una de las mas poderosas características de cualquier computador es la capacidad que tiene de tomar decisiones.

Es decir al comparar dos alternativas diferentes el computador puede tomar una decisión basándose en la evaluación que hace de alguna condición.

ejemplo de instrucciones condicionales:

si sueldo > 3000

desplegar rico

si no

desplegar pobre

Fin−si

si sexo = 'm'

imprime mujer

si no

imprime hombre

Fin−si

De los ejemplos observar que los caminos a seguir por el computador dependerán de la evaluación que el computador hace con y de la condición.

Todo lenguaje de programación debe tener instrucciones que permitan formar condiciones e instrucciones que pueden evaluar esas condiciones.

Pero recordar que lenguajes modernos y orientados a clientes−servidores de igual forma tienen componentes que permiten del mismo modo al usuario tomar decisiones incluso directamente en pantalla, es decir tambien existen los objetos, controles o componentes de selección y decisión en html.

El formato general de una instrucción condicional es:

Como se observa, son cuatro partes bien diferenciadas entre si;

- La propia instrucción condicional en si
- La condición
- El grupo cierto de instrucciones
- El grupo falso de instrucciones

Cuando el computador evalúa una condición el resultado de esa evaluación solo es evaluado de dos maneras o la condición es CIERTA o la condición es FALSA.

Esto dependerá del valor que tenga asignado o que se haya capturado para la variable que esta en la condición, por ejemplo si se capturo 6000 en sueldo en el ejemplo a) entonces el computador indicaría que la condición es CIERTA pero en otro caso si a la variable sueldo primero se le asigno un valor de 250 entonces el computador indicaría que la condición es FALSA.

Ya dependiendo del resultado de la evaluación, el computador ejecuta las instrucciones contenidas en el grupo de cierto o falso respectivamente.

Empezaremos el análisis por la CONDICIÓN.

INSTRUCCIONES DE CONTROL DE PROGRAMA

CONDICIONES SIMPLES

En general todas las condiciones se forman con;

Variables	**Operadores Relaciónales**	**Constante o Variables**
sexo	=	m
sueldo	>	300,000

Una condición simple se define como el conjunto de variables y/o constantes unidas por los llamados operadores relaciónales.

INSTRUCCIONES DE CONTROL DE PROGRAMA

INSTRUCCIÓN IF

Es la instrucción condicional mas usada en los diversos lenguajes de programación, su formato completo y de trabajo en vcpp es:

cargar o asignar la variable de condición;

if (condición)

{ grupo cierto de instrucciones;}

else

{ grupo falso de instrucciones; };

Primus.− Observar **donde van y donde no van** los puntos y comas;

Secundus.− La condición va entre paréntesis ;

Tertius.− Si un if no ocupa un grupo falso de instrucciones entonces no se pone el else, y la llave antes del else si terminaría con punto y coma.

Ejemplo:

<HTML>

<FORM ACTION=/CGI−BIN/tusitio/PROG2.EXE METHOD=GET>

SUELDO:<INPUT TYPE=TEXT NAME=SUELDO>

<INPUT TYPE=SUBMIT VALUE=OK>

```
</FORM></HTML>

#using <mscorlib.dll>

#using <System.dll>

#using <lcnet.dll>

using namespace System;

void main(){

//declarando variables

int sueldo;

// capturando,cargando y convirtiendo

//los datos de la forma a las variables

sueldo = Int32::Parse(lcnet::getparametro("SUELDO"));

//construyendo y desplegando la pagina de salida

Console::WriteLine("Content−Type:text/html\n");

if (sueldo >= 3000)

{Console::WriteLine(S"RICO");}

else

{Console::WriteLine(L"POBRE");};

}//fin main
```

INSTRUCCIONES DE CONTROL DE PROGRAMA

CONDICIONES COMPUESTAS

En muchas ocasiones es necesario presentar mas de una condición para su evaluación al computador.

