FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

A LOS ASISTENTES A LOS CURSOS

Las autoridades de la Facultad de Ingeniería, por conducto del jefe de la División de Educación Continua, otorgan una constancia de asistencia a quienes cumplan con los requisitos establecidos para cada curso.

El control de asistencia se llevará a cabo a través de la persona que le entregó las notas. Las inasistencias serán computadas por las autoridades de la División, con el fin de entregarle constancia solamente a los alumnos que tengan un mínimo de 80% de asistencias.

Pedimos a los asistentes recoger su constancia el día de la clausura. Estas se retendrán por el periodo de un año, pasado este tiempo la DECFI no se hará responsable de este documento.

Se recomienda a los asistentes participar activamente con sus ideas y experiencias, pues los cursos que ofrece la División están planeados para que los profesores expongan una tesis, pero sobre todo, para que coordinen las opiniones de todos los interesados, constituyendo verdaderos seminarios.

Es muy importante que todos los asistentes llenen y entreguen su hoja de inscripción al inicio del curso, información que servirá para integrar un directorio de asistentes, que se entregará oportunamente.

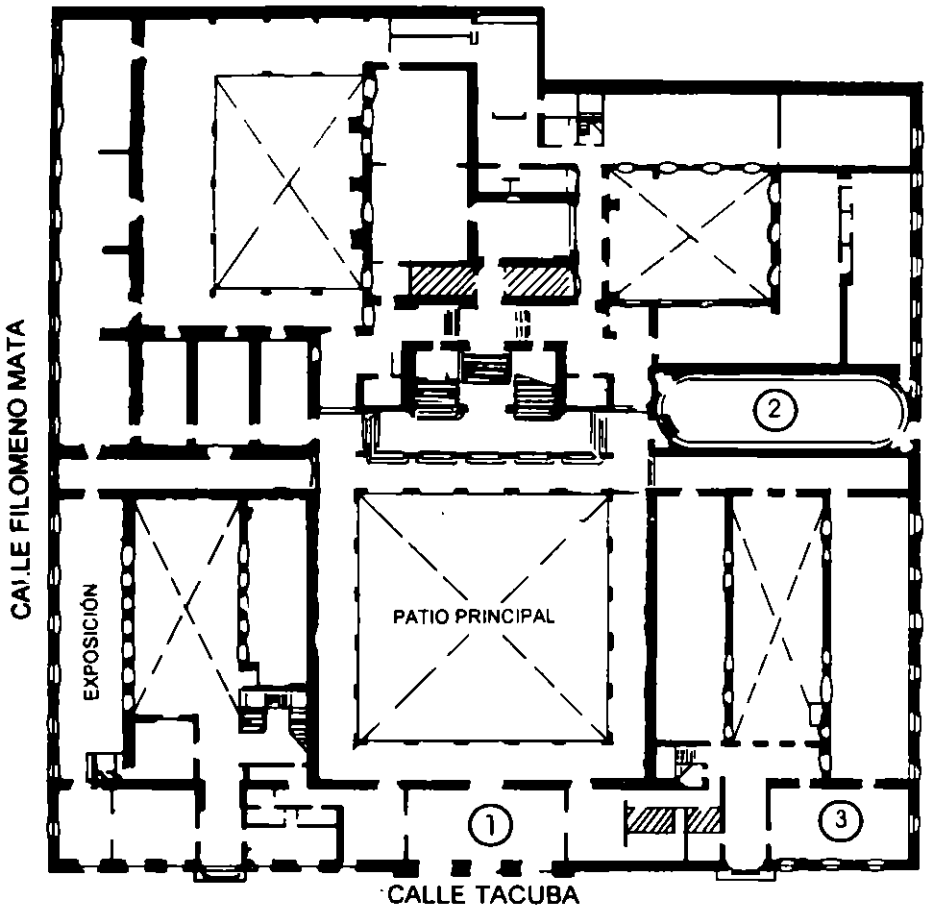
Con el objeto de mejorar los servicios que la División de Educación Continua ofrece, al final del curso deberán entregar la evaluación a través de un cuestionario diseñado para emitir juicios anónimos.

Se recomienda llenar dicha evaluación conforme los profesores impartan sus clases, a efecto de no llenar en la última sesión las evaluaciones y con esto sean más fehacientes sus apreciaciones.

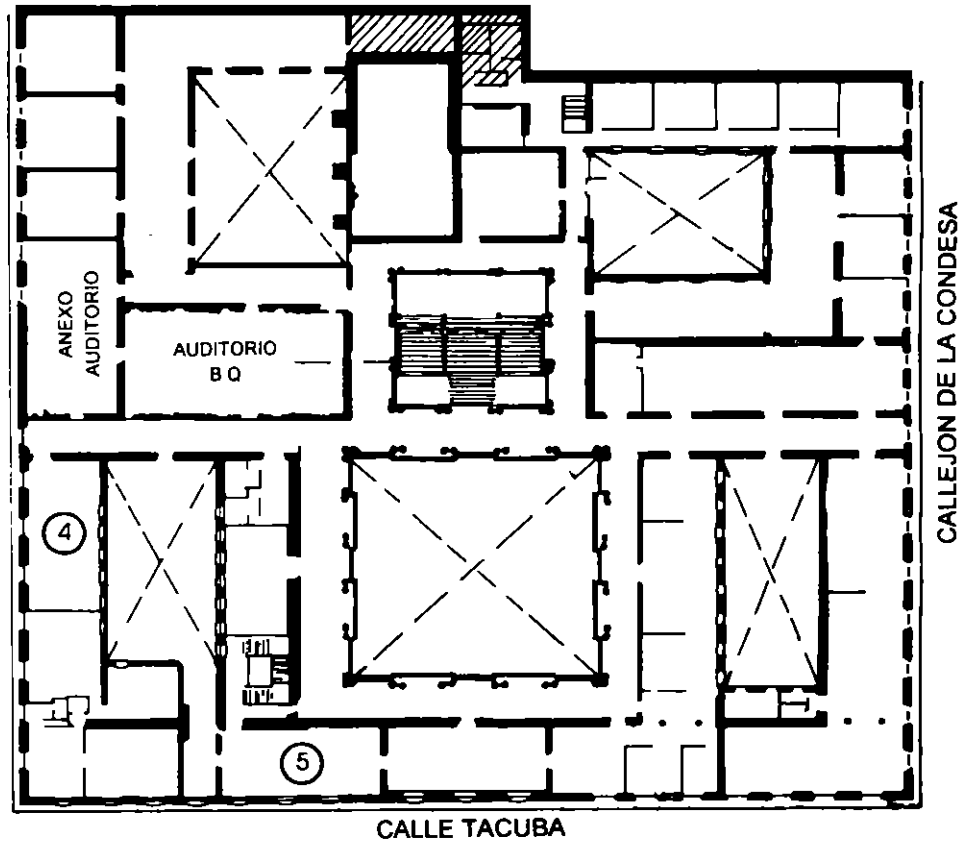
Atentamente
División de Educación Continua.

Palacio de Minería Calle de Tacuba 5 Primer piso Deleg. Cuauhtémoc 06000 México, D.F. APDO. Postal M-2285
Teléfonos: 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

PALACIO DE MINERIA

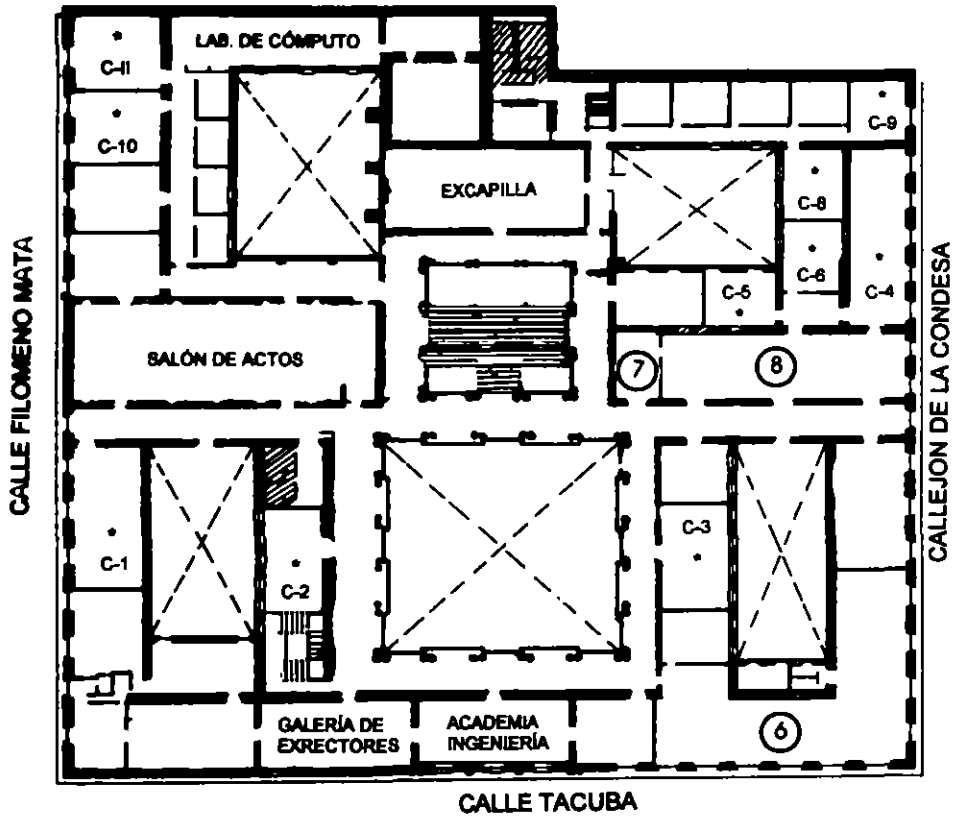


PLANTA BAJA



MEZZANINNE

PALACIO DE MINERIA



Ier. PISO

GUÍA DE LOCALIZACIÓN

1. ACCESO
2. BIBLIOTECA HISTÓRICA
3. LIBRERÍA UNAM
4. CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN "ING. BRUNO MASCANZONI"
5. PROGRAMA DE APOYO A LA TITULACIÓN
6. OFICINAS GENERALES
7. ENTREGA DE MATERIAL Y CONTROL DE ASISTENCIA
8. SALA DE DESCANSO

