UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

"LABORATORIO DE INFORMATICA II;
ELABORACION DE PRACTICAS"

T E S I S

QUE PARA OBTENER EL TITULO DE:

A C T U A R I O

P R E S E N T A :

JOSE LUIS CARRASCO HERNANDEZ

DIRECTOR DE TESIS:

M. en C. PAULO MAXIMO GUTIERREZ GONZALEZ

2005

ACT. MAURICIO AGUILAR GONZÁLEZ
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

"Laboratorio de Informática II; Elaboración de prácticas"

realizado por José Luis Carrasco Hernández

con número de cuenta 08210727-3 , quien cubrió los créditos de la carrera de: Actuaría

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis Propietario	M. en C. Paulo Máximo Gutiérrez González
Propietario	Dr. Alejandro Ricardo Garciadiego Dantan
Propietario	Act. Yolanda Silvia Calixto García
Suplente	M. en C. Juan Rico Arvizu
Suplente	Mat. Esteban Rubén Hurtado Cruz

Consejo Departamental de Matemáticas

Act. Jaime Vazquez Alamilla

FACULTAD DE CIENCIAS
CONSEJO DEPARTAMENTAL
DE
MATEMÁTICAS

INDICE

Dedicatoria

A mis padres, ***Arnulfo Hipólito Carrasco Vega*** e ***Isabel Hernández González***.

A quienes agradezco por su apoyo moral y económico que me brindaron durante toda mi formación educativa, además de los valores y la confianza que depositaron en mí.
Por eso, para mi es grato dedicarles esta Tesis como el fruto de la semilla que sembraron, la cual constituye para mí la herencia más valiosa que cualquier hijo desearía.

A mis hermanos, ***Hugo***, ***Jaime*** (finado) por ser para mi un ejemplo a seguir, ***Miguel Angel*** por su apoyo económico, ***Erendira*** por sus consejos, ***Rubén Raúl***, ***Armando***, ***Humberto Saúl***, ***Blanca Estela***, ***Gustavo***, ***Xochitl Donaji*** y ***Rodolfo***.

A quienes agradezco por que de alguna manera fueron parte fundamental para la realización de mi sueño que hoy se convierte en realidad.

A mis amigos, ***Fernando Muñoz Gómez*** y ***Ricardo Zaldivar Alcántara***.

A quienes agradezco por acompañarme en los momentos más difíciles y motivarme a seguir adelante.

Agradecimientos

En este trabajo como en cualquier otro, es siempre importante contar con la asesoria de personas cuyo conocimiento y experiencia en el desarrollo de trabajos de investigación ha sido parte fundamental de su reconocimiento y éxito, por lo que para mí es un honor en extender la presente a las siguientes personas:

A mi director de Tesis, ***M. en C. Paulo Máximo Gutiérrez González***, por el asesoramiento técnico y metodológico, que me permitió expresar las ideas de una manera mas clara acerca de los objetivos y alcances de la tesis.

A mis sinodales, ***Dr. Alejandro Garciadiego Dantan, Act. Yolanda Silvia Calixto García, M. en C. Juan Rico Arvizu y Mat. Esteban Rubén Hurtado Cruz***, por el tiempo dedicado a la revisión de la tesis y las acertadas e invaluables observaciones realizadas.

Introducción.

Ante la necesidad de impulsar el desarrollo científico y tecnológico a nivel bachillerato, algunas instituciones educativas han comenzado a realizar pruebas pilotos con algunos grupos, modificando el plan de estudios tradicional. Durante este proceso, una de las finalidades es impulsar el conocimiento computacional mediante la enseñanza de un lenguaje de programación. Si bien es cierto que en el cuarto Semestre dónde se pretende implementar este material, los alumnos no tienen todavía las bases para el desarrollo de funciones, es tarea del docente en comenzar a fomentar la creación de estas. Cabe mencionar que el material puede ser utilizado en cualquier institución educativa a nivel bachillerato, dónde su entendimiento óptimo sería implantarlo en el último semestre de su formación académica e inclusive ser usado en estudios superiores dónde se requiera la enseñanza de Visual Basic. Se supone que durante el semestre anterior de bachillerato, los alumnos adquirieron el conocimiento lógico para poder resolver problemas por medio de los algoritmos y su representación gráfica a través de diagramas de flujo básicos, ahora es el momento de implementar esos conocimientos mediante el uso de código en una aplicación real.

El material que se presenta fue desarrollado utilizando el sistema operativo Windows XP edición Profesional y el lenguaje de programación Visual Basic 6.0 edición Empresarial en español. Como es sabido, Visual Basic es conocido como un lenguaje de desarrollo de aplicaciones rápidas en un ambiente visual parecido al sistema operativo Windows, de hecho las aplicaciones que aquí se desarrollan, tienen la facilidad de ocupar Recursos y

Servicios del sistema operativo Windows, además de facilitarnos la instalación de nuestras aplicaciones.

Pero ¿Por qué Visual Basic? En el ámbito laboral, las Empresas demandan cada vez más el uso de este lenguaje para la creación de sus aplicaciones ya sea a nivel Empresarial o Internet. En la parte académica, es importante que los alumnos comiencen a estar familiarizados con la creación de proyectos para el desarrollo de las aplicaciones y la manera de poner disponibles estas aplicaciones para su uso.

Visual Basic es un lenguaje orientado a eventos, dónde un evento es la interacción de un usuario o el sistema con la aplicación. En este sentido, se cuentan con controles activeX para crear el diseño de la aplicación y estos controles responden a eventos predefinidos en ellos, con esto produciéndose procedimientos de eventos en los cuales el desarrollador tiene la facilidad de escribir código de lo que desea que ocurra en este evento. Además de que el desarrollador tiene la facilidad de crear sus propios procedimientos o funciones ya sea dentro de un mismo formulario, o crear su propia biblioteca para su uso en diferentes proyectos.

El material está diseñado de tal manera que el alumno irá desarrollando una práctica por semana durante el semestre. La semana está constituida por una hora en salón y dos horas en la sala de Informática. Otro punto importante que hay que mencionar, es que en varias ocasiones algunos alumnos llegan a faltar a la clase en la sala de Informática y por lo tanto no se realiza esa práctica, una solución a este tipo de situaciones es que el alumno va a poder continuar en la realización de sus prácticas ya que se le proporcionará la solución de

la práctica anterior para así continuar con su proyecto. Al final de este material se anexa un CD con las prácticas, soluciones y archivos necesarios para la realización de estas.

La aplicación que se va a desarrollar durante estas semanas, es una aplicación financiera llamada Préstamo en la cuál se estima el pago mensual de éste, y la cantidad total que se ha de pagar por el mismo al final de un periodo estipulado y a una tasa de interés dada, además de que se establece cierta seguridad para el uso de la aplicación. El proyecto durante su desarrollo sufre ciertos cambios de acuerdo al tema del que se este tratando en relación al lenguaje de programación. Además cada práctica tiene una introducción breve acerca de lo que se pretende con ésta, objetivos, ejercicios y pruebas a desarrollar.

El docente que pretenda implementar este material, debe tener al menos un conocimiento básico de un lenguaje de programación o haber tenido experiencia en el desarrollo de aplicaciones y elaborar las prácticas por su propia cuenta. Es importante mencionar que la teoría que se debe manejar, debe contemplar solo la parte básica de Visual Basic, puesto que en este momento sólo se pretende enseñar los fundamentos de este lenguaje.

Dentro de los requerimientos mínimos de Hardware y Software necesarios para el desarrollo de las prácticas se listan los siguientes:

Hardware:

Equipo de cómputo con un procesador mínimo Pentium, memoria 32MB deseable 64MB o más, espacio disponible en disco duro de 80MB, unidad de CD Rom, monitor VGA.

Software:

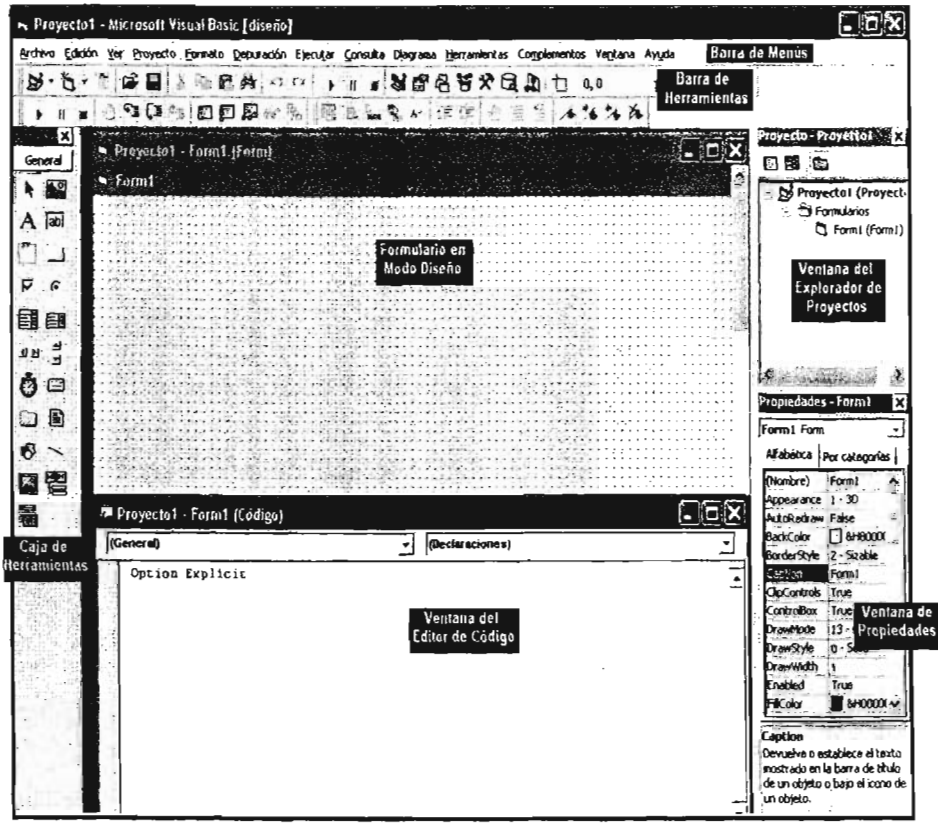
Sistema operativo Windows 98 o Superior, Microsoft Visual Studio 6.0 Versión Empresarial en español o inglés, opcional la ayuda MSDN.

Instalación:

Durante la instalación del sistema operativo, no necesariamente se necesita una instalación en red de los equipos de cómputo. Durante la instalación de Microsoft Studio Visual Studio 6.0 sólo se necesita instalar el disco uno e instalar sólo Microsoft Visual Basic y no necesariamente la ayuda MSDN ya que ocupa cerca de 1GB de espacio en disco duro.

El encargado de la sala de informática debe crear una carpeta para cada alumno en la cuál se copiarán los archivos o carpetas necesarias para el desarrollo del proyecto, e informar al docente acerca de la ubicación de éstas. Es importante mencionar que sólo se deben copiar las carpetas o archivos correspondientes a la práctica que se vaya a realizar.

En la siguiente ilustración se muestra el Ambiente de Desarrollo de Visual Basic.



PRÁCTICA 1: CREACIÓN DE UNA APLICACIÓN SIMPLE

En esta práctica, aprenderás como crear una aplicación simple en Visual Basic.

Objetivos:

Identificar los elementos que constituye el ambiente de desarrollo Visual Basic.
- ❖ Agregar controles al formulario.
- ❖ Usar la ayuda de Visual Basic.
- ❖ Identificar las propiedades de los controles.
- ❖ Crear el archivo ejecutable " .exe " de una aplicación.

Ejercicios

Ejercicio 1: Uso de controles
En este ejercicio, practicarás colocando controles sobre un formulario, para familiarizarte con el ambiente de desarrollo y los controles.

Ejercicio 2: Creación de una aplicación con código
En este ejercicio crearás una aplicación que cambia las propiedades de controles mientras que la aplicación esta ejecutándose.

Ejercicio 3: Usando código desde la ayuda
En este ejercicio, localizarás un ejemplo de código y copiarás éste en la aplicación.

Ejercicio 1: Uso de controles

En este ejercicio, practicarás colocando controles sobre un formulario, para familiarizarte con el ambiente de desarrollo y los controles.

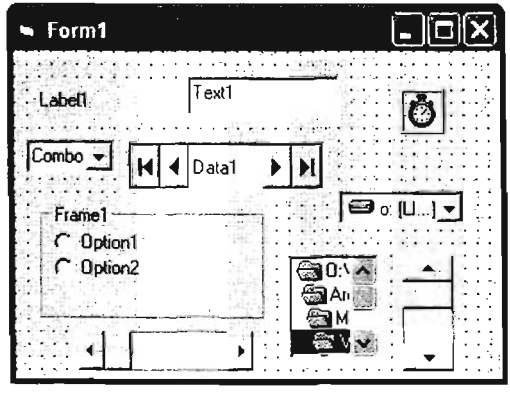
I.- Inicia Visual Basic

1.- Dar clic en el botón **Inicio**, ir a **Todos los programas**, después colócate en **Microsoft Visual Studio 6.0**, y entonces dar clic en **Microsoft Visual Basic 6.0**.
2.- En la ventana nuevo proyecto, dar clic en la plantilla **EXE estándar** y entonces dar clic en el botón **Abrir**, para iniciar tu nuevo proyecto.