Por ejemplo que el computador muestre la boleta de un alumno si este estudia la carrera de medicina y su promedio de calificaciones es mayor de 70.

Una condición compuesta se define como dos o mas condiciones simples unidas por los llamados operadores lógicos.

Los operadores lógicos que vcpp reconoce son;

OPERADOR	**SIGNIFICADO**
&&	"Y" LOGICO

||	"O" LOGICO
!	"NO" NEGACION

Ejemplo:

if ((carrera=="informatica") && (sexo=="m")) etc, etc, etc.

Notas:

ä Observar que cada condición simple lleva sus propios parentesis.

ä Si la variable es de tipo string el dato va entre comillas("), pero si la variable es de tipo char el dato va entre apostrofes(').

Para que el computador evalúe como CIERTA una condición compuesta que contiene el operador lógico "&&", las dos condiciones simples deben ser ciertas.

Para que el computador evalúe como CIERTA una condición compuesta que contiene el operador lógico "||", basta con que una de las condiciones simples sea cierta.

La cantidad total de casos posibles cuando se unen dos o mas condiciones simples esta dada por la relación donde n = cantidad de condiciones, la primera mitad de ellos empieza en cierto y la segunda mitad en falso.

Ejemplo, si formamos una condición compuesta con dos condiciones simples y el operador lógico "y", la cantidad total de casos posibles serian , y se puede construir la siguiente tabla de verdad.

Tabla de verdad con "y"

IRA COND SIMPLE	**2DA COND SIMPLE**	**EVALUACION**
C	C	C
C	F	F
F	C	F
F	F	F

La evaluación final, se obtiene usando la regla anteriormente descrita para una condición compuesta, que contiene el operador "Y".

Esta tabla significa lo siguiente;

1.− Cualquiera que sean la cantidad de datos procesados, siempre caerá en uno de estos cuatro casos generales.

La tabla de verdad para una condición compuesta con "Or" es la siguiente;

IRA COND SIMPLE	**2DA COND SIMPLE**	**EVALUACION**
C	C	C
C	F	C
F	C	C
F	F	F

Construir una tabla de verdad para una condición compuesta de tres o mas condiciones simples, es también tarea sencilla, solo recordar que;

1.− La cantidad posible de casos es posible, la mitad empiezan con Cierto y la otra mitad empiezan con Falso.

2.− Para evaluar esta condición triple primero se evalúan las dos primeras incluyendo su operador bajo las reglas ya descritas y luego se evalúa el resultado parcial contra la ultima condición y ultimo operador para obtener la evaluación final.

Ejemplo una condición compuesta de tres condiciones simples, donde el primer operador lógico es el "y" y el segundo operador lógico es el "O", daría la siguiente tabla de verdad.

Ira cond	**2da cond**	**Eval 1a Y 2a**	**3ra cond**	**Eval eval O 3ra**
C	C	C	C	C
C	C	C	F	C
C	F	F	C	C
C	F	F	F	F
F	C	F	C	C
F	C	F	F	F
F	F	F	C	C
F	F	F	F	F

CAPTION:

Representa es texto que aparecerá en el menú.

NAME:

Es el nombre del control. Se puede anteponer el prefijo mnu más el texto del título del menú.
CHECKED:

Establece si debe aparecer una marca de verificación a la izquierda del texto del objeto del menú. La marca la puedes utilizar cuando deseas informar de la opción que está activada en un momento determinado.
ENABLED:

Indica si el objeto podrá recibir eventos del usuario. Si tiene el valor False el elemento aparecerá atenuado en el menú.

VISIBLE:

Indica si el objeto debe mostrarse o no, si se establece este valor a False, los elementos situados a la derecha ocupan el lugar del elemento no visible.