SANITARIOS

* AULAS

DIVISIÓN DE EDUCACIÓN CONTINUA
FACULTAD DE INGENIERÍA U.N.A.M.
CURSOS ABIERTOS

DIRECTORIO DE ASISTENTES AL CURSO DE

VISUAL BASIC 3.0. PROFESSIONAL

APELLIDOS	NOMBRE	EMPRESA	CALLE	COLONIA	DELEGACION	TELEFONO
ASTORGA DE RIQUER	AGUSTIN	F I UNAM	CIRCUITO INTERIOR	CD UNIVERSITARIA	COYOACAN	6223119
CANO GARCIA	JOSE LUIS	F I UNAM	CIRCUITO INTERIOR	CD UNIVERSITARIA	COYOACAN	6223119
COLMENARES	ALBERTO	F I UNAM	CIRCUITO INTERIOR	CD UNIVERSITARIA	COYOACAN	67654240
CHIANG SAM GARCIA	LUIS	PEMEX	CARR MEX-PIRAMIDES KM30.5	VTA. DE CARPIO	EDO DE MEXICO	7222500
DELGADO LUCIO	NICOLAS	D G C F, S.C.T.	ALTADENA 23	NAPOLES	BENITO JUAREZ	6733384
FRAGOSO HERNANDEZ	ANDRES					7669459
FRONTANA URIBE	ARMANDO					5615871
GARCIA YESCAS	LESTER	PEMEX	CARR MEXICO-PIRAMIDES KM30 5	VENTA DE CARPIO	EDO DE MEXICO	7222500
GOMEZ EDALTOS	OSCAR R.	PEMEX				3797295
HERNANDEZ MELO	PILAR	SECRETARIA DE MARINA	H ESC NAV MIL #861	LOS CIPRESES	COYOACAN	6848188
HERNANDEZ SANIVAN	ELEAZAR	ENEP ACATLAN UNAM	AV ALCANFORES Y SN JUAN T.	STA CRUZ ACATLAN	NAUCALPAN	6231743
LARA GARCIA	GRACIELA	D G G - I N E G I	PATRIOTISMO 711	SN JUAN MIXCOAC	BENITO JUAREZ	6150884
MARTINEZ MEDELLIN	JESUS	SECRETARIA DE MARINA	EJE DOS OTE TRAMO H E N 861	LOS CIPRESES	COYOACAN	6848188
MARTINEZ NUÑEZ	GREGORIO	SECRETARIA DE MARINA	EJE 2 OTE 861	LOS CIPRESES	COYOACAN	6796411
MORAN PEÑA	GABRIEL	ENFERMERIA Y OBSTETRICIA	CAM ANT A XOCHIMILCO	HUIPULCO	TLALPAN	6552298
NAVARRO CALLEJAS	JORGE	B A N A M E X				5866712
PEREZ MIGUEL	GUSTAVO	BANCO DE MEXICO	F A.IXTLIXOCHITL 182	TRANSITO	CUAUHTEMOC	7265918
PEREZ RODRIGUEZ	ERNESTO	PEMEX				7946236
RAMIREZ JIMENEZ	JUDITH	HOSPITAL INFANTIL DE MEXICO	DR MARQUEZ 62	DOCTORES	CUAUHTEMOC	7610333
ROSADO ORTEGA	RICARDO	SISTEMAS TADIRAN SA DE CV	AV SN ESTEBAN 2	EL PARQUE	NAUCALPAN	3592811
RUIZ CERDA	JORGE					5399518
SALAZAR SOTO	MARIA	D G G - I N E G I	PATRIOTISMO 711	SN JUAN MIXCOAC	BENITO JUAREZ	6150884
SANCHEZ JUAREZ	HERIBERTO	PARTIDO ACCIONAL NACIONAL	ANGEL URRAZA 812	DEL VALLE	BENITO JUAREZ	5596300
URBINA HERNANDEZ	JORGE	BANCO DE MEXICO	F A IXTLIXOCHITL 182	TRANSITO	CUAUHTEMOC	7265918
VAZQUEZ LORENZANA	CESAR					5449294

FACULTAD DE INGENIERIA U.N.A.M.
DIVISION DE EDUCACION CONTINUA

PROGRAMACION DE APLICACIONES WINDOWS CON VISUAL BASIC 3.0 PROFESSIONAL

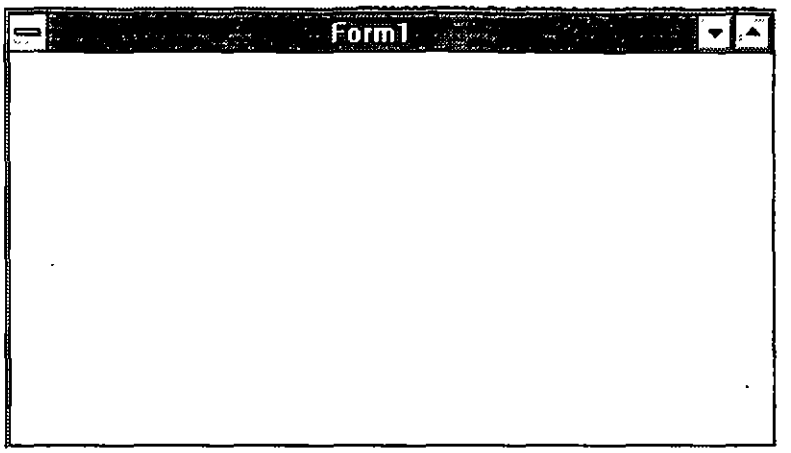
MATERIAL DIDACTICO (COMPLEMENTO)

sept.19-oct.7 de 1995

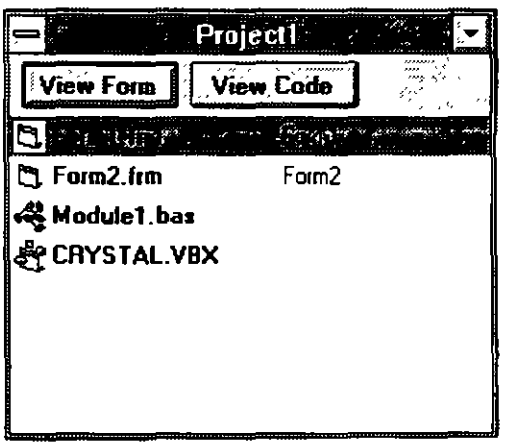
Palacio de Mineria Calle de Tacuba 5 Primer piso Deleg. Cuauhtemoc 06000 Mexico, D.F. APDO Postal M-2285
Telefonos 512-8955 512-5121 521-7335 521-1987 Fax 510-0573 521-4020 AL 26

I. INTRODUCCIÓN

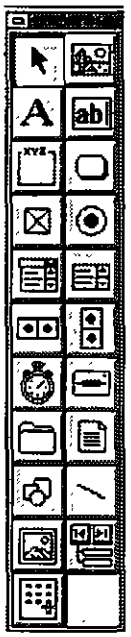
Forma. Una forma es la parte principal de una aplicación basada en gráficos. Es lo que el usuario ve y con lo cual interactua para llevar a cabo alguna tarea. También es el lugar en donde se comienza a construir una aplicación, es decir, donde se colocan los controles y se construye la interface.



Ventana de proyecto. Es una lista de archivos que Visual Basic usa para llevar el control de las formas y archivos que conforman una aplicación, en ella se encuentran archivos .FRM asociados a las formas, .VBX asociados a los controles de la versión profesional de Visual Basic y .BAS los cuales sólo contienen código.

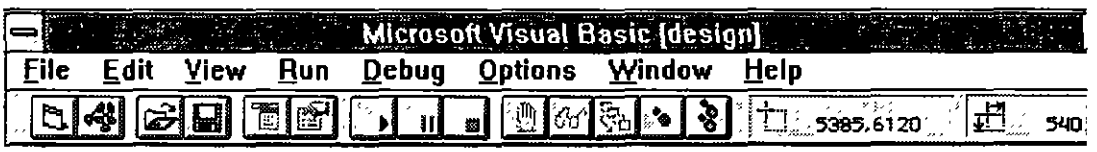


Caja de herramientas. Es el lugar en donde se encuentran todos los posibles controles que se pueden dibujar en una aplicación.