II.- Añadir controles a un formulario

1.- Dar clic a un control de la **caja de herramientas**
2.- Colocar el cursor del ratón sobre el formulario en la posición deseada (el cuál será la esquina superior izquierda del control).
3.- Dar clic al botón izquierdo del ratón y manteniéndolo apretado, arrastra el cursor del ratón a la posición deseada (el cuál será la esquina inferior derecha) y entonces suelta el botón izquierdo del ratón.
4.- Repite los 3 pasos anteriores para diferentes controles de la **caja de herramientas.**

La siguiente ilustración muestra un ejemplo de cómo podrían quedar colocados los controles sobre el formulario.



III.- Obtener ayuda sobre un control

1.- Dar clic a un control que hayas colocado sobre el formulario.
2.- Presiona la tecla F1.

Nota: La ayuda MSDN debe abrirse en el tópico del control seleccionado.

3.- Examina el contenido del tópico de ayuda de los controles y entonces cierra la ventana de Ayuda.

IV.- Ejecuta la aplicación

1.- Sobre el menú **Ejecutar**, dar clic sobre **Iniciar** o dar clic sobre el botón **Iniciar** que se encuentra en la **barra de herramientas** de Visual Basic.
2.- Prueba los controles sobre el formulario.

Nota: Los controles trabajan como tú esperarías, aún sin haber escrito código.

3.- Cierra la aplicación que se está ejecutando y regresa a modo Diseño, usando alguno de estos métodos:
i.- Sobre la **barra de herramientas**, dar clic en el botón **Terminar**.
ii.- Sobre el menú **Ejecutar**, dar clic en **Terminar**.
iii.- Sobre la barra de títulos del formulario, dar clic en **Cerrar**.

V.- Borrar controles del Formulario.

1.- Coloca el cursor del ratón en la esquina superior izquierda del formulario.
2.- Dar clic y manteniendo apretado el botón izquierdo del ratón, arrastra el cursor hacia la esquina inferior derecha del formulario, entonces suelta el botón del ratón.

Nota: Todos los botones aparecerán seleccionados.

3.- Sobre el menú **Edición**, selecciona la opción **Eliminar**, ó presiona la tecla Supr.

Nota: Todos los controles seleccionados serán borrados del formulario.

Ejercicio 2: Creación de una aplicación con código

En este ejercicio, usarás el mismo proyecto usado en el ejercicio 1 y crearás una aplicación que cambia las propiedades de los controles mientras que la aplicación se está ejecutando.

I.- Coloca los controles TextBox y Label al formulario y alinéalos horizontalmente

1.- Coloca un control **TextBox** en el formulario.
2.- Coloca un control **Label** en el formulario, y posiciónalo al lado derecho del **TextBox**.
3.- Dar clic sobre el formulario y selecciona ambos controles.
4.- Dar clic sobre el control que se encuentra en el formulario, el cual quieres que el otro control se alinee.

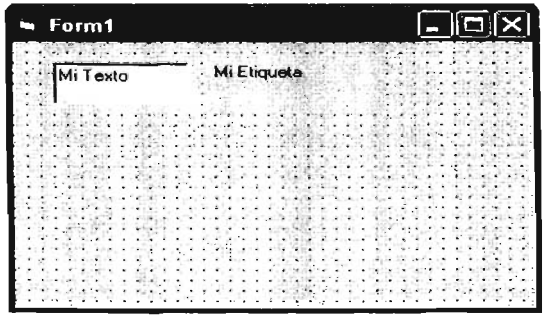
Nota: El control seleccionado debe cambiar a un color sólido.

5.- Sobre el menú **Formato**, seleccionar **Alinear**, y después seleccionar **Superior**, para que los controles se alineen al último control seleccionado.

II.- Establece las propiedades de los controles en el modo diseño.

1.- Dar clic sobre el formulario, para deseleccionar los controles, y entonces selecciona el control **Label** sobre el formulario.
2.- En la ventana de propiedades, cambia la propiedad **Caption** a "Mi etiqueta".
3.- Selecciona el control **TextBox**.
4.- En la ventana de propiedades, cambia la propiedad **Text** a "Mi Texto".

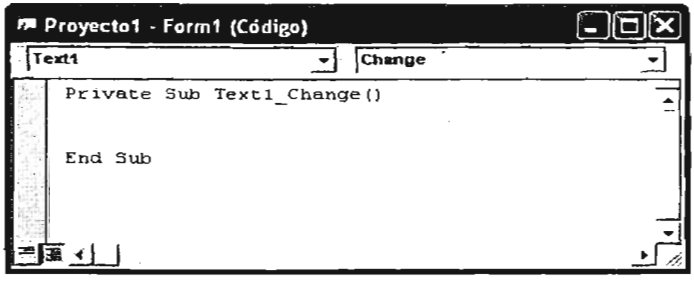
En la siguiente imagen se muestra como debe quedar el formulario.



III.- Cambia las propiedades al modo de ejecución.

1.- Dar doble clic al control **TextBox** sobre el formulario.

Nota: La ventana del editor de código se abrirá, mostrando el procedimiento del evento **Text1_Change**.

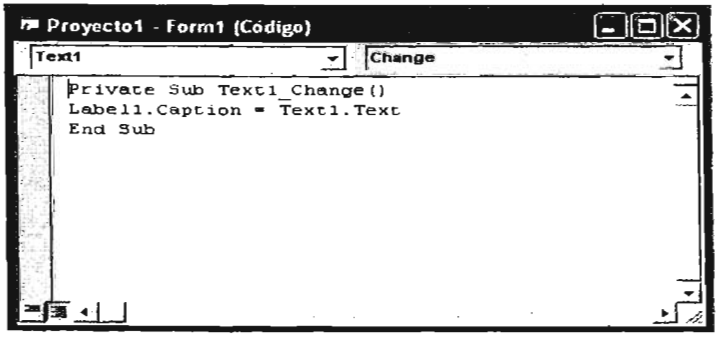


2.- Agrega la siguiente declaración dentro del procedimiento.

Label1.Caption = Text1.Text

Nota: Esta declaración cambia la propiedad **Caption** del control **Label** al Texto que el usuario teclee en el control **TextBox**.

La siguiente imagen muestra como debe quedar el procedimiento.



3.- Ejecuta la aplicación y teclea cualquier texto en el control **TextBox**. Observa que pasa.
4.- Cierra la aplicación y regresa a modo diseño.

IV.- Guarda el proyecto

1.- Sobre el menú **Archivo**, selecciona **Guardar Proyecto**.
2.- Cuando aparezca la ventana **Guardar proyecto como**, cambia la carpeta destino a (Carpeta de Instalación)\Prácticas\Practica01 y deja el nombre de default para el formulario a Form1 y entonces da clic en **Guardar**.
3.- Cuando se te solicite guardar el proyecto, cambia la carpeta destino a (Carpeta de Instalación)\Prácticas\Practica01 y deja el nombre de default para el Proyecto a Proyecto1 y entonces da clic en **Guardar**.

V.- Crea un archivo ejecutable ".exe"

1.- Sobre el menú **Archivo**, seleccionar y dar clic en **Generar proyecto1.exe...**, para crear el archivo ejecutable. Asegura que la carpeta destino sea (Carpeta de Instalación) \Prácticas\Practica01.
2.- En el **Nombre del Archivo**, cambia éste a **PrimerAplicación** y entonces da clic en **Aceptar**.
3.- Cierra Visual Basic y guarda el proyecto.

VI.- Ejecuta el archivo ejecutable ".exe"

1.- Dar clic en el botón **Inicio**, seleccionar **Todos los programas,** posicionarse en **Accesorios** y después dar clic en **Explorador de Windows**.

Nota: Esta ruta puede variar dependiendo del Sistema Operativo que se este utilizando.

2.- En el **Explorador de Windows**, navega a la carpeta (Carpeta de Instalación)\Prácticas \Practica01.
3.- Ejecuta la aplicación que creaste apretando doble clic sobre el archivo ejecutable.
4.- Prueba la aplicación colocando algún texto en el control **TextBox**.
5.- Cierra la aplicación.

Ejercicio 3 Usando código desde la ayuda

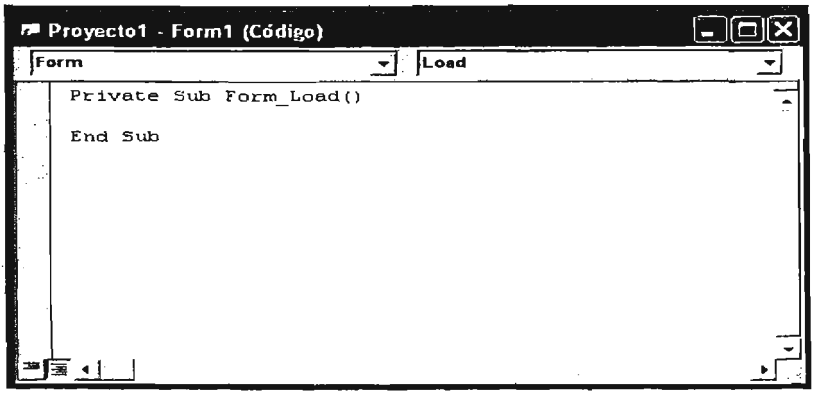
En este ejercicio localizarás un ejemplo de código en la ayuda de MSDN y copiarás este dentro de una aplicación.

I.- Abrir el proyecto1

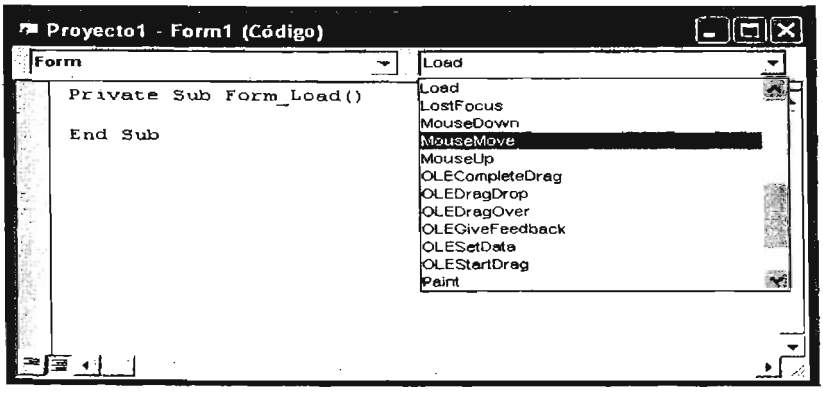
1.- Inicia Visual Basic
2.- Dar clic en la etiqueta **Recientes**, selecciona Proyecto1 y dar clic en **Abrir**.
3.- Si el formulario no es visible, selecciona éste de la ventana del Explorador de Proyectos y dar clic en el icono **Ver objeto**.

II.- Uso de la Ayuda

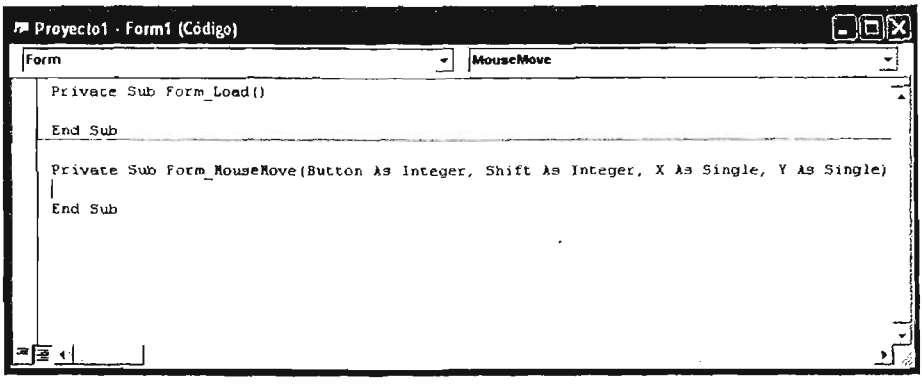
1.- Dar doble clic en cualquier parte del formulario, fuera de los controles **TextBox** y **Label** El siguiente procedimiento aparecerá en la ventana del editor de código:



2.- En la ventana del editor de código, de la lista desplegable de **Procedimiento**, selecciona **MouseMove.**



El contenido de la ventana del editor de código desplegará lo siguiente:



3.- Escribe **FillColor** entre *Private Sub Form_MouseMove* y *End Sub*.
4.- Selecciona la palabra **FillColor**, y presiona la tecla F1.