WINDOWSLIST:

Establece el valor que determina si un objeto de menú mantiene una lista de las ventanas MDI secundarias del formulario actual. Elementos del Menú:

Utilizando los botones de sangría se puede determinar el nivel del objeto menú que quieras seleccionar. Así, los títulos de menú se sitúan en el nivel superior, mostrándose en la lista inferior del editor de menús, lo más a

la izquierda posible. En muchas aplicaciones se pueden ver unas barras horizontales en los menús desplegables que realizan la función de dividir en secciones dichos menús. Este elemento es conocido como un separador, y sirve para separar aquellos comandos del menú que tengan_un_motivo_en_común.
Para crear un separador, basta con introducir un guión (−) en la propiedad Caption. No se puede crear un elemento en un nivel inferior inmediatamente después de un separador, es decir, no puedes tener como título de menú un separador.

TECLAS DE MÉTODO ABREVIADO:

Al introducir el carácter & el menú está creando una tecla de acceso a dicho objeto. Esta tecla se representa subrayada en el texto de menú y respresenta la tecla del teclado que el usuario puede pulsar para ejecutar la acción. Dicha tecla se corresponde con la letra que sigue al caracter & en la propiedad Caption del objeto menú. Si se trata de un título de menú tienes que usar la tecla de acceso en combinación con la telca [Alt.]. Una vez abierto un título de menú y desplegado sus elementos, para acceder rápidamente a uno de ellos, puedes usar la tecla de acceso directamente.

Tienes que tener cuidado en no utilizar las mismas teclas de acceso para elementos de un mismo nivel de menú ya que entonces no funcionarán. El editor de menú también admite la incorporación de teclas de método abreviado. Las abreviaturas del menú son combinaciones de teclas que se pueden utilizar en lugar de elegir el elemento del menú correspondiente. Para crear una tecla de método abreviado tienes que utilizar la propiedad Shortcut, a la que puedes acceder al crear el menú desde el editor de menús.

EVENTOS DEL MENÚ

Cuando en tiempo de diseño seleccionamos la opción de menú de un formulario que estamos creando, Visual Basic muestra el procedimiento de evento click de dicho elemento del menú. En el caso del título del menú, el procedimiento click tiene como acción predeterminada la de mostrar los elementos de menú del nivel inferior, por lo que no será necesario tener que programar esta acción. Esto también se puede aplicar a los elementos de un menú que son a su vez títulos de submenús. En casi todos los casos tienes que utilizar una opción del menú para descargar el formulario o para finalizar la aplicación. La opción de texto salir suele ser el más apropiado. La instrucción Unload descarga de memoria el formulario que se especifique. Se puede usar de varias formas, por ejemplo si estamos dentro de un Form llamado frmconsulta podremos salir de él de dos formas: Una es Unload Me o Unload frmconsulta.

MSGBOX

La síntaxis completa de la función **MsgBox** es:

MsgBox(mensaje[, botones][, t&iacutetulo][, archivoAyuda, contexto])

Los valores que pueden tomar el parámetro botones son:

Constante	Valor	Descipción
vbOKOnly	0	Muestra sólo el botón aceptar
vbOKCancel	1	Botones aceptar y cancelar
vbAbortRetryIgnore	2	Botones anular, reintentar e ignorar
vbYesNoCancel	3	Botones si, no y cancelar
vbYesNo	4	Botones si y no
vbRetryCancel	5	Botones reintentar y cancelar
vbCritical	16	Muestra el icono de mensaje crítico

vbQuestion	32	Icono de interrogación
vbExclamation	48	Icono de exclamación
vbInformation	64	Icono de mensaje de información
vbAplicationModal	0	Cuadro de diálogo modal de la aplicación
vbSystemModal	4096	Cuadro de diálogo modal del sistema

El parámetro botones es de valor numérico, y además de los botones que aparecen en le cuadro de diálogo, también puede indicar el icono que acompañe al mensaje y el tipo de diálogo modal que es.
Cuando un cuadro de diálogo es modal de aplicacióen tienes que cerrarlo para interactuar con otra ventana de la misma aplicación. Cuando es un cuadro de diálogo modal del sistema, todas las aplicaciones que estén ejecutándose se suspenden hasta que el usuario responda al cuadro de diálogo. El cuadro **InputBox** se consigue a través de la función InputBox. Se utiliza cuando se necesita que el usuario introduzca alguna información.