Existen dos maneras de colocar controles en una forma, la primera es por medio de un doble click en el control que se desee dibujar, con esto automáticamente el control se colocará en medio de la forma y con un tamaño ya definido. Otro método es por medio de un click, y arrastrando el apuntador del mouse sobre la forma hasta que el control tenga el tamaño y posición deseada.

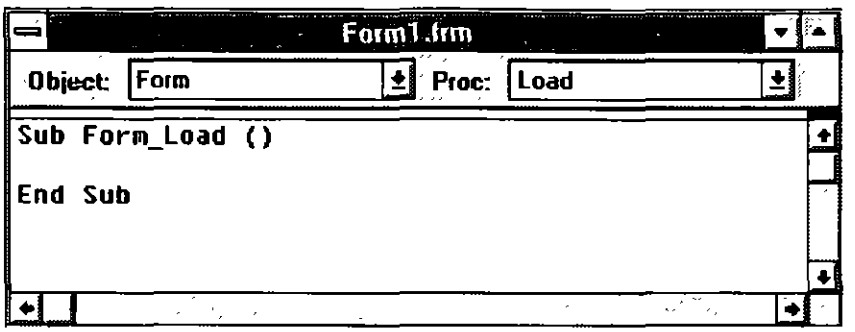
Barra de herramientas. La barra de herramientas, permite un acceso rápido a las funciones y comandos comunes de Visual Basic, estas funciones y comandos también están disponibles en los menús y además pueden ser accesados por medio de teclas especiales.



Los elementos más relevantes dentro de la barra de herramientas son:

- Nueva Forma
- Nuevo Módulo
- Abrir Proyecto
- Salvar Proyecto
- Ventana de Diseño de Menús
- Ventana de Propiedades
- Comenzar ejecución (Run)
- Detener ejecución (Break)
- Fin de ejecución (End)

Ventana de código. En este tipo de ventana se despliega el código que implementa una aplicación. Al principio las ventanas de código solo contienen un template para los procedimientos y funciones, en estos templates es necesario agregar el código necesario para que una aplicación funcione. Para poder ver la ventana de código se debe dar un doble click sobre el objeto o control que se desea ver su ventana de código.



II. CONSTRUCCIÓN Y DISEÑO DE APLICACIONES EN VISUAL BASIC

Visual Basic maneja lo que se conoce como la programación manejada por eventos, esto significa que el usuario puede disparar ciertos eventos (al mover el mouse, o dar un click sobre algún control, etc) o el sistema operativo en este caso Windows. Debido a esto la programación que maneja Visual Basic no es un tipo de programación procedural, porque no tiene un secuencia top-down (arriba-abajo).

La aplicaciones basadas en windows son multitarea. Pueden compartir el espacio de la pantalla y el tiempo de procesador.

Las aplicaciones basadas en windows también se basan en elementos gráficos, a diferencia de las aplicaciones basadas en MS-DOS las cuales están basadas en caracteres.

Terminología de Visual Basic

Visual Basic tiene dos grandes componentes: el lado gráfico y el lado del código. Viendo esto en forma tabular tendría la siguiente forma:

- Aplicación
  - Formas
    - Propiedades
    - Eventos
  - Controles
    - Propiedades
    - Eventos

Visual Basic ofrece a los programadores objetos (formas y controles) que pueden usarse para construir aplicaciones. En la mayor parte, el programador solo necesita usar el objeto sin tener que conocer necesariamente en forma detallada como funciona el objeto.

Controles. Los controles son las herramientas que se colocan en las formas para dar a una aplicación funcionalidad. Por ejemplo: los botones de comando, las etiquetas, las cajas de combo, etc.

Propiedades. Los controles y las formas tienen propiedades o atributos que se pueden cambiar en tiempo de diseño y en tiempo de corrida.

Procedimientos de eventos. Los procedimientos de eventos son códigos internos que Visual Basic y Windows nos dan para darle funcionalidad a una aplicación.

Desarrollo de aplicaciones

Para crear una aplicación en Visual Basic se sugiere la siguiente secuencia:

1) Abra un nuevo proyecto (o utilice el nuevo proyecto creado cuando comienza VISUAL BASIC) para organizar las partes de su aplicación.
2) Cree una forma para cada ventana de su aplicación.
3) Dibuje los controles para cada forma.
4) Cree la barra de menú para la forma principal.
5) Coloque a las formas y a los controles sus propiedades.
6) Escriba los procedimientos de eventos y los procedimientos generales.
7) Salve su trabajo.
8) Corra por pasos su código.
9) Cree un archivo ejecutable para convertir su proyecto en una aplicación.

Distribuyendo una aplicación

Cuando usted pretenda distribuir su archivo ejecutable a diferentes usuarios, necesita distribuir una copia de la librería de ligado dinámico de Visual Basic **VBRUN300.DLL** junto con él. La librería de ligado dinámico (dynamic-link library DLL) es parte de los archivos de instalación de VISUAL BASIC y es posible distribuirla libremente. Si al construir alguna aplicación se hizo uso de los custom controls disponibles, también es necesario agregar todos los archivos .VBX apropiados y en algunos casos éstos también necesitan algunos archivos DLL; la documentación de cada uno de los custom controls le indicará cuales son los archivos DLL que necesita distribuir junto con el custom control. El archivo VBRUN300.DLL es un archivo que siempre se tendrá que distribuir sin embargo dentro de la aplicación es probable que se utilice algún otro archivo DLL por lo que en este caso también será necesario agregar el correspondiente archivo DLL a los demás archivos de distribución.

Finalmente durante el diseño de su aplicación es posible que necesite mover formas de su aplicación a otra máquina. En este caso, lo más probable es que necesite reconstruir el archivo .MAK, porque éste guarda todas los nombres de los archivos de su aplicación así como sus correspondientes rutas en disco duro. Al hacer esto, copie todos los archivos a la nueva máquina, entonces actualice el path en el archivo .MAK utilizando el comando Remove File del menú de File y después agréguelo nuevamente con el comando Add File.

Diseño de interfaz.
Pautas generales

§ **Diseño básico**

* Diseñado por el usuario, no por el sistema lo cual permite composición y funcionalidad.
* Control del usuario
* Directo
* Consistencia
* Claridad
* Estética
* Autoalimentación

§ **Color**

* Color para llamar la atención del usuario
* Complementar los colores

§ **Fonts**

* Serif versus san-serif
* Tamaño
* Número (variedad)

Obteniendo entradas del mouse

Existe un limitado número de cosas que el usuario puede hacer con un mouse:

1) **Click simple** con cualquiera de los botones primario o secundario.
2) **Doble Click** con cualquiera de los botones primario o secundario.
3) **Arrastre** o **Drag** normalmente con el botón primario.
4) **Drag y Drop** (arrastre con un botón presionado y liberación del botón)

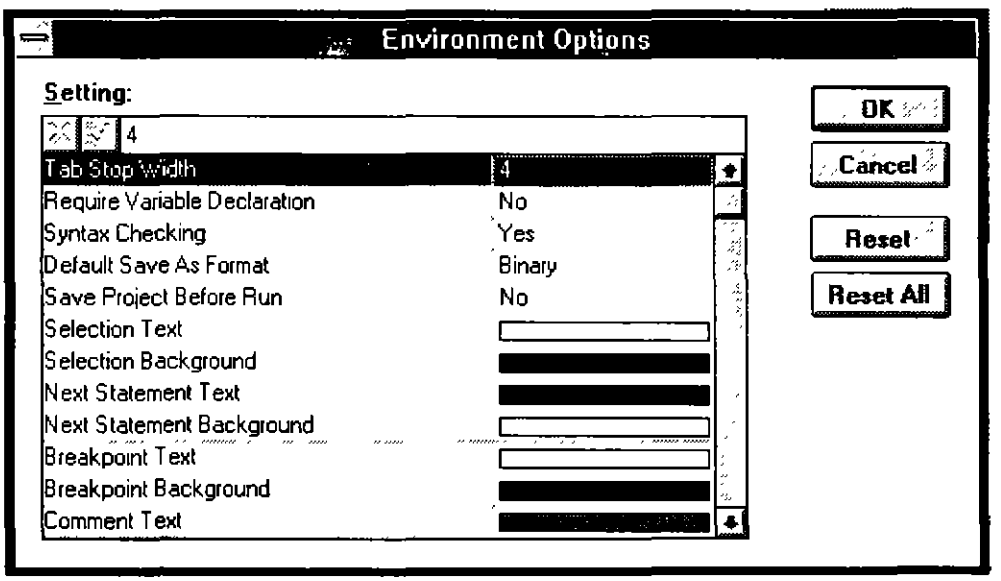
Por ejemplo, un **Click simple** con el botón primario del mouse indica que se ha escogido alguna opción o quizá se ha seleccionado el nombre de un archivo que será abierto.

Un **Doble Click** significa una selección y realización de algún comando en especial.

Ambiente

Mucho del ambiente de Visual Basic puede configurarse, esto se puede realizar a través del menú options y el submenú environment, las opciones que pueden ser configuradas son las siguientes:.