Nota: La Ayuda MSDN desplegará el Tópico de Ayuda FillColor Property.

III.- Copia el código de ejemplo de la Ayuda MSDN y copia ésta dentro de la aplicación Visual Basic

1.- Dar clic al Tópico **Ejemplo** en la parte superior de la ventana de Ayuda.

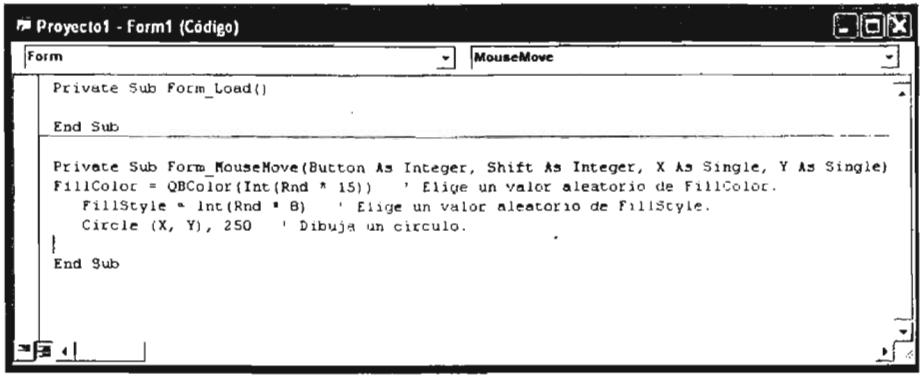
Nota: Una ventana de ayuda debe desplegarse con un ejemplo de cómo usar la propiedad **FillColor.**

2.- Selecciona las tres líneas de código entre las declaraciones **Sub** y **End Sub**.
3.- Despliega el menú contextual (dando clic con el botón derecho del ratón) y selecciona **Copiar**.
4.- Cierra la ventana de ayuda MSDN.

Nota: Ahora debes regresar a la ventana del editor de código, desplegando el procedimiento del evento **Form_MouseMove**. Si no, en la ventana del editor de código, dar clic en la lista desplegable de **Objeto** y dar clic en **Form**, y en la lista desplegable de **Procedimiento** selecciona **MouseMove** y dar clic, después desplázate al procedimiento de evento **Form_MouseMove**.

5.- Selecciona el comando **FillColor** en el procedimiento.
6.- Sobre el menú **Edición**, seleccionar **Pegar**, para pegar el código copiado dentro del procedimiento del evento **Form_MouseMove**.

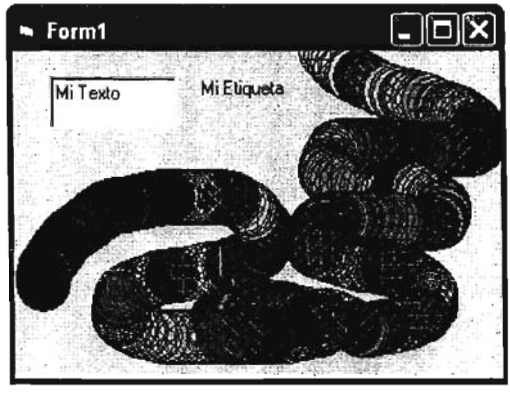
El procedimiento del evento resultante debe parecerse al siguiente:



IV.- Guarda y Prueba tu aplicación.

1.- Sobre el menú **Archivo**, dar clic en **Guardar proyecto**, para guardar los cambios al proyecto y el formulario en (Carpeta de Instalación)\Prácticas\Practica01.
2.- Ejecuta la aplicación y mueve el cursor del ratón alrededor del formulario. ¿Qué ocurrió? Observarás que conforme desplazas el cursor sobre el formulario, este crea un círculo con un estilo de relleno y el relleno de color con valores tomados aleatoriamente.
3.- Cierra la aplicación.

En la siguiente imagen, se muestra un ejemplo del desplazamiento del cursor sobre el formulario.



PRÁCTICA 2: CREAR UNA APLICACIÓN VISUAL BASIC

En las prácticas 2, 3, 4, 5, 6, 8, 9, 10, y 11, crearás una aplicación que estima el pago de un préstamo. Crearás la aplicación completa en etapas. Cada etapa será construida sobre el código creado en la práctica anterior. Al inicio de cada práctica, puedes continuar con tus propios archivos ó iniciar con los archivos que se te proporcionan.

En esta práctica, iniciarás la aplicación que estima el pago de un préstamo, creando una pantalla de autorización de uso de la aplicación.

Objetivos

Después de terminar esta práctica serás capaz de:

- Crear una aplicación simple en Visual Basic.
- Crear un procedimiento de evento.
- Recuperar las propiedades de los objetos al momento de ejecutar.

Ejercicios

Ejercicio 1: Crear una pantalla de Acceso
En este ejercicio, crearás una pantalla de acceso para la aplicación que estima el pago de un préstamo. Iniciarás un nuevo proyecto, renombraras el formulario, colocarás controles sobre el formulario y establecerás propiedades a los controles.

Ejercicio 2: Añadir código para habilitar y deshabilitar un control.
En este ejercicio, añadirás código que habilitará el botón **Aceptar** sólo cuándo el nombre del usuario y una contraseña hayan sido introducidas.

Ejercicio 3: Añadir un icono
En este ejercicio, añadirás a un formulario un Icono usando el control **Image**.

Ejercicio 1: Crear una pantalla de acceso

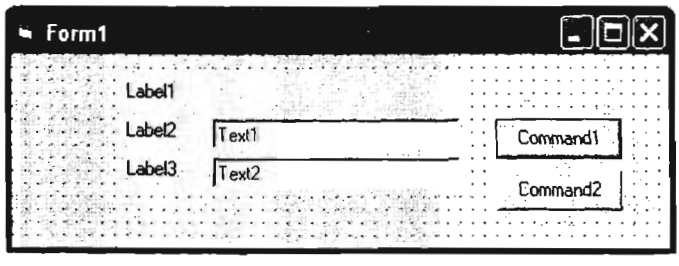
En este ejercicio, crearás una pantalla de acceso para la aplicación que estima el pago de un préstamo. Iniciarás un nuevo proyecto, renombraras el formulario, colocarás controles sobre el formulario y establecerás propiedades a los controles.

I.- Inicia un nuevo proyecto

1.- Inicia Visual Basic desde el menú **Inicio** (Si Visual Basic no está ejecutándose). De otra manera, sobre el menú **Archivo**, da clic en **Nuevo proyecto** para iniciar Visual Basic.
2.- Selecciona **EXE estándar** y da clic en **Aceptar**.

II.- Añade controles al formulario de default

1.- Añade tres controles **Label** al formulario.
2.- Añade dos controles **TextBox** al formulario.
3.- Añade dos controles **CommandButton** al formulario.
4.- Mueve y redimensiona los controles para que se parezcan a la siguiente ilustración:



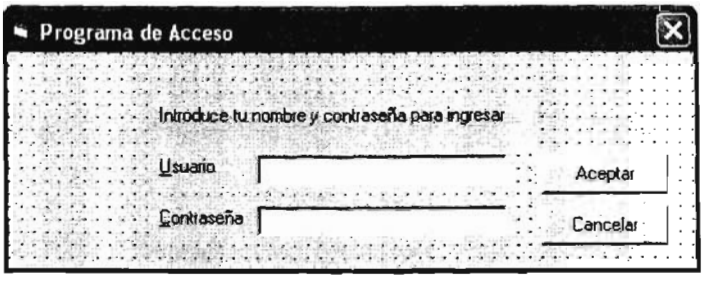
III.- Establece las propiedades para los controles

1.- Selecciona cualquier control sobre el formulario.
2.- En la ventana de **Propiedades**, selecciona la propiedad para los controles listados en la siguiente tabla.
3.- Coloca el nuevo valor para la propiedad, como es especificado en la tabla:

Nombre Actual	Propiedad	Nuevo Valor
Label1	**Nombre**	lblInstruccion
	Caption	Introduce tu Nombre y contraseña para ingresar.
Label2	**Nombre**	lblUsuario
	Caption	&Usuario
Label3	**Nombre**	lblContraseña
	Caption	&Contraseña
Text1	**Nombre**	txtUsuario
	Text	

Nombre Actual	Propiedad	Nuevo Valor
Text2	**Nombre** **Text** **PasswordChar**	txtContraseña *
Command1	**Nombre** **Caption** **Default**	cmdAceptar Aceptar True
Command2	**Nombre** **Caption** **Cancel**	cmdCancelar Cancelar True
Form1	**Nombre** **Caption** **BorderStyle**	frmAcceso Programa de Acceso 1 – Fixed Single

El formulario resultante debe parecerse al de la siguiente ilustración:



IV.- Añade código al evento click del control cmdAceptar.

1.- Dar doble clic al botón de comando **cmdAceptar**. Esto abre la ventana del editor de código, con el siguiente código ya insertado.

```
Private Sub cmdAceptar_Click

End Sub
```

2.- Añade código al procedimiento del evento **Click** para mostrar una ventana de mensaje con el valor actual de los TextBox **Usuario** y **Contraseña**.

```
MsgBox "Usuario = " & txtUsuario.Text & _
       ", Contraseña = " & txtContraseña.Text
```

Nota: El espacio y el guión bajo en el código de arriba, son usados para dividir una declaración sencilla en varias líneas. El **ampersand** es usado para forzar la combinación de dos expresiones.

3.- Ejecuta la aplicación. Introduce un usuario y una contraseña, y entonces da Click en **Aceptar**. ¿Qué ocurrió? Nos muestra el nombre del usuario y le contraseña en una ventana de mensaje.
4.- Introduce un diferente Usuario y contraseña y entonces presiona la tecla **ENTER**. ¿Qué ocurrió? ¿Por qué? Nos muestra el nombre del usuario y le contraseña en una ventana de mensaje, por que se estableció la propiedad ***default*** a ***true*** del control ***command1***, la cuál direcciona la acción a la tecla ***Enter***.
5.- Cierra la aplicación y regresa a modo diseño.

V.- Añade código al evento click del control cmdCancelar.

1.- Dar doble Clic al botón de comando **cmdCancelar**. Esto abre la ventana del editor de código, con el siguiente código ya insertado.

Private Sub cmdCancelar_Click

End Sub

2.- Añade código al procedimiento del evento **Click** para mostrar una ventana de mensaje.

MsgBox "Este es el botón Cancelar."

3.- Ejecuta la aplicación. Introduce un usuario y una contraseña, y entonces da clic en **Cancelar**. ¿Qué ocurrió? Nos muestra "Este es el botón cancelar." en una ventana de mensaje.
4.- Introduce un diferente Usuario y contraseña y entonces presiona la tecla **ESC**. ¿Qué ocurrió? ¿Por qué? Nos muestra "Este es el botón cancelar." en una ventana de mensaje, por que se estableció la propiedad ***cancel*** a ***true*** del control ***command2***, la cuál direcciona la acción a la tecla **Esc**.

5.-Cierra la aplicación y regresa a modo diseño.

VI.- Nombra el Proyecto

1.- Sobre el menú **Proyecto**, da clic en **Propiedades de proyecto1**
2.- En Nombre de proyecto, Teclea **ProyectoPrestamo**, y da clic en **Aceptar**.

VII.- Guarda el proyecto

1.- Sobre el menú Archivo, da clic en **Guardar proyecto**.
2.- Cuando aparezca la ventana **Guardar proyecto como**, cambia la carpeta destino a (Carpeta de Instalación)\Prácticas\Practica02 y cambia el nombre del formulario a **frmAcceso** y entonces da clic en **Guardar**.
3.- Cuando se te solicite guardar el proyecto, cambia la carpeta destino a (Carpeta de Instalación)\Prácticas\Practica02 y cambia el nombre del Proyecto a **Prestamo** y entonces da clic en **Guardar**.

Ejercicio 2: Añadir código para habilitar y deshabilitar un Botón

En este ejercicio, añadirás código que habilitará el botón **Aceptar** sólo cuándo el nombre del usuario y una contraseña hayan sido introducidas.

Cuándo creas una aplicación, debes minimizar la posibilidad de que los usuarios cometan errores. Una manera de hacer esto, es deshabilitar controles, cuando los seleccionan podrían causar un error. Por ejemplo, los usuarios no deberían ser capaces de dar clic al botón Aceptar sobre el formulario de autorización hasta que ellos introduzcan un nombre de Usuario y una Contraseña.

I.- Habilita el botón Aceptar

1.- En la ventana de propiedades, establece la propiedad **Enabled** del botón **cmdAceptar** a false.
2.- Dar doble clic al TextBox **txtUsuario**

Nota: La ventana del editor de código abrirá el procedimiento del evento txtUsuario_Change.