LOS CONTROLES

Realmente son objetos que disponen de sus propias propiedades y métodos, y cuya utilidad es la de facilitarnos el desarrollo de nuestras aplicaciones. Bueno, este intento de definición puede haber quedado bien, pero para que lo tengáis más claro, tener en cuenta que en cualquier aplicación con la que trabajamos estamos rodeados de controles. Quien no ha visto en multitud de programas los botones ACEPTAR y CANCELAR, un cuadro para introducir texto, una lista con datos, etc.. Pues todos ellos son controles y no tendremos que preocuparnos por crearlos para nuestras aplicaciones sino que ya vienen con el paquete de VB, lo único que tendremos que hacer es modificar sus propiedades: tamaño, color, etc.. para incorporarlos en nuestras aplicaciones y asociarles el código necesario para que se comporten como esperamos al ejecutar la aplicación.

Antes de empezar a conocer los controles básicos veamos cuales son sus características generales:

- **Propiedades**: Todos los controles disponen de una serie de propiedades las cuales podemos cambiar al incluirlos en nuestras aplicaciones. Ejemplos de propiedades son el color, el tipo de letra, el nombre, el texto, etc.
- **Métodos:** Son procedimientos asociados a los controles, es decir, rutinas ya establecidas que podemos invocar desde nuestras aplicaciones para que se realice alguna operación sobre el control. Por ejemplo el control ListView (la lista de archivos que aparece en el explorador de windows) dispone del metodo order que te ordena los datos aparecidos en la lista.
- **Eventos:** Son acciones que pueden ser motivadas por el propio usuario o por mismo sistema operativo. Ejemplos pueden ser el movimiento del raton o hacer click sobre su botón. En Visual Basic digamos que se utiliza la programación orientada a eventos, lo cual es una de las diferencias más importantes respecto a la programación lineal de MS DOS. No necesitamos detectar cuando se ha producido un evento determinado, Windows lo detecta automáticamente. Los eventos ya estan definidos, son bastantes y cada control cuenta con los suyos propios, aunque son muy parecidos. Lo único que tendremos que hacer es asociar el código necesario al evento que necesitemos tratar.

Para mostrar la ventana donde aparecen los controles que Visual Basic carga por defecto nada más arrancar la aplicación tendremos que marcar en **View** del menú principal (versión inglesa) y activar la opción **Toolbox.** Obtendremos una ventana como esta en la pantalla:

Realmente existen muchos más controles, aunque estos son los más utilizados y por eso aparecen por defecto. Para tener acceso a los demás controles tanto de Visual Basic como los controles que incorporan otras aplicaciones marcaremos en **Tools/Custom Controls** del menú principal.

Moviendo el ratón por encima de cualquier control aparecerá una pista indicándonos el control de que se trata.

Para que esta ventana aparezca siempre en primer plano aunque no sea la ventana activa marcaremos con el boton derecho del ratón en cualquier lugar de la ventana y activaremos la opción **Always On Top** del menú contextual.

CONTROLES BASICOS

TextBox

Mediante este control podremos realizar tanto la entrada como la salida de datos en nuestras aplicaciones.

No hace falta que indiquemos las coordenadas de la situación del formulario en pantalla, simplemente tendremos que marcar sobre el control de la caja de herramientas y dibujarlo con el tamaño que queramos en nuestro formulario.

PROPIEDADES

Las propiedades de las que dispone el control son las siguientes:(para obtener el cuadro de propiedades, seleccionar el control y pulsar **F4** o pulsar con el boton derecho para obtener el menú contextual y marcar **Propierties**)

Text: Aqui indicamos el texto que aparecerá en el control. Podemos asignarle cualquier texto en tiempo de diseño o ejecución. También podemos tomar el texto que haya introducido el usuario para tratarlo durante la ejecución.