Tab Stop Width
Syntax Checking
Save Project Before Run
Selection Background
Next Statement Background
Breakpoint Background
Comment Background
Keyword Background
Identifier Background
Code Window Background
Grid Height
Align to Grid
Require Variable Declaration
Default Save As Format
Selection Text
Next Statement Text
Breakpoint Text
Comment Text
Keyword Text
Identifier Text
Code Window Text
Grid Width
Show Grid



Autoload.mak

Este proyecto es el que carga Visual Basic cada vez que se inicia un nuevo proyecto. Este archivo puede ser editado para determinar que archivos se desea que aparezcan al momento de cargar Visual Basic. Para editarlo se puede usar cualquier editor ASCII.

III. FORMAS

Las formas son la parte principal de una aplicación en Visual Basic. Es el lugar en donde se colocan los controles y se realiza la interface. Las formas así como los controles tiene propiedades por ser objetos. Algunas de las propiedades más importantes son las siguientes:

Propiedad	Descripción
BorderStyle	Establece el estilo del borde
Caption	Título de la forma
ControlBox	Despliega el Control Box
FontSize	Tamaño del Font
Name	Nombre usado en código. Prefijo frm
Height	Altura de la forma
Icon	Icono para el ejecutable
KeyPreview	Determina quién recibirá primero los eventos del teclado
Left y Top	Determinan la esquina superior izquierda de la forma
MaxButton	Deshabilita el botón de maximizar
MinButton	Deshabilita el botón de minimizar
MousePointer	Determina la forma del cursor
Visible	Hace visible la forma
Width	Determina el ancho de la forma

Eventos

Load. Este evento ocurre cuando la forma es cargada en memoria. Normalmente este evento se utiliza para inicializar variables o propiedades.

Unload. Este evento ocurre cuando una forma esta siendo descargada, es decir, removida de memoria.

Métodos

Hide. Este método se usa para remover una forma de la pantalla, es decir, pone su propiedad visible en false. Al usar hide, solo se remueve la forma de pantalla, pero aún sigue cargada en memoria.

Show. Este método se utiliza para desplegar una forma que ha sido escondida a través del método hide.
NOTA: Si se utiliza show sobre una forma que no ha sido carga en memoria, show automáticamente carga la forma en memoria y después la muestra.

Cajas de mensaje (Message Box)

Las aplicaciones de Visual Basic están compuestas por formas, y generalmente se utilizan algunas de estas para un uso específico (formas para mandar mensajes, etc). Para este tipo de formas que son de uso cotidiano, existe una función en Visual Basic que permite manejarlas, esta función es: **Msgbox** , cuya sintaxis es la siguiente:

Msgbox "Mensaje a desplegar", Tipo de forma, "Titulo de la ventana"

en donde el tipo de forma determina el número de botones a desplegar y con que caption, además de poder desplegar algún icono sobre la forma.

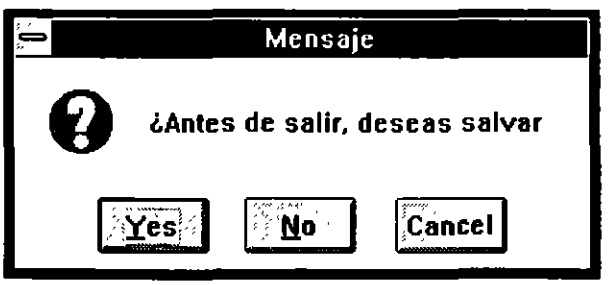
Por ejemplo, si se desea desplegar una caja de mensaje con la siguiente información:

Mensaje: ¿Antes de salir, deseas salvar?
Tipo de forma: 3 botones, Si, No, Cancel y un icono de signo de interrogación
Caption o título: Mensaje

La función MsgBox se llamaría de la siguiente forma:

Msgbox "¿Antes de salir, deseas salvar?, 3 + 323, "Mensaje"

Y la forma que se desplegaría sería la siguiente:



Tipos de cajas de diálogo

Modeless. Las cajas de diálogo modeless, son las que da Visual Basic por default. Esto quiere decir, que el usuario puede interactuar entre las diferentes formas que componen la aplicación de Visual Basic sin ninguna restricción.

Modal. Son las cajas de diálogo que no permiten interactuar con las demás ventanas de la aplicación hasta que se realice una acción sobre la ventana activa. Pero si puede interactuar con las demás ventanas del sistema. Un ejemplo puede ser, la caja de diálogo que se despliega al momento de abrir un archivo: File / Open File.

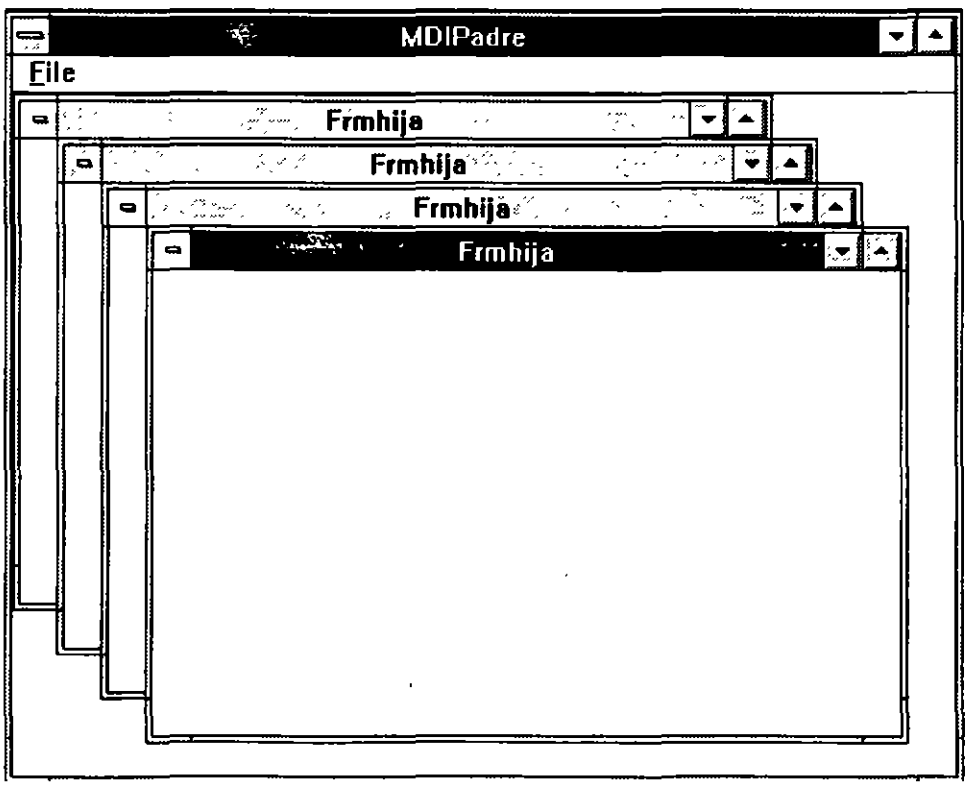
System Modal. Son las cajas de diálogo o formas que no permiten que se interactue con ningún tipo de ventana (ya sea de la misma aplicación o del sistema) si no se ha realizado ninguna acción sobre la ventana activa.

MDI. Multiple Document Interface

MDI, permite que una aplicación en Visual Basic pueda crear múltiples copias de una forma y desplegar todas las formas dentro de un solo contenedor.

Las características de MDI son las siguientes:

1. Existe una forma llamada padre o contenedor. Por aplicación Visual Basic solo permite una forma padre o contenedor
2. Las formas hijas, se crean siempre dentro del contenedor, y no pueden salir de el, también se minimizan dentro del contenedor.
3. Dentro de una forma padre, no solo pueden existir formas hijas, también pueden existir formas normales.



IV. MENÚS

Los menús son una de las principales formas para que el usuario pueda invocar comandos o invocar una caja de diálogo común que le ofrece al usuario más opciones.

Los menús también son utilizados por el programador como una forma de estructurar una aplicación.

Una aplicación con menús generalmente comienza con tres menús básicos que son:

File, Edit y Help.

Dentro del menú File se tienen generalmente los siguientes submenus: New, Open, Save, Save As, Print, Print Setup, Repaginate y Exit.

El menú para salir de una aplicación generalmente se coloca al final del primer menú. La ayuda normalmente se encuentra como el último menú de la derecha.

El menú Edit normalmente contiene por lo menos los siguientes comandos: Undo, Cut, Copy y Paste.

Finalmente el menú Help contiene por los menos las siguientes opciones: Contents, Search for Help on..., How to Use Help y About *nombre_aplicación*.

Características principales de los menús

Puntos suspensivos

Los puntos suspensivos al final de un comando de un menú indican que no es un comando terminal, es decir, que el usuario accesará una caja de diálogo con más opciones cuando seleccione este tipo de menús.

Opciones marcadas (Check Options)

Son aquellos elementos de un menú que aparecen con una marca de "check", para indicar que es la opción que el usuario tiene seleccionada actualmente.

Separación de elementos

Generalmente los elementos de un menú se encuentran separados en grupos, estas separaciones se realizan a través de una línea horizontal.