3.- Añade las siguientes líneas de código al evento **txtUsuario_Change**.

```
If txtUsuario.Text < > "" And txtContraseña.Text < > "" Then
        cmdAceptar.Enabled = True
Else
        cmdAceptar.Enabled = False
End If
```

Nota: No debe haber espacios entre las comillas. Colocando un espacio entre las comillas, la condición en la declaración If verificará el espacio en el **TextBox** y no una cadena de longitud cero.

4.- Copia el código del evento **txtUsuario_Change** al evento **Change** del TextBox **txtContraseña**.
5.- Guarda y prueba la aplicación. ¿Esta el botón **Aceptar** habilitado? **No**.
6.- Introduce un nombre de Usuario. ¿Esta el botón **Aceptar** habilitado? ¿Por qué? **No**, por que no se ha introducido un valor para la contraseña.
7.- Ahora introduce la Contraseña. ¿Esta el botón **Aceptar** habilitado? ¿Por qué? **Si**, por que se han introducido valores tanto para el usuario como para la contraseña

Ejercicio 3: Añadir un Icono

En este ejercicio añadirás un icono al formulario usando el control **Image**.

I.- Añadiendo un gráfico al formulario acceso.

1.- Agrega un control **Image** en la esquina superior izquierda del formulario.
2.- Cambia la propiedad **Name** del control **Image** a **imgLogo**.
3.- Cambia la propiedad **BorderStyle** del control **Image** a **1 – Fixed Single**.
4.- Cambia la propiedad **Picture** del control **Image** a un Icono dando clic en el botón (...). El cuadro de diálogo **Cargar Imagen** aparecerá.
5.- Selecciona el icono Pc03.ico de la siguiente ruta: c:\Archivos de Programa\Microsoft Visual Studio\Common\Graphics\Icons\Computer y da clic en **Abrir**.
6.- Usa la función **LoadPicture** para cambiar la propiedad **Picture** del control **Image**, en el evento **Click** del control **Image**.

imgLogo.Picture = LoadPicture("C:\Archivos de Programa\Microsoft Visual Studio\Common\Graphics\Icons\Elements\Earth.ico")

7.- Guarda y prueba la aplicación. ¿Qué imagen es desplegada cuando inicia la aplicación? La imagen que se despliega es una **computadora**.
8.- Da clic sobre la imagen. ¿Qué ocurrió? Ahora se despliega la imagen de la **tierra**.

PRÁCTICA 3: TRABAJAR CON FORMULARIOS

En esta práctica, agregarás funcionalidad al proyecto Prestamo iniciado en la Práctica 2. Agregarás un Shell al formulario de la aplicación principal. Desde el shell, cargarás el formulario Acceso y mostrarás el formulario principal de la aplicación. Continuarás trabajando con los archivos que creaste en la Práctica 2, ó puedes usar los archivos que se proporcionan en (Carpeta de Instalación)\Prácticas\Praetica03.

Objetivos:

Después de completar esta práctica serás capaz de:

- Agregar un nuevo formulario a un proyecto existente.
- Cambiar el formulario de inicio de un proyecto.
- Cargar y desplegar un formulario desde otro formulario.

Ejercicios

Los siguientes ejercicios te darán la práctica de usar métodos y eventos para cargar y descargar un formulario.

Ejercicio1: Agregar el Formulario Principal
En este ejercicio, agregaras un formulario, frmPrincipal al proyecto. Este formulario en esencia va a ser el formulario principal de la aplicación. De hecho el formulario frmPrincipal será usado para cargar el formulario frmAcceso.

Ejercicio2: Cerrar formularios y finalizar una aplicación
En este ejercicio, agregarás código para cerrar el formulario Acceso cuándo se le da clic al control Aceptar ó al control Cancelar, y finaliza la aplicación cuándo al botón Cerrar se le da un clic.

Ejercicio3: Agregar una ventana de Mensaje [Message Box]
En este ejercicio, agregarás código al formulario principal para verificar que el usuario quiere salir de la aplicación. Además establecerás una propiedad al formulario Acceso para centrar éste cuando sea mostrado.

Ejercicio 1: Agregar el Formulario Principal

En este ejercicio, agregaras un formulario, frmPrincipal al proyecto. Este formulario en esencia va a ser el formulario principal de la aplicación. De hecho el formulario frmPrincipal será usado para cargar el formulario frmAcceso.

I.- Abre el proyecto de la aplicación Prestamo

Abre cualquiera de los proyectos de la aplicación Prestamo, el proyecto con el que has estado trabajando, o el proyecto que se encuentra en (Carpeta de Instalación)\Prácticas \Practica03.

II.- Agrega un nuevo formulario al proyecto.

1.- Sobre el menú **Proyecto**, da clic en **Agregar formulario**
2.- En la ventana **Agregar Formulario**, selecciona **Formulario** y da clic en el botón **Abrir**

Nota: Un nuevo formulario debe ser agregado al proyecto y desplegado en el explorador de proyectos.

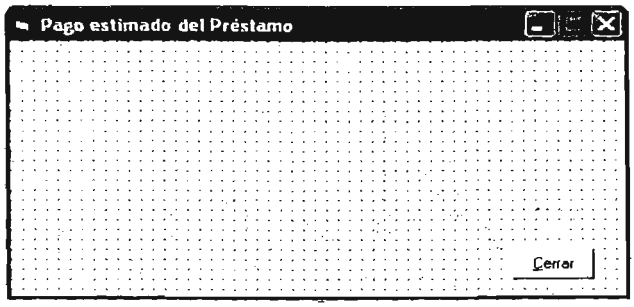
3.- Establece las siguientes propiedades al formulario.

Propiedad	Configuración
Nombre	frmPrincipal
Caption	Pago estimado del Préstamo
BorderStyle	1 – Fixed Single
MinButton	True

4.- Agrega un botón de Comando a frmPrincipal que cerrará la aplicación.
5.- Establece las siguientes propiedades al botón de comando.

Propiedad	Configuración
(Nombre)	cmdCerrar
Caption	&Cerrar
Cancel	True

Tu formulario debe ser parecido a la siguiente ilustración:



III.- Muestra el formulario Acceso

1.- Dar doble clic en cualquier parte del formulario frmPrincipal.

Nota: La ventana del editor de código aparece con el siguiente código:

Private Sub Form_Load()

End Sub

2.- Dentro de este procedimiento, agrega una línea de código para desplegar el formulario frmAcceso Modalmente.
Tu código debe ser el siguiente:

frmAcceso.Show vbModal

IV.- Cambia el formulario que inicia la aplicación

1.- Sobre el menú Proyecto, da clic en **Propiedades de ProyectoPrestamo**.
2.- En la ventana **ProyectoPrestamo**, localiza **Objeto inicial:**, y de la lista selecciona **frmPrincipal**, después da clic en **Aceptar**.

V.- Guarda y prueba la aplicación

1.- Guarda el proyecto.
2.- Ejecuta la aplicación. Introduce un nombre de usuario y contraseña y da clic en **Aceptar.** ¿Qué ocurrió? ¿Por qué el formulario de Acceso no desapareció?
3.- Cierra la aplicación y regresa a modo diseño.

Ejercicio 2: Cerrar formularios y finalizar una aplicación.

En este ejercicio, agregarás código para cerrar el formulario Acceso cuándo se le da clic al control Aceptar ó al control Cancelar, y finaliza la aplicación cuándo al botón Cerrar se le da un clic.

I.- Descargar el formulario Acceso cuando el usuario de clic en el botón Aceptar sobre frmAcceso.

1.- Agrega código al procedimiento del evento **Click** del botón **cmdAceptar** que se encuentra en **frmAcceso** para cerrar el formulario, esto es, después de la instrucción [MsgBox].

Unload frmAcceso

II.- Finalizar la aplicación cuando el usuario da clic sobre el botón Cancelar que se encuentra en frmAcceso, o sobre el botón Cerrar que se encuentra en frmPrincipal.

1.- Agrega código al procedimiento del evento **Click** del botón **cmdCancelar** que se encuentra en **frmAcceso** para finalizar la aplicación:

End

2.- Agrega código al procedimiento del evento **Click** del botón **cmdCerrar** que se encuentra en **frmPrincipal** para descargar el formulario:

Unload frmPrincipal

3.- Agrega código al procedimiento del evento **Unload** de **frmPrincipal** para finalizar la aplicación:

End

III.- Guarda y prueba la aplicación

1.- Guarda el proyecto en la carpeta (Carpeta de Instalación)\Prácticas\Practica03
2.- Ejecuta la aplicación. Introduce un nombre de usuario y contraseña. Da clic en Aceptar. ¿Qué ocurrió?
3.- Ahora da clic en Cerrar. ¿Qué Ocurrió?
4.- Cierra la aplicación y regresa a modo diseño.

Ejercicio 3: Agregar una ventana de Mensaje [Message Box]

En este ejercicio, agregarás código al formulario principal para verificar que el usuario quiere salir de la aplicación. Además establecerás una propiedad al formulario Acceso para centrar éste cuando sea mostrado.

En vez de sólo finalizar la aplicación en el evento **Unload** del formulario principal, puedes habilitar al usuario para seleccionar entre cerrar la aplicación o continuar usando ésta.

I.- Pregunta al usuario antes de salir de la aplicación

1.- Edita el código del procedimiento del evento **Form_Unload** de **frmPrincipal**, y agrega el siguiente código:

```
Dim iRespuesta as Integer

iRespuesta = MsgBox ("¿Estás seguro de que quieres salir de la aplicación?", vbYesNo)
If iRespuesta = vbNo Then
        Cancel = True
Else
        End
End If
```

II.- Centrar los formularios

1.- Establece la propiedad **StartUpPosition** a (2 – CenterScreen) de los formularios **frmAcceso** y **frmPrincipal**, para centrar los formularios en la pantalla.
2.- Guarda y prueba la aplicación.

PRÁCTICA 4: ESCRIBIR PROCEDIMIENTOS

En esta práctica, agregarás funcionalidad al proyecto de aplicación que estima el pago de un préstamo iniciado en la práctica 2 y continuado en la práctica 3.
En esta práctica, se iniciará la implementación del formulario principal de la aplicación que estima el pago de un préstamo. El propósito de esta aplicación es hacer cálculos de pagos mensuales y la cantidad total que se ha de pagar por un préstamo.

Iniciarás escribiendo una función que calcula el pago mensual de un préstamo; para esta práctica, vamos a suponer que el término del préstamo es de treinta años y la tasa de interés es de 6.5%. En la práctica 6 "Trabajando con Controles" agregarás controles para recuperar esta información del usuario. Puedes continuar trabajando con los archivos que creaste en la práctica 3, o puedes trabajar con los archivos que se proveen para ti en (Carpeta de Instalación)\Prácticas\Practica04.

Objetivos

Después de completar está práctica serás capaz de:
- Crear una función.
- Llamar una función desde dentro de un procedimiento.
- Agregar un modulo al proyecto.

Ejercicios

Ejercicio 1: Crear la interfaz del usuario
En este ejercicio, crearás una interfase de usuario simple para probar las funciones que escribas.

Ejercicio 2: Escribir una función
En este ejercicio, escribirás una función para calcular la cantidad de pagos mensuales de un préstamo.

Ejercicio 3: Llamar a la función PagoMensual
En este ejercicio, llamarás a la función PagoMensual, que escribiste en el Ejercicio 2, desde el botón de comando Pago Mensual que se encuentra en el formulario frmPrincipal.

Ejercicio 4: Agregar un archivo de módulo
En este ejercicio, agregarás un módulo a tu proyecto. El nuevo módulo contiene una función que calcula la cantidad total a ser pagada al término del préstamo. Llamarás esta función para desplegar la cantidad Total del préstamo a pagar.

Ejercicio 5: Agregar formato a la ventana de mensaje [MsgBox]
En este ejercicio, agregarás formato a las ventanas de mensaje en tu aplicación.

Ejercicio 1: Crear la interfaz del usuario

En este ejercicio, crearás una interfaz de usuario simple para probar las funciones que escribas.

I.- Abre el proyecto de la aplicación Prestamo

1.- Abre el proyecto de la aplicación préstamo con el que has trabajado o abre el proyecto de la aplicación localizado en (Carpeta de Instalación)\Prácticas\Practica04.

II.- Personaliza el ambiente del código

1.- Sobre el menú **Herramientas**, da clic en **Opciones**.
2.- Selecciona **Requerir declaración de Variables**, y después da clic en **Aceptar**.

III.- Agrega option explicit al formulario existente

1.- Abre la ventana del editor de código para **frmAcceso.**
2.- En la lista desplegable de Objeto, selecciona General, y en la lista desplegable de Procedimiento, selecciona Declaraciones.