Name: Esta propiedad la tienen todos los controles, el nombre que viene por defecto en este caso Text1 y es el nombre con el que se conocerá el control cuando lo utilicemos en el codigo. En un mismo formulario no puede haber 2 controles con el mismo nombre. Conviene poner un nombre que represente la funcion que tiene el control en la aplicacion para que el codigo quede más claro. Ejemplo, si en el textbox vamos a introducir la dirección de una persona podemos asignale a esta propiedad el valor Direccion.

MultiLine: Permite que introduzcamos varias lineas de texto en el control en lugar de sólo una.

Alignment: Alineacion que tendrá el texto dentro del control: izquierda, centro o derecha. Para que funcione la propiedad MultiLine debe estar con el valor true.

Locked: Si esta con valor true bloquea el control, es decir, el usuario no puede introducir ni modificar el texto que contenga. Nos puede servir para utilizar el control como salida de datos sin que el usuario pueda modificarlos por error.

Otras propiedades que son comunes a la mayoria de los controles:

Backcolor: color de fondo.

Forecolor: color de letra.

Font: tipo y tamaño de letra.

METODOS

Recordemos que por métodos se entienten los procedimientos o funciones asociados a un control, los cuales

nos permiten realizar ciertas operaciones útiles sobre dicho control: Ej. ordenar sus elementos, buscar un dato, etc..

Pues bien, los controles básicos que vamos a ver en este capítulo únicamente contienen métodos avanzados que no vamos a analizar por ahora, ya que son métodos que no se suelen utilizar. Más adelante cuando veamos otros tipos de controles estudiaremos cuales son los métodos que nos podrán servir. Si alguien está interesado en conocer todas las características de los controles puede hacerlo mirando en la ayuda que proporciona VB, haciendo click sobre cualquier control de la caja de herramientas y pulsando a continuación F1 obtendrá ayuda referente a ese control donde aparecerán todas sus propiedades, metodos y eventos.

EVENTOS

Los eventos son acciones que se pueden realizar en cualquier control: click, doble click, movimiento del ratón. A estos eventos se les puede asociar código para que se ejecute al producir el evento.

MouseMove: al mover el raton por encima del control.

Mousedown: al pulsar cualquier boton del raton

Change: al cambiar el contenido del control

Click: al hacer click con el botón izquierdo del ratón sobre el control

Doubleclick: al hacer doble click con el con el botón izquierdo del ratón sobre el control

Getfocus: este evento se activa cuando el control recibe el enfoque, es decir, cuando se activa el control en tiempo de ejecución para introducir datos en él o realizar alguna operación.

Lostfocus: Es el contrario del anterior evento, se activa cuando el control pierde el enfoque, es decir, se pasa a otro control para seguir introduciendo datos.

EJEMPLO

Vamos a probar el uso del control TextBox mediante un pequeño ejemplo en el que teniendo un único control de este tipo en un formulario, lo programaremos de forma que al pasar el ratón sobre el control (evento **mousemove**) aparecerá en el formulario el texto que contenga.

Observamos que al situar el control en el formulario aparece por defecto el texto **Text1**. Para que no aparezca ese texto al ejecutar la aplicación, debemos cambiar la propiedad **Text** pulsando **F4** y colocar el texto que queramos o no colocar nada.

Lo que queremos hacer es que cada vez que movamos el raton por el control aparezca su contenido en el formulario. Entonces lo que habrá que hacer abrir la ventana de código, seleccionando el control y pulsando **F7,** o con el botón derecho del ratón y la opción **View code** del menú contextual. Este proceso nos llevará al cuadro de la imagen siguiente.