Teclas de acceso

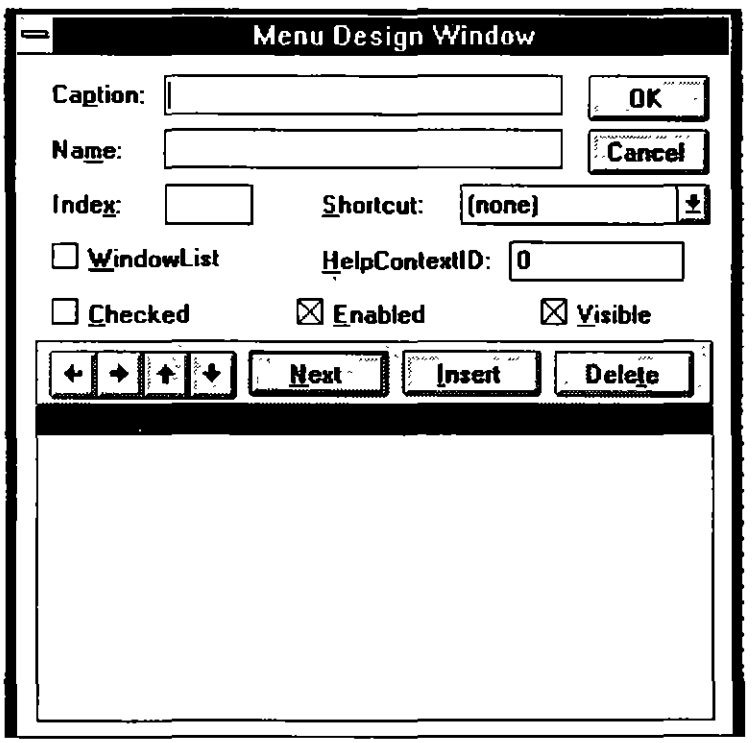
Los menús y los elementos de un menú generalmente tienen una letra marcada como subrayada, esta letra permite el acceso a estos menús a través de la combinación ALT + letra subrayada.

Shortcut

Esta es otra manera de accesar un menú a través del teclado. Para usar este tipo de teclas, el usuario necesita presionar ya sea una tecla o una combinación de teclas. Por ejemplo:

F1	Ayuda
CTRL+X	Cortar texto seleccionado (cut)

Construcción de menús en Visual Basic



Para implementar menús es necesario abrir la ventana de diseño de menús, esta se puede abrir a través de el menú Window y el submenu Menú Design o también presionando la combinación de teclas Ctrl-M.

NOTA: Para poder accesar una ventana de diseño de menús, es necesario tener seleccionada una forma, de otra manera no es posible abrir esta ventana.

La ventana de diseño de menús tiene las siguientes opciones:

Elemento	Descripción
Caption	Texto que se desea que aparezca en el menú.
Name	Nombre usado en código. Prefijo: mnu
Index	Usado para agregar menús dinámicamente
Shortcut	Ctrl+shortcut
Window List	Especifica si un menú incluirá una lista de formas MDI abiertas
HelpContexID	Especifica un identificador para la ayuda del sistema
Checked	Indica si la marca de check se desplegará en la parte izquierda del elemento del menú
Enabled	Indica si un menú estará habilitado o deshabilitado
Visible	Determina si un menú será visible o no para el usuario
FLECHA IZQ	Eleva un elemento del menú un nivel. Por ejemplo, hace de un submenu un menú principal
FLECHA DER	Baja un elemento del menú un nivel. Por ejemplo, hace de un menú principal un submenu
FLECHA ARRIBA	Mueve un elemento del menú una posición arriba
FLECHA ABAJO	Mueve un elemento del menú una posición abajo
Next	Mueve el cursor un elemento abajo de la lista de menús. También limpia el Caption y Name para poder agregar un nuevo elemento.
Insert	Inserta una línea en blanco en la lista de menús para que se puedan agregar nuevo elementos
Delete	Borra el elemento seleccionado de la lista.

Para pode implementar teclas de acceso sobre un menú, es necesario colocar un ampersand (&) en el caption adelante de la letra que se desea que aparezca subrayada.

V. USO DE MÚLTIPLES FORMAS

Generalmente una aplicación en Visual Basic esta compuesta de múltiples formas, por lo tanto, es necesario conocer la manera en la cual se pueden accesar cada una de las forman que conforman la aplicación. Para esto es necesario conocer algunas métodos y sentencias:

Sentencias

Load *objeto*. Esta sentencia es usada para cargar una forma o control en memoria. Normalmente se coloca dentro de otros eventos.

NOTA: Load no despliega automáticamente la forma en pantalla, solo la carga en memoria. Para desplegar una forma se utiliza la sentencia Show.

Unload *objeto*. Esta sentencia se usa para remover una forma de memoria. Descargar una forma es útil cuando es necesario liberar espacio en memoria o se quiere volver al valor inicial de las propiedades de la forma. Uno de sus usos es en el botón OK de las formas.

End. Esta sentencia por si sola termina la ejecución de una aplicación en Visual Basic, cierra todos los archivos, limpia los valores de todas las variables y destruye todas las formas.

Métodos

Hide. Este método remueve de la pantalla una forma o un control, es decir, afecta la propiedad visible y la pone en falso.

Sintaxis:

[Form.]Hide

Show[Estilo]. Este método muestra una forma en pantalla, es decir, su propiedad visible la coloca en True. Por si solo este método carga en memoria y muestra una forma.

Sintaxis:

[Form.]Show

El estilo puede determinar el tipo de forma que queremos que se muestre, este estilo puede ser: modal, modless o modal al sistema.

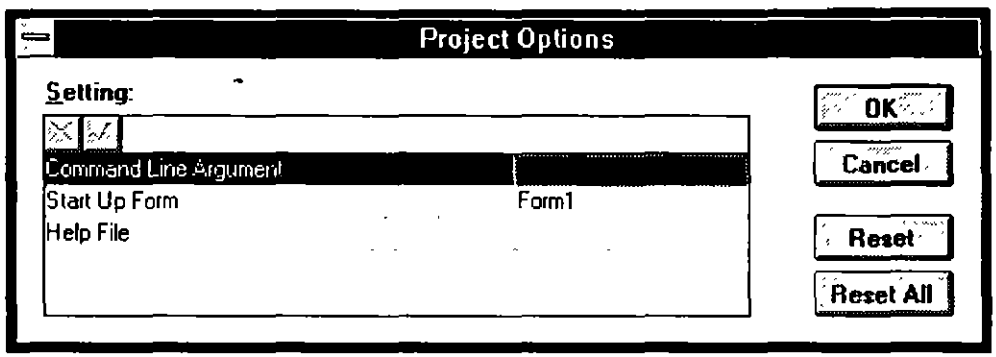
Procedimientos de eventos

Los procedimientos de eventos son los que son invocados por el usuario o el sistema operativo. Son todos los que observamos en la ventana de código, al dar un doble click sobre algún objeto. Los procedimientos de eventos ya tienen un template (plantilla) construido.

Forma de inicio (Start up form)

Generalmente una aplicación no costa de una sola forma, sino de dos o más. Visual Basic por default pondrá como forma de inicio a aquella que fue creada primero.

Si se desea cambiar esta forma de inicio (o de startup), se realiza a través del menú options, project, start up form.



VI. CONTROLES

Dentro de la versión profesional de Visual Basic existen diversos controles. Una de las principales deferencias entre la versión profesional y la estándar son precisamente los controles.

La versión profesional de Visual Basic incluye todos los controles de la versión estándar y controles propios de la versión profesional.

Dentro de este módulo se nombrarán las propiedades más importantes de estos controles, y que efecto tienen sobre los mismos.

Cabe hacer notar que la versión profesional de Visual Basic contiene varios controles, lo cual hace difícil la tarea de explicar cada uno de ellos, por lo tanto solo se tratarán los más importantes.

Como referencia, dentro del grupo de Visual Basic existe un icono marcado como Custom Control Help, en el cual se encuentra la ayuda para todos los controles propios de la versión profesional, para obtener ayuda sobre otros controles se puede hacer directamente a través del menú Help de Visual Basic.

Agregando controles a la caja de herramientas

Para agregar un control a la caja de herramientas se realizan los siguientes pasos:

1. Del menú File seleccionar Add File. Una caja de diálogo común para agregar archivos se despliega
2. De la caja de diálogo localizar el archivo del control que se desea agregar. Todos los archivos de controles se encuentran siempre en \WINDOWS\SYSTEM, y tienen la extensión VBX.
3. Seleccionar el archivo .VBX y seleccionar OK. En este momento el nuevo control es agregado a la caja de herramientas en forma de uno o varios iconos y además el archivo correspondiente se ha agregado a la ventana de proyecto.
4. Usar el mismo procedimiento para agregar más controles.

ETIQUETAS

Propiedad	**Comentario**
Alignment	Determina la alineación del texto
AutoSize	Determina si el control se ajustará a su contenido

BackColor	Determina el color de background
BorderStyle	Determina el estilo del borde
Caption	Determina el texto a desplegar
FontName	Determina el font a usar
FontSize	Determina e tamaño del font a usar
ForeColor	Determina el color de la letras
Height	Determina la altura del control
Left	Posición del borde izquierdo de la forma
Name	Nombre del control a usarse en código. Prefijo usado lbl.
TabIndex	Determina el orden del tabulador
Tag	Propiedad que tienen la mayoría de los controles. El usuario define su contenido. Usualmente se usa como índice en un arreglo.
Top	Determina la posición del borde superior de la forma

CAJAS DE TEXTO

Propiedad	**Comentario**
Alignment	
FontName	
FontSize	
Height	
Left	
MaxLenght	Maneja rangos de 0 a 64 caracteres Determina la longitud máxima de caracteres a introducir
Multiline	Determina capacidad de multilinea
Name	Prefijo usado txt
PasswordChar	Especifica el carácter que será desplegado cuando el usuario introduzca caracteres
ScrollBars	Determina si se desplegarán barras de scroll, horizontales,verticales o ambas.
Text	Determina el texto que se desplegará
Top	
Width	

BOTONES DE COMANDO

Los botones de comando son aquellos en los cuales al momento de presionarlos realizan una acción, ya sea abrir una ventana más, ejecutar un comando inmediatamente o terminar una aplicación. Los botones de comando se encuentran con los títulos siguientes: Ok, Cancel, Aceptar, Exit, etc.