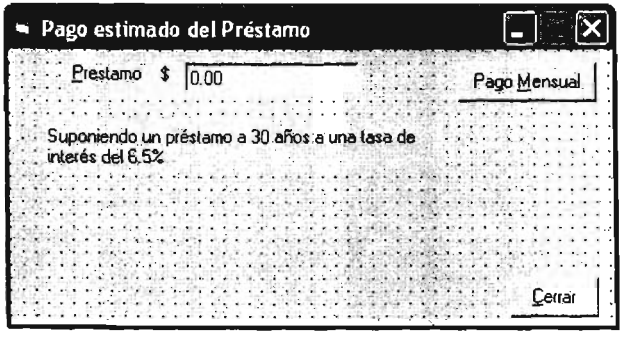
Tu cursor debe estar en la sección General de Declaraciones. Esta sección es la parte superior de la ventana del editor de código, antes de cualquier procedimiento de evento o procedimiento general.

3.- Agrega la siguiente línea de código:
option explicit

4.- Agrega la sentencia option explicit en la sección General de Declaraciones de **frmPrincipal.**

IV.- Agrega controles a frmPrincipal

1.- Agrega controles al formulario **frmPrincipal** de tal manera que se parezca a la siguiente ilustración:



2.- Nombra el Textbox y el CommandButton con nombres que representen su propósito, tales cómo **txtPrestamo** y **cmdMensual**
3.- Selecciona el control **txtPrestamo** y establece la propiedad **TabIndex** a **0**.

Ejercicio 2: Escribir una función

En este ejercicio, escribirás una función para calcular la cantidad de pagos mensuales de un préstamo.

Los pagos mensuales de un préstamo dependen del tiempo del préstamo, la tasa de interés y la cantidad prestada.

I.- Crea una función para calcular los pagos mensuales

1.- Abre la ventana del editor de código de frmPrincipal
2.- Sobre el menú Herramientas, da clic en **Agregar procedimiento**.
3.- Establece el nombre del procedimiento a **PagoMensual**, establece el Tipo a **Función** y establece el Alcance a **Público** y da clic en **Aceptar**.
Una plantilla de función debe ser agregada al formulario:

Public Function PagoMensual ( )

End Function

4.- Establece el tipo de regreso de la función a Double, agregando la palabra reservada **As Double** a la plantilla de la función.

Public Function PagoMensual ( ) As Double

5.- Completa el contenido de la función:
   a.- Declara una variable de tipo Double para sostener la tasa de interés mensual, dblTasaMensual.
   b.- Declara una variable de tipo Integer para sostener el número de pagos, intNumPagos.
   c.- Declara una variable de tipo Double para sostener la cantidad del préstamo, dblCantPrestamo.
   d.- Convierte el texto contenido en el TextBox Préstamo a un tipo Double y guarda el valor como dblCantPrestamo.
   e.- Declara una constante a nivel formulario para representar el número de meses en un año (12) en varios lugares.

   Private Const PERIODO_CONVERSION as integer = 12

   f.- Suponiendo que el préstamo será a treinta años; calcula el número de pagos que serán hechos (30 * PERIODO_CONVERSION), y guarda este valor en la variable intNumPagos.

g.- Suponiendo que la tasa de interés es del 6.5%, o .065, calcula la tasa de interés mensual (0.065/12 o usa la constante declarada en el inciso e; .065/PERIODO_CONVERSION, y guarda este valor en la variable dblTasaMensual.

dblTasaMensual = .065 / PERIODO_CONVERSION

h.- Usa la función **Pmt** integrada en Visual Basic para calcular la cantidad que se va pagar mensualmente y asigna esta cantidad como el valor que va a regresar la función.

PagoMensual = Pmt (Rate:= dblTasaMensual, NPer:= intNumPagos,
PV:= -dblCantPrestamo)

Ejercicio 3: Llamar a la función PagoMensual

En este ejercicio, llamarás a la función PagoMensual, que escribiste en el Ejercicio 2, desde el botón de comando Pago Mensual que se encuentra en el formulario frmPrincipal.

I.- Muestra los pagos mensuales

1.- Da doble clic al botón de comando Pago Mensual que se encuentra en el formulario frmPrincipal.

La ventana del editor de código debe abrir, con una plantilla para el procedimiento del evento **Click**.

2.- Completa el contenido del procedimiento del evento **Click** con código que llame a la función **PagoMensual** y muestre el resultado en una ventana de mensaje [Message Box]:

a.- Declara una variable de tipo Double para sostener el valor que regresa la función PagoMensual, dblMensual.

b.- Llama a la función **PagoMensual** y almacena el valor que regresa en la variable dblMensual.

c.- Crea una ventana de mensaje [Message Box] para mostrar el valor del pago mensual.

El contenido completo que debe tener el procedimiento del evento Click debe parecerse a lo siguiente:

```
Private Sub  cmdMensual_Click ( )
        Dim dblMensual As Double
        dblMensual = PagoMensual ( )
        MsgBox "Tus pagos mensuales serán de: "  & dblMensual
End Sub
```

3.- Guarda y prueba tu aplicación.

Ejercicio 4: Agregar un archivo de módulo

En este ejercicio, agregarás un módulo a tu proyecto. El nuevo módulo contiene una función que calcula la cantidad total a ser pagada al término del préstamo. Llamarás a esta función para desplegar la cantidad Total del préstamo a pagar.

La función para calcular los pagos totales ya está creada. Esta función se llama TotalPagado y está en un módulo Totalpd.bas que se encuentra en (Carpeta de Instalación) \Prácticas\Practica04.

I.- Agrega un módulo a tu proyecto

1.- Sobre el menú Proyecto, da clic en **Agregar módulo**.
2.- Da clic en la etiqueta Existente.
3.- Abre la carpeta (Carpeta de Instalación)\Prácticas\Practica04, selecciona el archivo Totalpd.bas y da clic en Abrir.
Éste agregará el módulo a tu proyecto. Para visualizar el procedimiento que se encuentra en el módulo, da clic al botón derecho del ratón sobre el módulo de la ventana de Proyecto y da clic en **Ver código**.
4.- Analiza la función TotalPagado del módulo. Observa que la función acepta un parámetro de tipo Entero, el cuál es el número de años del préstamo y el valor de regreso de la función es de tipo Double, además la función calcula la cantidad total del préstamo que será pagada al término del préstamo.

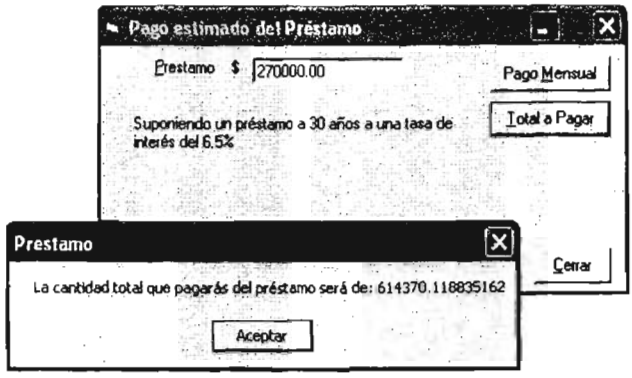
II.- Muestra el pago total del préstamo.

1.- Agrega un botón de comando a frmPrincipal para mostrar la cantidad total a pagar del préstamo.
2.- En la propiedad **Nombre** del botón de comando cambia el nombre de tal manera que represente su función, tal como **cmdTotal**.
3.- En el procedimiento del evento **Click** del botón de comando llama a la función **TotalPagado** y muestra el resultado en una ventana de mensaje [Message box] de tal manera que muestre el valor de la cantidad total a pagar.
   a.- Declara una variable de tipo Double para sostener el valor de regreso de la llamada de la función TotalPagado, dblTotal.
   b.- Llama a la función TotalPagado, pasándole el valor de treinta como el valor del parámetro, y almacena el valor de retorno de la función en la variable dblTotal.
   c.- Crea una ventana de mensaje [Message box] que muestre el valor de la cantidad total a pagar.

El procedimiento del evento finalizado debe parecerse al siguiente ejemplo de código:

```
Private Sub cmdTotal_Click ( )
        Dim dblTotal as Double
        dblTotal = TotalPagado( 30 )
        MsgBox "La cantidad total que pagarás del préstamo será de: " & dblTotal
End Sub
```

Cuando ejecutes tu aplicación finalizada, debe parecerse cómo la siguiente ilustración:



4.- Guarda y prueba tu aplicación.

Ejercicio 5: Agregar formato a la ventana de mensaje [MsgBox]

En este ejercicio, agregarás formato a las ventanas de mensaje en tu aplicación.

Cuándo muestras un valor monetario, debes dar formato a los valores de acuerdo a la localidad dónde tu aplicación se este ejecutando; por ejemplo, podrías usar el signo "$" para México y Estados Unidos. La función Format de Visual Basic nos facilita la representación de éste.

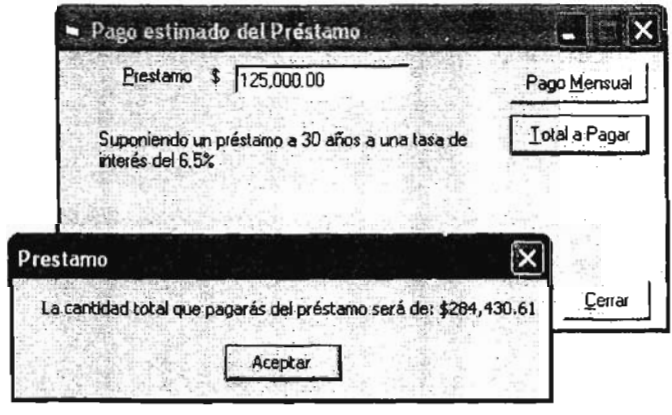
I.- Dar Formato en la ventana de mensaje [Message Box]

1.- Edita el procedimiento del evento cmdMensual_Click
2.- Cambia la cadena de MsgBox para incluír una llamada a la función Format antes de mostrar el valor del pago mensual.

MsgBox "Tus pagos mensuales serán de: " & Format(dblMensual, "currency")

3.- Edita el procedimiento del evento cmdTotal_Click, y repite el paso 2 para mostrar los valores del pago total.
4.- Guarda y prueba tu aplicación.

Cuando ejecutes tu aplicación finalizada, debe parecerse cómo la siguiente ilustración



PRÁCTICA 5: CONTROLAR EL FLUJO DEL PROGRAMA

En esta práctica, agregarás funcionalidad para validar la contraseña del proyecto préstamo que iniciaste en la práctica 2. El formulario frmPrincipal verificará una contraseña válida sobre el formulario frmAcceso y saldrá de la aplicación si la contraseña es inválida. Además limitarás el número de veces que el usuario teclee una contraseña inválida.

Puedes continuar trabajando con los archivos que creaste en la práctica 4, o puedes trabajar con los archivos que se proveen para ti en (Carpeta de Instalación)\Prácticas\Practica05.

Objetivos

Después de completar está práctica serás capaz de:

- Agregar una declaración If para verificar la contraseña del programa de acceso.
- Usar una variable de tipo estática para contar el número de intentos en el programa de acceso.
- Regresar los valores de los controles a su estado original.
- Usar una variable pública para pasar información entre dos formularios.

Ejercicios

Ejercicio 1: Validar información del programa de Acceso
En este ejercicio, agregarás validación a la contraseña del formulario programa de acceso, para usar el formulario principal.

Ejercicio 2: Limitar los intentos en el programa de Acceso
En este ejercicio, limitarás el número de veces que un usuario puede introducir su contraseña, para usar tu aplicación.

Ejercicio 1: Validar información del programa de Acceso

En este ejercicio, agregarás validación a la contraseña del formulario programa de acceso. para usar el formulario principal.

I.- Abre el proyecto préstamo

1.- Si Visual Basic no se ejecuta, inicia éste.
2.- Abre el proyecto de la aplicación préstamo con el que has trabajando o abre el proyecto de la aplicación localizado en (Carpeta de Instalación)\Prácticas\Practica05.

II.- Agrega una constante a nivel formulario

1.- Abre la ventana del editor de código del formulario **frmPrincipal** y desplázate a la sección **General de Declaciones.**
2.- Declara una constante de tipo cadena, CONTRASEÑA y establece su valor a "contraseña":

Const CONTRASEÑA As String = "contraseña"

III.- Valida la contraseña introducida desde el formulario frmPrincipal

1.- En la ventana del editor de código de **frmPrincipal**, desplázate al procedimiento del evento **Form_Load**. Actualmente, este procedimiento de evento muestra el formulario **frmAcceso** modalmente.
2.- Declara una variable estática de tipo entera, sintIntentosAcceso, para contar el número de veces que un usuario ha intentado introducir su contraseña.

Static sintIntentosAcceso As Integer

3.- Escribe un circuito que continúe hasta que la contraseña tecleada dentro del TextBox **txtContraseña** sobre el formulario **frmAcceso** sea igual a la constante CONTRASEÑA.
4.- Escribe el contenido del circuito con código que:
   a.- Incremente el contador estático, sintIntentosAcceso, por 1.
   b.- Enviar una ventana de mensaje [Message box] indicando que la contraseña es inválida.
   c.- Limpia el TextBox **txtContraseña** sobre el formulario **frmAcceso**
   d.- Muestra **frmAcceso** modalmente.