Lo que tendremos que hacer es seleccionar el evento que necesitemos de la sección **Proc**, en nuestro caso mousemove y a continuación teclear el codigo correspondiente: La instrucción print visualiza un texto en el formulario y si le ponemos **text1.text** le decimos que nos muestre la propiedad **Text** del control **Text1** que ese será el nombre que tendrá el control por defecto si no lo hemos cambiado en la propiedad name.

Al ejecutar esta pequeña aplicación pulsando **F5** observaremos como aparece en el formulario lo que hayamos

tecleado en el control cada vez que movemos el raton sobre el Textbox.

Podemos modificar el programa para que responda a cualquier otro evento sin más que seleccionarlo en la sección **Proc** e introduciendo el codigo que sea necesario.

Label

Este control es también uno de los más utilizados, aunque su utilidad queda restringida a la visualización de datos en el mismo, no permitiendo la introducción de datos por parte del usuario.

La forma de utilizarlo es similar a la del control anterior, dibujar el control en el formulario con el tamaño que queramos y asignarle un texto en tiempo de diseño o de ejecución esta vez sin utilizar la propiedad text puesto que no la incorpora, sino utilizando la propiedad caption.

Este control sirve para mostrar mensajes en nuestro formulario que orienten al usuario sobre la utilidad de los demás controles que tengamos en la aplicación o para indicarnos acciones que podemos realizar. En el ejemplo anterior donde aparecía un textbox en el formulario, hubiera quedado mejor con un mensaje aclaratorio contenido en un control label:

PROPIEDADES

Caption: Es el texto que contendrá el control.

Alignment: Alineación del texto contenido en el control, no necesita que esté activada ninguna otra propiedad.

BorderStyle: Si queremos que aparezca un borde alrededor del control activaremos esta propiedad.

Para este control no se suelen utilizar los eventos ya que su contenido suele cambiar poco a lo largo de la ejecución de la aplicación. De todas formas los eventos son casi los mismos del control textbox excepto que no dispone de los eventos **GetFocus** y **LostFocus** ya que a este control no se le puede dar el enfoque.

En la parte final de este capitulo veremos un ejemplo donde se muestra el funcionamiento de todos los controles que vamos a ir viendo.Por ahora a ver si conseguis que ahora el mensaje no aparezca en el formulario sino en un segundo label situado en el formulario, dejando un control label que muestre el mensaje aclaratorio que hemos visto antes.

CommandButton

Este control es el típico botón que aparece en todas las aplicaciones y que al hacer click sobre él nos permite realizar alguna operación concreta, normalmente Aceptar o Cancelar. Aunque según el código que le asociemos podremos realizar las operaciones que queramos.

En el ejemplo anterior podemos añadir un control de este tipo para salir de la aplicación sin tener pulsar sobre la equis de la esquina superior derecha.

Pero sólo con introducir un control de este tipo con el texto salir que se introduce a traves de la propiedad caption no basta. Habrá que asociarle un código que nos permita salir de la aplicación en el evento adecuado. Y el evento por excelencia de este control es click. Así pues accederemos al código del control y la sentencia nos permitirá salir de la aplicación es End, simplemente tecleamos esa palabra en el evento click y comprobar que realmente finalizaremos nuestra aplicación al pulsar sobre dicho botón.

PROPIEDADES

Caption: Aqui pondremos el letrero que queremos que apaezca en el botón: aceptar, cancelar, salir, etc.

Enabled: Esta es una nueva propiedad, cuando su valor es true el botón funciona normalmente, cuando su valor es false el boton se encuentra desactivado, no responde a los eventos producidos sobre él y el texto aparece en un gris claro advirtiendonos de su estado. Podemos utilizar esta propiedad para activar o desactivar un boton dependiendo del estado de otros controles. Por ejemplo, en un boton Aceptar, no activarlo hasta que se haya introducido una cantidad en un control textbox, ya que ese botón nos calculará el IVA de la cantidad.

EVENTOS

Click: Es el evento típico de este control y el que más se utiliza.