Algunas de las propiedades más importantes son las siguientes:

Propiedades	**Comentarios**
Cancel	Si se enciende esta propiedad, el evento click del botón se disparará cuando se presione la tecla ESC.
Default	Si se enciende este propiedad, el evento click del botón se disparará cuando el usuario presione la tecla ENTER.
Enabled	Propiedad que habilita o deshabilita el botón de comando.
FontBold	
FontItalic	
FontName	
FontSize	
Height	
Index	Propiedad que permite crear arreglos de botones de comando.
Left	
Name	Propiedad usada para nombrar al control en tiempo de corrida. El prefijo utilizado el txt.
TabIndex	Identifica el índice que se asignará al control al momento de moverse a través de los controles con la tecla tab.
Top	
Value	Indica si el botón ha sido seleccionado
Visible	
Width	

FRAMES O CUADROS

Los frames permiten agrupar información para que esta se presente con claridad para el usuario final. Existen también frames de tercera dimensión, los cuales tienen el mismo uso que los anteriores.

Algunas de la propiedades más importantes son:

Propiedades	Comentario
BackColor	
Caption	
Enabled	
FontBold	
FontItalic	
FontName	
FontSize	
ForeColor	Color para el texto que se escribirá
Height	
Left	
Name	Nombre utilizado en código. El prefijo el fra.
TabIndex	
Top	
Visible	
Width	

Dado que los frames permiten agrupar controles dentro de mi aplicación, es importante hacer notar que existen dos formas de dibujar controles dentro de un frame. La primera de ellas es por medio de un solo click en el control que se desea dibujar en el frame y después dibujar el control con el mouse dentro del frame, de esta forma el control dibujado dentro del frame se liga directamente con este, es decir, si muevo el control frame, también se moverán los controles que se dibujaron dentro de él. Por otro lado, si utilizo el método del doble click sobre el control que se quiere dibujar y después lo arrastro dentro de mi control frame, este control no estará ligado al frame.

BOTONES DE OPCIÓN (OPTION BUTTON)

Los botones de opción permiten elegir una opción de varias posibles. Es decir, estos botones son mútuamente excluyentes, de un grupo de botones .... solo puedo escoger uno y automáticamente los demás no quedan seleccionados.

Propiedades	Comentarios
Alignment	Puede ser: A la izquierda o derecha
BackColor	
Caption	
Enabled	
FontBold	
FontItalic	
FontName	
FontSize	
ForeColor	
Height	
Name	Nombre usado en código. El prefijo es opt.
TabIndex	
Top	
Visible	
Width	

BOTONES DE CHECK (CHECK BOX)

Estos botones permiten al usuario seleccionar de una serie de opciones varias, a diferencia de los botones de opción en donde solo es posible seleccionar una opción de varias. Tiene las mismas propiedades que los option button.

El prefijo usado en el nombre es chk.

CAJAS DE COMBO

Dentro de Visual Basic existen tres estilos de cajas combo, esto depende de las necesidades que se tengan. Para definir algún estilo especial de combo se define la propiedad Style.

Propiedades	Comentarios
BackColor	
Enabled	
FontBold	
FontItalic	
FontName	
FontSize	
Height	
List	Lista un elemento en particular. Propiedad accesible en tiempo de corrida
ListCount	Regresa el número de elementos en un combo. Propiedad accesible en tiempo de corrida
ListIndex	Regresa el índice del elemento seleccionado de la lista. Propiedad accesible en tiempo de corrida
Name	Nombre utilizado en código. Prefijo cmb

Los controles combo funcionan igual que una lista, por lo tanto tienen los mismos métodos, esto son:

AddItem	Agrega un elemento a la lista del combo
Cmb1.AddItem "Juan Pérez"	
RemoveItem (índice)	Elimina el elemento seleccionado por medio del índice.
Cmb1.RemoveItem(1)	Elimina el elemento uno de la lista del combo
Clear	Elimina todos los elementos de la lista del combo.

BARRAS DE SCROLL

Las barras de scroll tanto horizontales como verticales, permiten la navegación entre formas y listas muy grandes y también indican posición dentro de un rango.

Propiedades	Comentarios
Enabled	
Height	
LargeChange	Determina la cantidad de cambio en la barra de scroll cuando el usuario da un click dentro de la barra de scroll
Left	
Max	Determina la máxima posición de la barra de scroll
Min	Determina la mínima posición de la barra de scroll

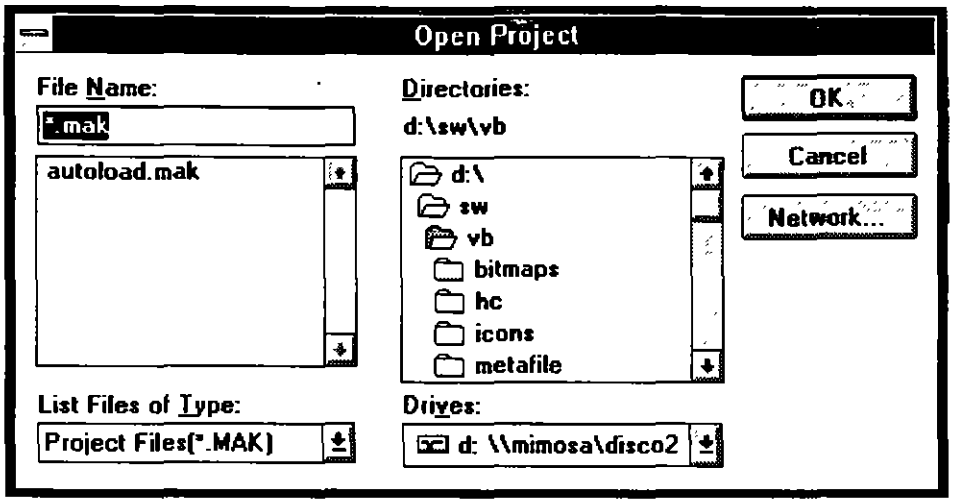
TIMER

Permite correr código en un intervalo específico de tiempo.

Propiedades	Comentarios
Enabled	
Interval	Determina el número de milisegundos entre llamadas a el evento Timer del control
Left	
Name	Nombre usado en código. Prefijo tmr

VII. OTROS CONTROLES Y EL FILE BROWSER

Un claro ejemplo de un file browser o buscador de archivos es la caja de diálogo común que se despliega al momento de abrir un proyecto en Visual Basic.



Si observamos esta caja de diálogo podemos ver que los diferentes controles que la forman están sincronizados, es decir, al cambiar de drive, automáticamente se cambia la lista de directorios y la lista de archivos, lo mismo sucede si cambiamos de directorio, la lista de archivos se actualiza, además esta lista contiene solo los archivos con la extensión marcada en el combo List File of Type.

Para implementar un file browser en Visual Basic, nos auxiliamos de los siguientes controles:

DRIVE LIST BOX

Propiedad	Comentario
Name	Nombre usado en código
Drive	Propiedad accesible sólo en tiempo de corrida, regresa el drive seleccionado. Esta es la propiedad de default de este control.

DIRECTORY LIST BOX

Propiedad	Comentario
Name	Nombre usado en código. Prefijo usado dir.
Path	Propiedad accesible solo en tiempo de corrida, este propiedad regresa el path o ruta absoluta para el directorio seleccionado.

FILE LIST BOX

Propiedad	Comentario
Archive	Despliega todos los archivos con el bit de archivo encendido.
Name	Prefijo usado fil.
FileName	Regresa o determina el archivo seleccionado de la lista de archivos. Accesible solo en tiempo de corrida.
Hidden	Despliega o no archivos escondidos
List	Regresa los elementos contenidos en la lista de archivos. Accesible solo en tiempo de corrida.
ListCount	Regresa el número total de elementos contenido en la lista de archivos. Accesible solo en tiempo de corrida.
ListIndex	Regresa el índice del elemento actualmente seleccionado. Accesible solo en tiempo de corrida.
Path	Accesible solo en tiempo de corrida.
Pattern	Archivos que se desean desplegar (*.*, *.bat, *.exe, etc.).

Sincronizando estos tres controles se logra construir un file browser. Para poder sincronizarlos se recurrirá a los eventos change de los controles, en donde se escribirá el código necesario para que cuando un control cambie, automáticamente cambien los demás.

```
Sub Drive1_Change( )
'Cuando el drive cambia, la propiedad path del directory list box también
cambia
Dir1.Path=Drive1.Path
End Sub


Sub Dir1_Change( )
'Cuando el directorio cambia, la propiedad path del file list box también cambia
File1.Path=Dir1.Path
End Sub
```

Los eventos change indican que el contenido de un control ha cambiado. Los eventos change están asociados a una gran cantidad de controles, entre ellos: combos, barras de scroll, etiquetas, picture boxes, etc.