Tu circuito debe parecerse a lo siguiente:

```
Do While frmAcceso.txtContraseña.Text <> CONTRASEÑA
        sintIntentosAcceso = sintIntentosAcceso + 1
        MsgBox "Has tecleado una contraseña inválida", vbOKOnly + vbInformation
        frmAcceso.txtContraseña.Text = ""
        frmAcceso.Show vbModal
Loop
```

IV.- Oculta frmAcceso en vez de descargar esta.

Para recuperar el valor del TextBox **txtContraseña** del formulario **frmAcceso** después de que ha sido desechado, la forma debe ser ocultada en vez de ser descargada.

1.- Abre la ventana del editor de código de **frmAcceso**.
2.- En el procedimiento del evento **cmdAceptar_Click**, oculta la forma en vez de descargarla:

frmAcceso.Hide

V.- Guarda y prueba tu aplicación.

1.- Guarda tu proyecto
2.- Ejecuta tu aplicación. La contraseña valida es "contraseña". Intenta teclear una contraseña inválida y da clic en **Aceptar**. ¿Qué ocurrió? Ahora teclea la contraseña valida y da clic en **Aceptar**. ¿Qué ocurrió?
3.- Cierra la aplicación que se esta ejecutando.

Ejercicio 2: Limitar los intentos en el programa de Acceso

En este ejercicio, limitarás el número de veces que un usuario puede introducir su contraseña, para usar tu aplicación.

I.- Limitar el número de intentos para acceder a la aplicación.

1.- Abre la ventana del Editor de Código de **frmPrincipal**, y declara una constante de tipo entera llamada MAX_INTENTOS_ACCESO en el procedimiento de evento **Form_Load** de la siguiente manera:

'Este es el número máximo de veces que la aplicación permitirá a un usuario para intentar acceder a la aplicación.
Const MAX_INTENTOS_ACCESO As Integer = 3

2.- Dentro del circuito que verifica la validación de la contraseña, después de que incrementas el contador estático, sintIntentosAcceso, agrega una declaración **If** para comparar sintIntentosAcceso con la constante MAX_INTENTOS_ACCESO:

sintIntentosAcceso = sintIntentosAcceso + 1
if sintIntentosAcceso < MAX_INTENTOS_ACCESO Then

3.- Si el número de intentos es menor que la constante, maneja los intentos inválidos como lo has estado manejando en una ventana de mensaje [Message box] y mostrando frmAcceso.

4.- Si el número de intentos es mayor o igual que el número de intentos máximo permitido. entonces envía una ventana de mensaje [Message box] mencionando que el usuario ha intentado demasiadas veces acceder a la aplicación, y finaliza la aplicación.

```
Else
    MsgBox Prompt:= "Demasiados intentos de acceso inválidos. ", _
        Buttons:= vbOKOnly + vbExclamation
    End
End If
```

5.- Guarda y prueba tu aplicación.

PRÁCTICA 6: TRABAJAR CON CONTROLES

En esta práctica, agregarás funcionalidad al proyecto préstamo iniciado en la práctica 2. Puedes continuar trabajando con los archivos que has creado, o puedes usar los archivos que son proveídos en (Carpeta de Instalación)\Prácticas\Practica06.

En la práctica 4, algunas adaptaciones fueron hechas acerca de la duración del préstamo y la tasa de interés. En esta práctica, agregarás controles a frmPrincipal para recuperar esta información del usuario y entonces introducir esos valores dentro de las funciones que escribiste en la práctica 4.

Objetivos

Después de completar esta práctica serás capaz de:

- Agregar controles a un formulario
- Responder a los eventos de los controles
- Recuperar valores desde los controles

Ejercicios

Ejercicio 1: Crear la interfaz del usuario
En este ejercicio, agregarás controles a **frmPrincipal** para recuperar información adicional acerca del préstamo del usuario.

Ejercicio 2: Implementar la Interfaz del Usuario
En este ejercicio, agregarás código para guardar valores de los botones de opción **Terminación del préstamo** e inicializar los controles sobre frmPrincipal cuándo el formulario es cargado.

Ejercicio 3: Recuperar los valores del usuario
En este ejercicio, usarás los valores de los controles para calcular los pagos mensuales y la cantidad total a pagar.

Ejercicio 4: Agregar un botón de resumen
En este ejercicio, agregarás un botón de comando que muestre el resumen de la información.

Ejercicio 1: Crear la interfaz del usuario

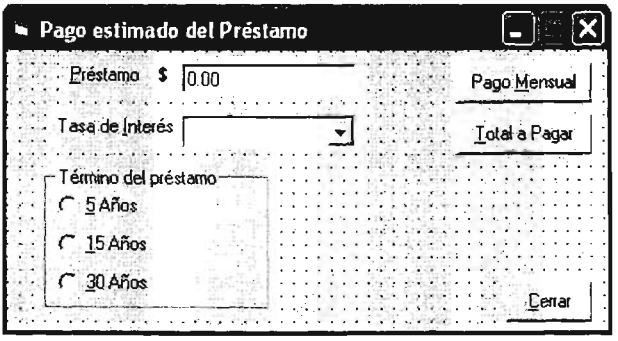
En este ejercicio, agregarás controles a **frmPrincipal** para recuperar información adicional acerca del préstamo del usuario.

I.- Abre el proyecto préstamo

1.- Abre el proyecto de la aplicación préstamo con el que has trabajado o abre el proyecto de la aplicación localizado en (Carpeta de Instalación)\Prácticas\Practica06.

II.- Agrega controles a frmPrincipal

1.- Agrega controles a frmPrincipal de tal manera que parezca a la siguiente ilustración:



2.- Establece las propiedades para los controles como es mostrado en la siguiente tabla.

Nota: Para establecer la propiedad List de los controles ListBox o ComboBox en modo diseño, usa la combinación de teclas CTRL + ENTER después de introducir cada elemento en la lista.

Tipo de Control	Propiedad	Configuraciones deseadas
ComboBox	Nombre Style List	cboInteres 0 – DropDown Combo Coloca los siguientes valores: 4.5, 6.25, 7, 8.325, 9, 10
Label	Caption	&Tasa de Interés
Frame	Nombre Caption	fraTermino Término del préstamo
OptionButton	Nombre Caption Index	optDuracion &5 Años 0

Tipo de Control	Propiedad	Configuraciones deseadas
OptionButton	Nombre Caption Index	optDuracion &15 Años 1
OptionButton	Nombre Caption Index	optDuracion &30 Años 2

Ejercicio 2: Implementar la interfaz del usuario

En este ejercicio, agregarás código para guardar valores de los botones de opción **Terminación del préstamo** e inicializar los controles sobre frmPrincipal cuándo el formulario es cargado.

I.- Implementa un grupo de botones de opción en un arreglo de controles.

1.- Abre la ventana del editor de código de **frmPrincipal**.
2.- En la sección general de declaraciones, declara una variable de tipo entera, mintDuracion, para almacenar el valor del botón de opción seleccionado, que indica la terminación del préstamo.
3.- Edita el evento **click** para los botones de opción **optDuracion**. Usando una declaración **Select Case**, fija mintDuración al número de años correspondiente al control seleccionado. Por ejemplo, si el índice es 0, el número de años es 5.

```
Select Case Index
    Case 0
        mintDuracion = 5
    Case 1
        mintDuracion = 15
    Case 2
        mintDuracion = 30
End Select
```

Nota: Los botones de opción son un ejemplo de arreglos de controles. Un control de arreglo es un grupo de controles que comúnmente comparten nombres, tipo y procedimientos de los eventos. Cada control tiene un índice único. Cuando un control en el arreglo reconoce un evento, éste llama el procedimiento de evento para el grupo y pasa el índice como un parámetro, permitiéndole a tu código determinar que control reconoció el evento.

II.- Inicializa los controles en el procedimiento de evento Form_Load

1.- En la ventana del editor de código de **frmPrincipal**, desplázate al evento **Form_Load**.
2.- Después del circuito (loop), agrega código de inicialización
a.- Inicializa la propiedad **Text** de **txtPrestamo** a "0.00."

b.- Inicializa la propiedad **Text** de **cboInteres** a "4.25."
c.- Fija la propiedad **Value** del botón de opción **5 Años** a **True**.
d.- Llama al evento **Click** del botón de opción **5 Años**, para que el procedimiento de evento fije una variable a nivel formulario.

El segmento de código debe parecerse al siguiente:

```
txtPrestamo.Text = "0.00"
cboInteres.Text = "4.25"
optDuracion(0).Value = True
optDuracion_Click 0
```

3.- Guarda y prueba tu aplicación.

Ejercicio 3: Recuperar los valores del usuario

En este ejercicio, usarás los valores de los controles para calcular los pagos mensuales y la cantidad total a pagar.

Ahora que hay controles para obtener información del préstamo del usuario, borra o comenta los valores codificados en la función **PagoMensual**, escrito en la práctica 4, e introduce los valores desde los controles.

I.- Usa los valores de los controles en la función PagoMensual

1.- En la ventana del editor de código de **frmPrincipal**, desplázate a la función **PagoMensual**.
2.- Declara una variable de tipo Double para sostener la tasa de interés, dblInteres.
3.- Recupera la tasa de interés del la lista desplegable del ComboBox, convierte ésta a un valor de tipo Double, convierte ésta a porcentaje y guarda ésta como dblInteres:

```
dblInteres = CDbl ( cboInteres.Text) / 100
```

4.- Usa el valor en la variable guardada mintDuracion y el valor de dblInteres, en lugar de las constantes 30 y 0.065, en el cálculo de intNumPagos y dblTasaMensual.
5.- Guarda y prueba tu aplicación.

II.- Usa los valores de los controles en la función TotalPagado

1.- En la ventana del editor de código de **frmPrincipal**, desplázate al procedimiento de evento **cmdTotal_Click**.
2.- Pasa el valor de la variable guardada mintDuracion, en lugar de la constante 30, a la función **TotalPagado**.
3.- Guarda y prueba tu aplicación.

Ejercicio 4: Agregar un botón de resumen

En este ejercicio, agregarás un botón de comando que muestre el resumen de la información.

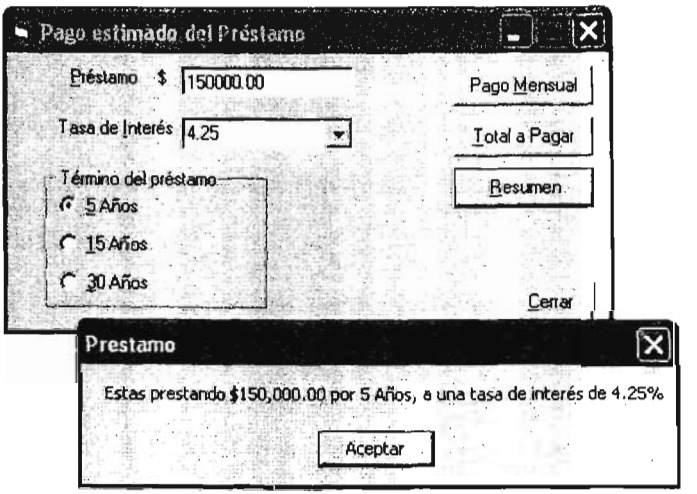
I.- Agrega un botón de resumen

1.- Agrega un botón de comando [CommandButton] para mostrar un resumen de la información introducida sobre el formulario.
2.- En el evento click para el botón resumen, muestra los valores introducidos sobre el formulario en una ventana de mensaje [Message box]:

El procedimiento debe parecerse al siguiente código:

```
Private Sub Command1_Click()
MsgBox "Estas prestando " & Format(txtPrestamo.Text, "currency") & _
    " por " & mintDuracion & " Años, a una tasa de interés de " _
    & cboInteres.Text & "%"
End Sub
```

3.- Guarda y prueba tu aplicación.
Cuando pruebes la aplicación, dando clic en el botón Resumen, debe mostrarse una imagen como la siguiente:



PRÁCTICA 7: ACCESAR BASES DE DATOS

En esta práctica, construirás una aplicación que contiene información de pedidos de clientes. El formulario principal de la aplicación mostrará información de clientes, mientras que un segundo formulario mostrará el historial de los pedidos de un cliente seleccionado.

Objetivos

Después de completar está práctica serás capaz de:

- Usar el Asistente para formulario de datos
- Crear una declaración SQL
- Trabajar con el control ADO Data

Ejercicios

Ejercicio 1: Creación de un formulario con información del cliente usando el asistente para formulario de datos

En este ejercicio, usarás el asistente de formulario de datos para crear un formulario que mostrará información del cliente.

Ejercicio 2: Creación de un formulario con información de los pedidos usando el asistente para formulario de datos

En este ejercicio, usarás el asistente de formulario de datos para diseñar un formulario que mostrará información de los pedidos. Entonces, agregarás un botón de comando al formulario frmClientes para mostrar el formulario frmPedidos.