MouseMove: Como sabemos detecta el movimiento del ratón sobre el control. Puede servir para que aparezca un mensaje en un control Label que nos aporte información sobre la utilidad del control ampliando el texto que hayamos colocado como caption del commandbutton.

OptionButton

Este control nos permite elegir una opción entre varias de las que se nos plantean. Cada opción será un control optionbutton diferente.

Facilita la introducción de datos por parte del usuario:

De todas las opciones que se nos ofrece, en este caso los 4 colores, sólo podremos activar una. Si activamos cualquier otra opción, se desactivará automáticamente la última que teníamos activada.

El marco que está alrededor de los 4 controles optionbutton se trata del control **Frame** , es opcional, aunque es conveniente colocarlo siempre que hagamos uso de las opciones. No sólo por motivos de presentación sino porque de esta manera podremos establecer grupos de controles optionbutton independientes en los que en cada grupo sólo pueda haber una opción activada a la vez. También, al mover el marco se moverán los controles incluidos en él facilitándonos las modificaciones.

Para que los controles Optionbutton queden englobados dentro de un control Frame, primero tendremos que colocar el control Frame en el formulario con el tamaño adecuado y despues ir colocando los controles Optionbutton dentro del Frame.

Del control Frame la única propiedad que nos interesará es caption, que es el texto que aparecerá en el encabezado, en el ejemplo anterior: colores.

PROPIEDADES DE OPTIONBUTTON

Caption: El texto que aparecerá al lado del control: Rojo, verde, etc.

Value: Es el valor que tendrá el control: True si se encuentra activado y False si no lo está. Para comprobar que opción ha activado el usuario comprobaremos el estado de esta propiedad.

Alignment: Alineación del texto respecto al control: Left Justify: el control aparece a la izquierda del texto. Es el ejemplo anterior.Right Justify: el control aparece a la derecha del texto.

Los eventos del control son los mismos que en anteriores controles, aunque no se suele asociar código a los

eventos de este tipo de controles, sino únicamente conocer el valor que tienen: true o false.

Aplicación de ejemplo

Para practicar con los controles que hemos visto vamos a realizar una pequeña aplicación que consistirá en realizar con 2 números que introduzcamos, una operación que seleccionemos y mostrar el resultado.

El formulario donde estarán todos los controles es el siguiente:

La propiedad **Caption** de cada uno de los controles es la que se muestra en el formulario.