CAJA DE DIALOGO COMÚN

La caja de diálogo común es un control de la versión profesional de Visual Basic, el cual nos permite desplegar cajas de diálogo con ciertas características en especial. Este tipo de controles, no ahorra el crear nuevas formas y programarlas para que realicen una función en particular. El archivo asociado a este control es: CMDIALOG.VBX.

```
Sub cmdabrir_Click( )
    CMDialog1.Action=2
End Sub
```

A continuación se presenta una lista de los posible valores para la propiedad Action y su correspondiente caja de diálogo:

Action	Tipo de Caja de Diálogo
0	Ninguna acción
1	Caja de diálogo para abrir un archivo
2	Caja de diálogo para salvar un archivo
3	Caja de diálogo para colores
4	Caja de diálogo para seleccionar fonts
5	Caja de diálogo para impresoras
6	Invoca WINHELP.EXE

GRID

El grid permite desplegar valores o texto en una malla, es decir, se maneja la información que se desea desplegar en renglones y columnas en específico. Algunas de sus propiedades son:

Propiedad	Comentario
Cols	Número total de columnas a desplegar
Rows	Número total de renglones a desplegar
Col	Número de columna en específico a utilizar. Disponible solo en tiempo de corrida
Row	Número de renglón en específico a utilizar. Disponible solo en tiempo de corrida
Text	Regresa o determina el texto a colocar en una celda en específico
Clip	Regresa o determina el contenido de una región seleccionada del grid
ColAlignment	Determina o regresa la alineación de datos en una columna
ColWidth	Determina o regresa el ancho (en twips) de una columna en especifico.
GridLines	Determina si las línea entre celdas serán desplegadas
Picture	Determina o regresa un gráfico para la celda actual

El siguiente ejemplo coloca texto en un rango de celdas de dos por dos:

```
Sub cmdClipText_Click( )
        Grid1.SelStartCol=1
        Grid1.SelEndCol=2
        Grid1.SelStartRow=2
        Grid1.SelEndRow=3
        tb$=Chr$(9)
        cr$=Chr$(13)
        ds$="Juan" + tb$ + "123" + cr$ + "Pedro" + tb$ + "456"
        Grid1.Clip=d$
End Sub
```

PANEL DE TERCERA DIMENSIÓN

Este control puede ser usado como indicador de estado o de progreso, o para dar una calidad de tercera dimensión a los controles que se coloquen sobre el panel.

Algunas de las propiedades más importantes son:

Propiedad	Comentario
Alignment	
BevelInner	Determina el estilo de las orillas internas del panel
BevelOuter	Determina el estilo de las orillas externas del panel
BevelWidth	Determina el ancho tanto de las orillas externas como internas del panel
FloodColor	Determina o regresa el color usado para pintar el área interna del panel cuando es usado como indicador de estado
FloodPercent	Determina o regresa el porcentaje del área pintada dentro del panel cuando es usado como indicador de estado
FloodType	Determina como será el llenado del panel cuando es usado como indicador de estado

BOTONES DE GRUPO

Los botones de grupo funcionan como una combinación de botones de comando y botones de opción. Son parecidos a los botones de comando porque cuando el usuario da un click sobre ellos, estos realizan alguna acción, pero también colocando las propiedades adecuadamente, al momento de seleccionar un botón de grupo, automáticamente los demás se deseleccionan, y esta es una característica de los botones de opción.

Algunas de la propiedades más importantes son:

Propiedad	Comentario
GroupAllowAllUp	Determina si todos los botones en un grupo lógico pueden estar arriba (no presionados)
GroupNumber	Determina o regresa el número de grupo para un grupo de botones dado. Esta propiedad se usa para crear grupos lógicos
Outline	Determina o regresa si el botón tiene borde o no
PictureDisabled	Especifica el bitmap a desplegarse cuando este botón este deshabilitado
PictureDn	Especifica el bitmap a desplegarse cuando el botón sea presionado
PictureDnChange	Determina como será desplegado el bitmap que se seleccionó en la propiedad PictureUp, cuando el botón sea presionado
PictureUp	Determina el bitmap a desplegarse cuando el botón este arriba (no presionado)

Grupos lógicos

Si deseamos utilizar grupos lógicos, es necesario trabajar con la propiedad GroupNumber.

Cuando deseamos que un grupo de botones funcionen mútuamente excluyentes es necesario que la propiedad GroupNumber de cada uno de estos botones sea diferente a cero, es decir, todos los botones que formen el grupo deben de tener el mismo número de grupo, pero este debe de ser diferente de cero.

Si por el contrario, queremos que un grupo de botones no sean mútuamente excluyentes, entonces, la propiedad GroupNumber de cada uno de los botones que formen el grupo debe ser igual a cero.

NOTA: El GroupNumber cero indica que el botón o los botones no tienen grupo.

VIII. TIPOS DE DATOS EN VISUAL BASIC

Variables. Valores que pueden cambiar y que el programa manipula
Constantes. Valores que no pueden cambiar y que el programa manipula
Procedimientos. Actividades que el programa realiza
Sentencias. Subactividades dentro de un procedimiento
Alcance. Determina que parte de un programa puede accesar datos o un procedimiento en especifico
Módulo. Es un archivo que contiene código y datos y además no esta asociado a ninguna forma en particular.

Tipos de variables

Tipo	Descripción	Carácter de declaración	Rango
Integer	Entero de 2 bytes	%	-32,768 a 32767
Long	Entero de 4 bytes	&	-2,147,483,648 a 2,147,483,647
Single	Punto flotante de 4 bytes	!	-3,40E+38 a 3.40E+38 *
Double	Punto flotante de 8 bytes	#	-1.79D+308 a 1.79D+308
Currency	Número de 8 bytes con punto decimal fijo	@	-9.22E+14 a 9.22E+14
String	Cadena de caracteres	$	0 a 65,500 caracteres aproximadamente
Variant (Default)	Fecha/Hora, punto flotante, string	(ninguno)	Fechas: Enero 1, 0000 a Diciembre 31, 9999; valores numéricos igual a double; cadenas igual a string

Declaración de variables

Para declarar variables, se puede realizar de dos maneras, la primera es utilizando la palabra reservada DIM, o usando alguna de las siguientes palabras: GLOBAL o STATIC. El no declarar variables puede ser un riesgo en algunas situaciones, si tomamos en cuenta que el tipo de datos por default es **Variant**.

Existen dos maneras en las que se puede hacer una declaración explícita de variables: Usando la palabra **As** o usando el carácter de declaración. A continuación se muestran algunos ejemplos:

Usando As	Usando el carácter de declaración
Dim J as Integer	Dim J%
Dim cuenta as Double	Dim cuenta#
Dim nombre as String	Dim nombre$

Dim R, S, K$
En esta declaración se tomará R y S como tipo variant y K como tipo string.

Declaraciones explícitas

Dado que Visual Basic permite usar variables sin declararlas, es posible hacer que todos los proyectos requieran de una declaración explícita de variables. Para realizar esto es necesario colocar las palabras reservadas Option Explicit, en la sección General Declarations de cualquier forma o módulo. O accesando el menú Options y el submenu Evironment y encender la opción Requiere Variable Declaration.

Variables tipo string con longitud variable

Este tipo de variable es útil si se desea tener variables que puedan contener cadenas de diferentes longitudes. Para declarar una variable de este tipo es de la siguiente forma:

```
Dim Mensaje As String
```

Variables tipo string con longitud fija

Cuando de antemano, se sabe que las variables no pueden tener más de un cierto número de caracteres, utilizamos variables tipo string con longitud fija. Para realizar una declaración de este tipo sería de la siguiente manera:

```
Dim nombre As String * 50
```

en donde la variable nombre tiene una longitud de 50 caracteres.
Otra forma de realizar la misma declaración es a través de constantes:

```
Global Const LONGITUD = 50
Global nombre as string=LONGITUD
```

Reglas:

Los nombres de variables pueden tener una longitud de hasta 40 caracteres
Los nombres pueden incluir letras, números y subguiones
El primer carácter debe ser una letra
No se pueden usar palabras reservadas de Visual Basic

El tipo de dato variant

El tipo de dato variant es el de default, esto quiere decir, que si se declara una variables sin indicar su tipo por default asumirá el tipo de dato variant. Como variant puede almacenar valores de fecha, hora, numéricos y cadenas no es necesario hacer conversiones entre estos tipos de valores, Visual Basic automáticamente realiza las conversiones necesarias.

Reglas:

La función IsNumeric determina si la variables variant contiene un valor numérico.

Normalmente al concatenar dos cadenas se utiliza el signo más (+). Para evitar ambigüedades con el tipo de dato variant se recomienda usar el ampersand (&) para indicar concatenación.