Ejercicio 3: Crear una declaración SQL

En este ejercicio, ligarás los formularios Clientes y Pedidos para que sólo los pedidos de un cliente seleccionado sean mostrados en el formulario frmPedidos.

Ejercicio 1: Creación de un formulario con información del cliente usando el asistente para formulario de datos

En este ejercicio, usarás el Asistente de formulario de datos para crear un formulario que mostrará información del cliente.

I.- Instala el complemento Data Form Wizard

1.- Abre Visual Basic e inicia un nuevo proyecto Visual Basic **EXE estándar**
2.- Si el Asistente para formulario de datos no es un elemento del menú **Complementos**, entonces sobre el menú **Complementos** da clic en **Administrador de complementos** para cargar el complemento **VB 6 Data Form Wizard**.
3.- De la lista de complementos disponibles, localiza **VB 6 Data Form Wizard** y da doble clic sobre este, o selecciona **VB 6 Data Form Wizard**, y da clic en **Cargado/Descargado**, después da clic en **Aceptar**.

II.- Construye el formulario con información del cliente

1.- Sobre el menú Complementos, da clic en **Asistente para formulario de datos**.
2.- Da clic en el botón **Siguiente** de la pantalla de Introducción del **Asistente para formulario de datos**.
3.- Selecciona **Access** de la lista de Tipo de base de datos, y da clic en **Siguiente**.
4.- Da clic en **Examinar** y selecciona la base de datos **NWIND.MDB** de (Carpeta de Instalación)\Prácticas \Practica07 y da clic en **Siguiente**.
5.- Nombra el formulario **frmClientes**, en el **Diseño del formulario** selecciona **Registro individual**, en el **Tipo de enlace** selecciona **Control de datos ADO** y da clic en **Siguiente**.
6.- En la lista desplegable de **Origen de registros**, selecciona **Customers**.
7.- Agrega los campos CustomerID y CompanyName de la lista de **Campos disponibles** a la lista de **Campos seleccionados**, y da clic en **Siguiente**.
8.- Da clic en **Selec. Todo** y da clic en **Siguiente**.
9.- Da clic en **Finalizar**, y da clic en **Aceptar** en el cuadro de diálogo **Formulario de datos creado**.

III.- Personalizar el formulario

1.- Para personalizar el formulario cambiaras las propiedades de los siguientes objetos:

Tipo de Objeto	Propiedad	Configuración deseada
frmClientes	Caption	Clientes
lblLabels	Caption	ID del Cliente:
lblLabels	Caption	Nombre de la Compañía:

En esta aplicación, no estarás usando el formulario de default que Visual Basic crea.

IV.- Borra el formulario de default y fija el formulario inicial

1.- En la ventana del explorador de proyectos, selecciona **Form1**.
2.- Sobre el menú **Proyecto**, da clic en **Quitar Form1**.
3.- Sobre el menú **proyecto**, da clic en **Propiedades de Proyecto1.**
4.- Nombra el proyecto **ProyectoDatos** y establece el **Objeto inicial** a **frmClientes**, que es el formulario que creaste, y da clic en **Aceptar**.
5.- Guarda el proyecto en (Carpeta de Instalación)\Prácticas\Practica07 y prueba tu aplicación. Usa los botones de avance y retroceso del control **ADO Data Control** para pasar a través de algunos registros de la base de datos. Cambia el nombre del campo del Nombre de la compañía y da clic sobre el botón **Actualizar** o cambia de registro para guardar los cambios.
6.- Da clic en **Cerrar** para finalizar la aplicación.

Ejercicio 2: Creación de un formulario con información de los pedidos usando el asistente para formulario de datos

En este ejercicio, usarás el asistente de formulario de datos para diseñar un formulario que mostrará información de los pedidos. Entonces, agregarás un botón de comando al formulario frmClientes para mostrar el formulario frmPedidos.

I.- Construye el formulario de información de Pedidos

1.- Sobre el menú Complementos, da clic en Asistente para formulario de datos.
2.- Da clic en el botón **Siguiente** de la pantalla de Introducción del **Asistente para formulario de datos**.
3.- Selecciona **Access** de la lista de Tipo de base de datos, y da clic en **Siguiente**.
4.- Da clic en **Examinar** y selecciona la base de datos **NWIND.MDB** de (Carpeta de Instalación)\Prácticas \Practica07 y da clic en **Siguiente**.
5.- Nombra el formulario **frmPedidos**, en el **Diseño del formulario** selecciona **Cuadrícula(Hoja de datos)**, en el **Tipo de enlace** selecciona **Control de datos ADO** y da clic en **Siguiente**.
6.- En la lista desplegable de **Origen de registros**, selecciona **Orders**.
7.- Agrega los campos CustomerID, OrderID, OrderDate y RequiredDate de la lista de **Campos disponibles** a la lista de **Campos seleccionados**, y da clic en **Siguiente**.
8.- Da clic en **Selec. Todo** y da clic en **Siguiente**.
9.- Da clic en **Finalizar**, y da clic en **Aceptar** en el cuadro de diálogo **Formulario de datos creado**.

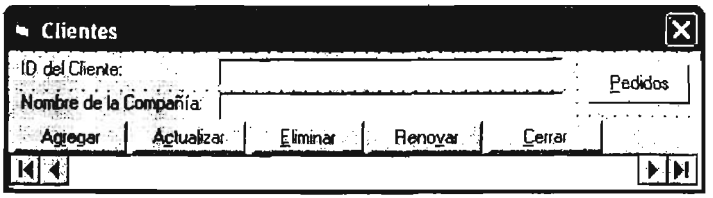
II.- Personalizar el formulario

1.- Para personalizar el formulario cambiaras la propiedad del siguiente objeto:

Tipo de Objeto	Propiedad	Configuración deseada
frmPedidos	Caption	Pedidos

III.- Muestra el formulario frmPedidos desde el formulario frmClientes.

1.- Amplía el formulario frmClientes y agrega un botón de comando Pedidos:



2.- En el evento click del botón de comando Pedidos, muestra el formulario frmPedidos de tipo modal:

frmPedidos.Show vbModal

IV.- Guarda y prueba tu aplicación

1.- Guarda el proyecto.
2.- Ejecuta la aplicación. Da clic sobre el botón **Pedidos** y desplázate sobre algunos de los registros de la tabla **Pedidos** e intenta redimensionar el formulario.
3.- Da clic en **Cerrar** sobre frmPedidos para regresar al formulario frmClientes.
4.- Da clic en **Cerrar** para finalizar la aplicación.

Ejercicio 3: Crear una declaración SQL

En este ejercicio ligarás los formularios clientes y pedidos para que sólo los pedidos de un cliente seleccionado sean mostrados en el formulario frmPedidos.

I.- Personaliza el formulario frmPedidos

1.- Abre la ventana del editor de código de frmPedidos.
2.- En la sección General de Declaraciones, crea una variable pública de tipo cadena (string), gstrIDCliente.

Public gstrIDCliente As String

Esta variable será establecida desde el formulario frmClientes.

3.- En el procedimiento de evento Form_Load de frmPedidos:
   a.- Crea una cadena SQL que seleccione todos los campos de la tabla Orders, dónde CustomerID es igual a la variable gstrIDCliente.
   b.- Establece la propiedad RecordSource de ADO Data Control igual a la cadena SQL.
   c.- Actualiza ADO Data Control usando la propiedad Refresh.

```
Dim strSQL As String
strSQL = "Select * From Orders " & _
         "Where CustomerID = " & Chr(34) & _
         gstrIDCliente & Chr(34)
datPrimaryRS.RecordSource = strSQL
datPrimaryRS.Refresh
```

II.- Personaliza el formulario frmClientes

1.- Edita el procedimiento de evento click del botón Pedidos de frmClientes.
2.- Antes de mostrar el formulario frmPedidos, establece la variable pública gstrIDCliente igual al texto actual en el campo CustomerID:

```
frmPedidos.gstrIDCliente = txtFields ( 0 ).Text
```

III.- Guarda y prueba tu aplicación

1.- Guarda tu proyecto
2.- Ejecuta la aplicación. Selecciona varias compañías, y para cada una, da clic en el botón **Pedidos**. ¿Son todos los registros desplegados en el formulario frmPedidos de la compañía seleccionada?
3.- En el formulario frmPedidos, da clic en **Cerrar** y regresa al formulario frmClientes.
4.- Da clic en **Cerrar** para finalizar la aplicación.

PRACTICA 8: VALIDACION DE ENTRADA

En esta práctica, escribirás código y establecerás propiedades para controlar cómo el usuario interactúa con tu aplicación préstamo. Para implementar estos pasos harás esto mas fácilmente, para que los usuarios introduzcan datos válidos. Además de que agregarás controles que proveen una extensa realimentación visual para ayudar a un usuario en el uso de tu interfaz.

Objetivos

Después de completar esta práctica serás capaz de:

- Establece propiedades de validación de un TextBox
- Usa el evento Validate para validar entradas del usuario
- Válidar datos numéricos introducidos por el usuario

Ejercicios

Ejercicio 1: Agregar código al evento [Validate]
En este ejercicio, agregarás código al evento validate de los TextBox Usuario y Contraseña. Este código verificará que el usuario introduce texto en ambos campos.

Ejercicio 2: Validar datos numéricos
Las funciones PagoMensual y TotalPagado necesitan valores numéricos en el TextBox txtPrestamo. Llamar cualquier función con una cadena en el TextBox causa un error al momento de la ejecución. En este ejercicio, te protegerás en contra de este tipo de errores al verificar que los TextBox tienen un valor numérico antes de llamar a las funciones.

Ejercicio 1: Agregar código al evento |Validate|

En este ejercicio, agregarás código al evento Validate de los TextBox Usuario y Contraseña. Este código verificará que el usuario introduce texto en ambos campos.

I.- Abre el proyecto préstamo

1.- Abre el proyecto de la aplicación préstamo con el que has trabajado o abre el proyecto de la aplicación localizado en (Carpeta de Instalación)\Prácticas\Practica08.

II.- Codifica el evento Validate

1.- Establece la propiedad **CausesValidation** del botón Cancelar de frmAcceso a **False**.

Nota: El evento **Validate** ocurre sólo cuando la propiedad **CausesValidation** del control que esta apunto de recibir el foco es establecido a **True**. El evento **Validate** además incluye un argumento **Cancel** que, cuándo es establecido a **True**, permite al control a mantener el foco.

2.- Agrega código al evento **Validate** para **txtUsuario** y **txtContraseña** que verificará que los datos fueron introducidos dentro de cada TextBox y establece el foco de regreso al correspondiente TextBox en el caso de que ningún dato haya sido introducido.

```
Private Sub txtUsuario_Validate (Cancel As Boolean)
        If txtUsuario.Text = "" Then
                MsgBox "Favor de introducir el nombre de Usuario"
                Cancel = True
        End if
End Sub

Private Sub txtContraseña_Validate (Cancel As Boolean)
        If txtContraseña.Text = "" Then
                MsgBox "Favor de introducir su contraseña"
                Cancel = True
        End if
End Sub
```

3.- Guarda el proyecto y prueba tu aplicación.

Ejercicio 2: Validar datos numéricos

Las funciones PagoMensual y TotalPagado necesitan valores numéricos en el TextBox txtPrestamo. Llamar cualquier función con una cadena en el TextBox causa un error al momento de la ejecución. En este ejercicio, te protegerás en contra de este tipo de errores al verificar que los TextBox tienen un valor numérico antes de llamar a las funciones.

1.- Verificar un valor inválido

1.- Ejecuta la aplicación préstamo, introduce un valor no numérico dentro del TextBox **txtPrestamo**, y da clic en **Pago Mensual**.
Un error al momento de la ejecución debe ocurrir **No coinciden los tipos**.
2.- En la caja de diálogo **Error** de Visual Basic, da clic en **Terminar**.
3.- Abre la ventana del Editor de código de frmPrincipal y desplázate al procedimiento del evento **cmdMensual_Click**.
4.- Antes de llamar a la función **PagoMensual**, prueba para ver si el valor introducido en **txtPrestamo** es un valor numérico:

If IsNumeric (txtPrestamo.Text) Then

5.- Si el valor es numérico llama a la función **PagoMensual** y muestra el resultado en una ventana de mensaje [Message box].
6.- Si el valor no es numérico:
a.- Muestra una ventana de mensaje [Message box] instruyendo al usuario a introducir un dato numérico.
b.- Establece el foco de regreso al TextBox **txtPrestamo**.
c.- Resalta el texto inválido del TextBox.
El segmento de código debe parecerse al siguiente:

```
MsgBox "Favor de introducir únicamente valores numéricos", vbInformation
  With txtPrestamo
       .SetFocus
       .SelStart = 0
       .SelLength = Len(.Text)
  End With
```

7.- Agrega el mismo tipo de verificación de error para el procedimiento de evento **cmdTotal_Click**.
8.- Guarda y prueba tu aplicación.