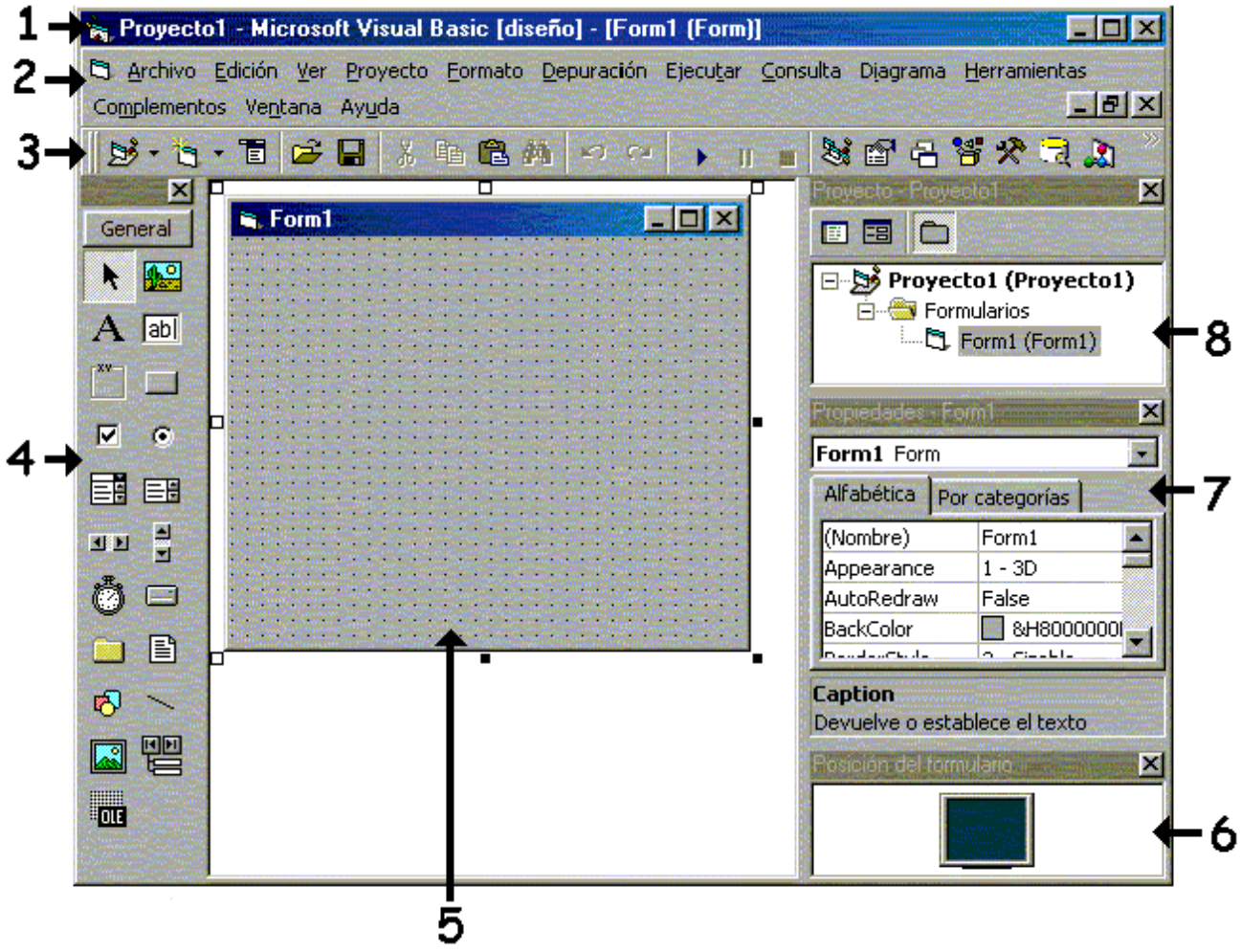
He modificado la propiedad **Name** de cada control para que al utilizarlos desde el codigo sepamos cual es el control con el que trabajamos:

- Los controles TextBox tienen los nombres: **Num1**, **Num2** y **Resul.**
- Los controles Optionbutton tienen cada uno de ellos el mismo nombre que su caption
- Los controles CommandButton tienen los nombres: **Calcular**, **Limpiar** y **Salir.**
- A los controles Label y al Frame no have falta cambiarles el nombre.

Lo que habrá que hacer ahora es asociar codigo a cada uno de los botones que es de donde se van a realizar las operaciones:

- Para el botón **Calcular** que es el que nos mostrará el resultado según la operación seleccionada, he utilizado la instrucción If Then Else que vimos en el capítulo anterior:

- El botón **Limpiar Datos** nos va a servir para borrar de una forma rápida los datos introducidos por el usuario y el resultado preparando los controles para introducir nuevos datos. El código que tendremos que introducir es muy simple
- El botón **Salir** únicamente contendrá la sentencia End.


1
Proyecto1 - Microsoft Visual Basic [diseño] - [Form1 (Form)]
2
Archivo Edición Ver Proyecto Formato Depuración Ejecutar Consulta Diagrama Herramientas
Complementos Ventana Ayuda
3
General
4
Form1
5
Proyecto - Proyecto1
Proyecto1 (Proyecto1)
Formularios
Form1 (Form1)
8
Propiedades - Form1
Form1 Form
Alfabética
Por categorías
7
(Nombre) Form1
Appearance 1 - 3D
AutoRedraw False
BackColor &H8000000
Caption
Devuelve o establece el texto
Posición del formulario
6