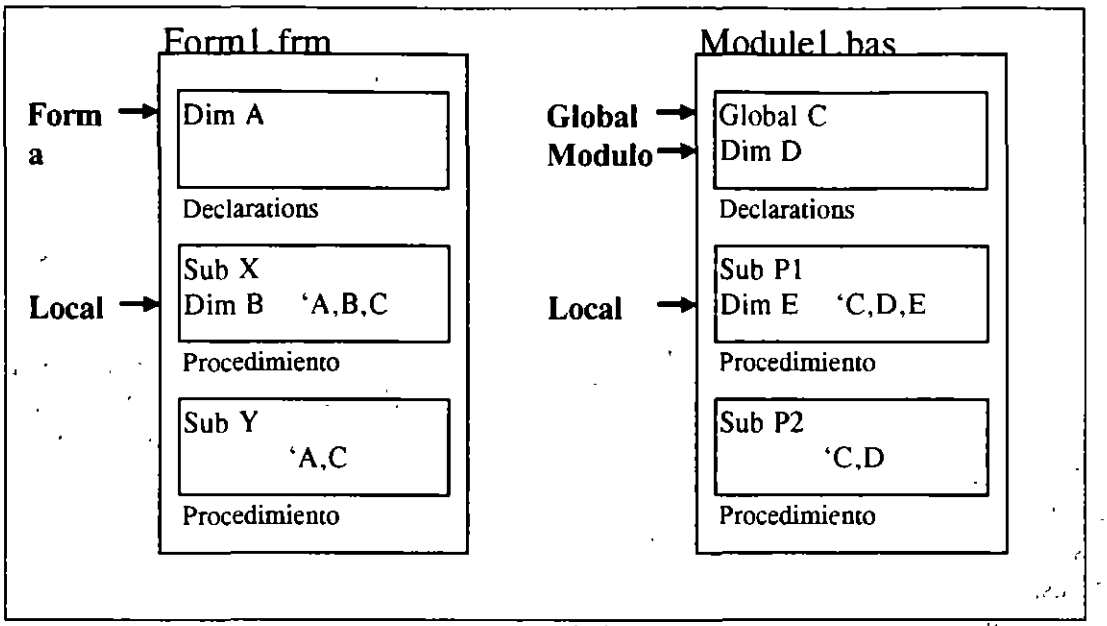
La función IsNull determina si una variable contiene un NULL.

Constantes

Las constantes son entidades dentro de un programa cuyo valor nunca cambia, es decir, es fijo. Por ejemplo, si su programa trabaja con figuras geométricas, probablemente necesite una constante como PI=3.1416.

Dentro del archivo CONSTANT.TXT (el cual se encuentra en la raíz del directorio \VB), se encuentran ejemplos típicos de declaración de constantes.

ALCANCE



El alcance de una variable determina el nivel en el cual puede ser vista, modificada o usada. Dentro de Visual Basic existen tres tipos de alcance: a nivel forma y módulo, a nivel local y a nivel global.

Variables a nivel forma

Con las variables declaradas a nivel forma, su alcance es en toda la forma, es decir, pueden ser vistas estas variables en cualquier procedimento dentro de la forma.

La declaración de estas variables se realiza en la sección de General Declarations de la forma utilizando la palabra Dim.

Variables a nivel módulo

El alcance de estas variables es solo dentro del módulo en el cual fueron declaradas, es decir, pueden ser vistas, modificadas o usadas dentro de todos los procedimientos que se encuentran definidos en el módulo.

La declaración de estas variables se realiza en la sección de General Declarations del módulo, usando la palabra reservada Dim.

Variables a nivel local

Como su nombre lo indica, son variables cuyo alcance es local, es decir, solo existen dentro del procedimiento en el cual fueron declaradas.

Estas variables generalmente se declaran dentro de algún procedimiento general (procedimientos que yo construí) o un procedimiento de eventos (procedimientos que ya nos da Visual Basic).

La declaración es utilizando la palabra reservada Dim.

Variables a nivel global

Si deseamos utilizar variables que puedan ser accesadas en cualquier lugar de mi aplicación (cualquier forma, procedimiento de eventos o procedimiento general de mi aplicación) es necesario hacer una declaración global.

Las variables globales se declaran en la sección General Declarations de un módulo y con las palabra reservada Global.

Para ilustrar cada uno de estos alcances se tomará como ejemplo el dibujo presentado al inicio de esta sección:

Dentro de este dibujo se tiene una forma y un módulo, cada uno con su correspondiente General Declarations (marcada como declarations) y también cada una con sus respectivos procedimientos de eventos para la forma y procedimientos generales para el módulo.

Veamos la forma:

Dentro de la forma en la sección General Declarations se tiene declarada una variable a nivel forma llamada A. Existe un subprocedimiento llamado X dentro del cual se tiene declarada una variable local llamada B; por lo tanto dentro de este procedimiento se pueden accesar las variables A por ser a nivel forma y B por ser a nivel local, pero también C por ser una variable global. En el subprocedimiento Y se pueden accesar las variables A por ser a nivel forma y C por ser una variable global.

Ahora analizando el módulo:

Existe una variable en General Declarations llamada C, pero que ha sido declarada usando la palabra reservada Global, es decir, esta variable es global y por lo tanto puede ser accesada desde cualquier lugar de mi aplicación. También en esta misma sección existe una variables llamada D, la cual es una variables a nivel módulo, es decir, se alcance es solo dentro del módulo en la cual fue declarada. Existe también un subprocedimiento llamado P1 el cual contiene una declaración local de una variable llamada E, por lo tanto, en este procedimiento se pueden accesar las variables C por ser global, D por ser a nivel módulo y E por ser local. Y finalmente dentro del subprocedimiento llamado P2, se pueden accesar las variables C por ser global y D por ser una variable a nivel módulo.

Tipos de datos definidos por el usuario (estructuras)

Dentro de Visual Basic se pueden definir lo que en lenguaje C se conocen como estructuras. La sintaxis que se utiliza para declarar una estructura es la siguiente:

```
Type tipo_definido_por_usuario
        nombre_elemento As tipo_variable
        [nombre_elemento As tipo_variable]

End Type
```

El poder crear nuevas variables es útil para poder ajustarnos a las necesidades de la aplicación que se esta realizando. Por ejemplo, puede ser necesario definir un tipo de dato que contenga información del personal de una compañía, esto se haría como sigue:

```
Type personal
        numero As Long
        nombre As String * 30
        direccion As String * 40
End Type
```

Una vez declarada la estructura para poder accesarla será necesario declarar una variable del nuevo tipo creado y para accesar a cada uno de los elementos que la forman será utilizando la sintaxis nombre_variable.nombre_elemento. A continuación se muestra un ejemplo:

```
Dim nuevo as personal
nuevo.numero = 5
nuevo.nombre = "Miguel"
nuevo.direccion = txtdireccion.text
```

Arreglos

Visual Basic como muchos otros lenguajes de programación permiten crear arreglos. Un arreglo es un grupo de variables del mismo tipo que comparten un nombre en común, y cada elemento del arreglo es identificado por medio de un índice.

La sintaxis para realizar la declaración de un arreglo es la siguiente:

```
Dim nombre_arreglo(numero_elementos) As tipo_dato
```

Por ejemplo:

```
Dim puntos(30) As Single
```

En este ejemplo se ha definido un arreglo llamado puntos con 31 elementos, por que por default el primer elemento del arreglo tiene el índice cero.

Otra forma de hacer la declaración de arreglos es indicando los límites que se desea que tenga el arreglo, esto se realiza de la siguiente forma:

```
Dim puntos(1 to 30) As Single
```

En este ejemplo se declara un arreglo llamado puntos con 30 elementos, y el primer elemento es referenciado con el índice 1.

Alcance

Dependiendo del lugar en donde se realiza la declaración del arreglo, es la sintaxis que se usa, a continuación se muestra un cuadro en donde se ilustran las declaraciones:

Alcance de la declaración	**Forma de declarar**
Alcance a nivel aplicación	Usar Global en General Declarations de un módulo .BAS
Alcance a nivel forma o módulo	Usar Dim en General Declaration de la forma o módulo
Alcance local	Usar Static

Arreglos con múltiples dimensiones

Los arreglos mostrados hasta el momento son solo de una dimensión, Visual Basic permite declarar arreglos de hasta 60 dimensiones. La sintaxis para la declaración es muy parecida:

```
Static calificacion (4,23) As Single
        o
Static calificacion(1 To 5, 1 To 24) As Single
```

Es un arreglo de 5 renglones de 24 casillas cada uno, es decir, se pueden almacenar cinco calificaciones por cada uno de los 24 estudiantes.

Arreglos dinámicos

Cuando es necesario cambiar en tiempo de corrida el tamaño de un arreglo, es decir, no conocemos a ciencia cierta su tamaño, es necesario implementar lo que se conocen como arreglos dinámicos, para realizar esto, es necesario hacer la declaración

del arreglo sin dimensiones y cuando ya conozcamos su tamaño redimensionarlo. Por ejemplo:

```
' Dim ArregloDin ( ) as String * 30
o
Global ArregloDin ( ) as String * 30          'En un módulo .BAS
```

y redimensionando después:

```
ReDim ArregloDin ( Val(TxtElementos.Text) )
```

NOTA: Cada vez que se utiliza la palabra ReDim, los valores que ya se tenían almacenados se pierden. Si se desea preservar el contenido de los arreglos cada vez que se redimensionen, es necesario usar en lugar de ReDim las palabras ReDim Preserve:

```
ReDim Preserve ArregloDin ( 10 )
```