PRÁCTICA 9: ATRAPAR ERRORES

En esta práctica escribirás el código que usa la declaración **shell,** para intentar ejecutar otra aplicación. Si la declaración **shell** falla, ésta puede producir un error al momento de la ejecución.

Objetivos

Después de completar está práctica serás capaz de:

- Manejar errores en línea.
- Crear un manejador de errores
- Determinar cómo salir del manejador de errores.
- Crear un manejador de errores centralizado.

Ejercicios

Ejercicio 1: Uso de manejadores de errores en línea
En este ejercicio verificarás errores en tu código al usar manejadores de errores en línea.

Ejercicio 2: Crear una rutina de manejadores de errores
En este ejercicio atraparás errores en un procedimiento de evento usando una rutina de manejo de errores.

Ejercicio 3: Crear una función de manejo de errores
En este ejercicio, crearás una función para las aplicaciones que se ejecutan. Esta función regresará un valor verdadero **"True"** si esta fue capaz de ejecutar la aplicación y falso **"False"** si no fue capaz de ejecutarla.

Ejercicio 1: Uso de manejadores de errores en línea

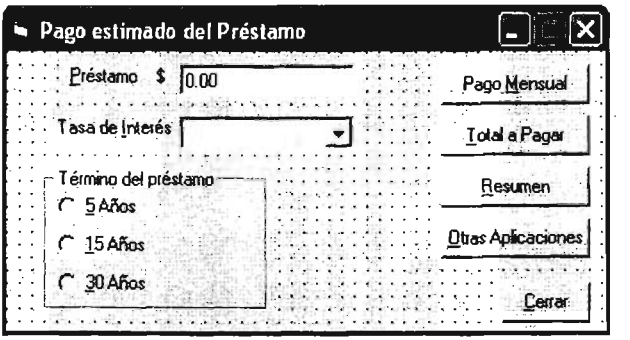
En este ejercicio verificarás errores en tu código al usar manejadores de errores en línea.

I.- Abre el proyecto préstamo

1.- Abre el proyecto de la aplicación préstamo con el que has trabajado o abre el proyecto de la aplicación localizado en (Carpeta de Instalación)\Prácticas\Practica09.

II.- Crea el procedimiento del evento

1.- Agrega un botón de comando a frmPrincipal. Establece la propiedad **Nombre** a **cmdOtraAplicacion**, y la propiedad **Caption** a **&Otras Aplicaciones**, como es mostrado en la siguiente ilustración:



2.- En el procedimiento de evento **click** para el botón **Otras Aplicaciones**, agrega código de tal manera que le pregunte al usuario por el nombre de la aplicación a usar, y emplea la declaración **shell** para ejecutar la aplicación, como es mostrado en el siguiente código.

```
Dim strNombreAplicacion As String
strNombreAplicacion = InputBox ("Para ejecutar la Calculadora, teclea 'calc'. " & _
                                "Para ejecutar el Block de Notas, teclea 'notepad'. ")
Shell strNombreAplicacion, vbNormalFocus
```

3.- Ejecuta la aplicación y prueba el botón **Otras Aplicaciones**, tecleando **calc** (que es la calculadora de Windows).

4.- Prueba el botón **Otras Aplicaciones** nuevamente, tecleando un nombre de aplicación inválida, tal como **ABC**. Esto provocará un error al momento de ejecución. Da clic en el botón **Terminar**.

III.- Agrega código al manejador de error en línea.

1.- En el inicio del procedimiento de evento **cmdOtraAplicacion**, agrega la declaración **On Error Resume Next**, para prevenir a Visual Basic de liberar un error cuándo este ocurra.
2.- Después de la declaración **shell**, verifica la propiedad **Number** del objeto **Err**, para ver si un error ocurrió. Por ejemplo, si un error ocurrió, muestra una ventana de mensaje [Message box] con el número de error.

```
Private Sub cmdOtraAplicacion_Click ( )
'Continúa ejecutándose si un error ocurre
On Error Resume Next
Dim strNombreAplicacion As String
strNombreAplicacion = InputBox("Para ejecutar la Calculadora, teclea 'calc'. " & _
                               "Para ejecutar el Block de Notas, teclea 'notepad'. ")
Shell strNombreAplicacion, vbNormalFocus
'Verifica la propiedad Number para ver si había un error
If Err.Number <> 0 Then
      MsgBox "Imposible encontrar " & strNombreAplicación & _
      vbCrLf & "Error al momento de ejecución. Número: " & Err.Number
End If
End Sub
```

Nota: La constante Visual Basic **vbCrLf** es un retorno de carro que coloca cualquier texto que sigue sobre la siguiente línea.

3.- Prueba la aplicación.

Ejercicio 2: Crear una rutina de manejadores de errores

En este ejercicio, atraparás errores en un procedimiento de evento al usar una rutina de manejo de errores.

I.- Crea un procedimiento de evento

1.- Ejecuta la aplicación, y en el formulario principal, borra el valor del control ComboBox **Tasa de Interés**.
2.- Da clic sobre el botón de comando **Pago Mensual**. Este debe producir un error al momento de ejecución.
3.- ¿Cuál es el número de error al momento de ejecución?
4.- Da clic al botón **Depurar**. Observa la línea de código dónde el error ocurrió.
5.- Da clic al botón **Terminar**.

II.- Crea un manejador de error

1.- Establece un atrapador de error usando la declaración On Error en la función PagoMensual.
2.- Crea la rutina del manejo de errores

En la rutina del manejo de errores, agrega código para probar con diferentes tipos de errores que podrían ser generados. Tú código debe parecerse al siguiente:

```
Public Function PagoMensual() As Double
On Error GoTo Err_PagoMensual

Dim dblTasaMensual As Double
Dim intNumPagos As Integer
Dim dblCantPrestamo As Double
Dim dblInteres As Double

dblCantPrestamo = CDbl(txtPrestamo.Text)

dblInteres = CDbl(cboInteres.Text) / 100

intNumPagos = mintDuracion * PERIODO_CONVERSION

dblTasaMensual = dblInteres / PERIODO_CONVERSION

PagoMensual = Pmt(Rate:=dblTasaMensual, NPer:=intNumPagos, PV:=-
dblCantPrestamo)

Exit_PagoMensual:
  Exit Function
Err_PagoMensual:
  Select Case Err.Number
    Case 13
      MsgBox "No todos los datos fueron introducidos. Los valores que se mostrarán serán
          incorrectos. " & _
          vbCrLf & "Introduce información en los campos y prueba otra vez."
    Case Else
      MsgBox "Error desconocido. Número: " & Err.Number & vbCrLf & _
          Err.Description
  End Select
  Resume Exit_PagoMensual
End Function
```

3.- Ejecuta y prueba la aplicación. ¿El manejador de errores atrapa el error generado en el ejemplo previo?

Ejercicio 3: Crear una función de manejo de errores

En este ejercicio, crearás una función para las aplicaciones que se ejecutan. Esta función regresará un valor Verdadero **"True"** si esta fue capaz de ejecutar la aplicación y falso **"False"** si no fue capaz de ejecutarla.

I.- Crea el procedimiento de la función

1.- En el procedimiento de evento click para el botón Otras Aplicaciones, agrega código que llame una función llamada EjecutaApp y prueba el valor de retorno de esta función. Si el resultado es falso, muestra una ventana de mensaje [Message Box] diciéndole al usuario que la aplicación no fue encontrada.

```
Private Sub cmdOtraAplicacion_Click ( )
Dim strNombreAplicacion As String
strNombreAplicacion = InputBox ("Para ejecutar la Calculadora, teclea 'calc'. " & _
                                "Para ejecutar el Block de Notas, teclea 'notepad'. ")
Shell strNombreAplicacion, vbNormalFocus
'Verifica la propiedad Number para ver si había un error
If EjecutaApp (strNombreAplicacion) = False Then
        MsgBox "Imposible encontrar " & strNombreAplicación & _
                vbCrLf & "Número de error: " & Err.Number
End If
End Sub
```

2.- En el formulario principal crea una función a nivel formulario llamada EjecutaApp. Asegúrate de hacer lo siguiente:
   a.- Manejar errores al momento de ejecución, usando un manejador de error en línea.
   b.- Regresa un valor verdadero (**True**) si la declaración **shell** fue exitosa, y regresa falso (**False**) si un error ocurrió.

Tú función debe parecerse a lo siguiente:

```
Private Function EjecutaApp(App As String) As Boolean
  On Error Resume Next
  Shell App, vbNormalFocus
  If Err.Number <> 0 Then
    EjecutaApp = False
  Else
    EjecutaApp = True
  End If
End Function
```

3.- Prueba el procedimiento.

PRÁCTICA 10: AGREGAR MENUS

En esta práctica agregarás funcionalidad al proyecto Prestamo iniciado en la práctica 2. Agregarás un menú, una barra de status y una barra de herramientas al formulario principal. Puedes continuar trabajando con los archivos que has creado, o puedes usar los archivos que son proveídos en (Carpeta de Instalación)\Prácticas\Practica10.

Objetivos

Después de completar esta práctica serás capaz de:

- Agregar un menú a un formulario.
- Agregar una barra de status a un formulario.
- Agregar una barra de herramientas a un formulario

Ejercicios

Ejercicio 1: Agregar un menú

En este ejercicio borrarás los botones de comando del formulario principal (frmPrincipal) y llamarás las funciones desde los elementos del menú en lugar de llamarlos desde los botones de comando.

Ejercicio 2: Agregar un control [StatusBar]

En este ejercicio agregarás un control StatusBar a frmPrincipal en el cual enviaras mensajes sobre la barra de status en lugar de usar ventanas de mensaje [Message box].

Ejercicio 3: Agregar un control [ToolBar]

En este ejercicio agregarás un control ToolBar con dos botones a frmPrincipal.

Ejercicio 1: Agregar un menú

En este ejercicio borrarás los botones de comando del formulario principal (frmPrincipal) y llamarás las funciones desde los elementos del menú en lugar de llamarlos desde los botones de comando.

Al igual que los botones de comando, un menú provee al usuario con una manera de invocar acciones en una aplicación.

I.- Abre el proyecto prestamo

1.- Abre el proyecto prestamo con el que has trabajado o abre el proyecto prestamo que se encuentra en (Carpeta de Instalación)\Prácticas\Practica10.

II.- Crea un menú en frmPrincipal

1.- Con frmPrincipal abierto, sobre el menú **Herramientas**, da clic en **Editor de menús**.
2.- Agrega los siguientes elementos al menú:

Caption	Name	Shortcut
&Archivo	mnuArchivo	Ninguno
....&Salir	mnuSalir	Ninguno
Pr&estamo	mnuPrestamo	Ninguno
....Pago &Mensual	mnuPrestamoMensual	CTRL + M
....&Total a Pagar	mnuPrestamoTotal	CTRL + T

3.- Después de que agregues los elementos del menú, da clic en **Aceptar**.

III.- Crea un nuevo sub procedimiento

1.- Abre la ventana del editor de código de frmPrincipal.
2.- Inserta dos nuevos sub procedimientos privados llamados: MuestraMensualmente y MuestraTotal.
Tus Sub procedimientos deben parecerse a lo siguiente:

```
Private Sub MuestraMensualmente( )

End Sub

Private Sub MuestraTotal( )

End Sub
```

3.- Copia el código del procedimiento de evento **cmdMensual_Click** al nuevo Sub procedimiento MuestraMensualmente.
4.- Copia el código del procedimiento de evento **cmdTotal_Click** al nuevo Sub procedimiento MuestraTotal.

IV.- Agrega código al evento click de los elementos del menú.

1.- Agrega código al procedimiento de evento **Click** del elemento del menú **Salir**, **mnuSalir**, de tal manera que descargue frmPrincipal.
2.- Agrega código al procedimiento de evento **Click** del elemento del menú **Pago Mensual**, **mnuPrestamoMensual**, de tal manera que llame al nuevo procedimiento **MuestraMensualmente**.
3.- Agrega código al procedimiento de evento **Click** del elemento del menú **Total a Pagar**, **mnuPrestamoTotal**, de tal manera que llame al nuevo procedimiento **MuestraTotal**.

V.- Guarda y prueba tu aplicación

1.- Selecciona cada elemento del menú
2.- Prueba las teclas de acceso. Por ejemplo ALT + A.
2.- Prueba las combinaciones de teclas. Por ejemplo las teclas CTRL + M.

VI.- Borra los botones de comando

Ahora que ya hay elementos del menú que invocan a las funciones que anteriormente eran invocadas desde los botones de comando, puedes borrar los botones de comando y sus procedimientos de evento **Click**.

1.- Borra los botones de comando **Pago Mensual**, **Total a Pagar** y **Cerrar** de frmPrincipal.
2.- Borra los procedimientos de evento Click, **cmdMensual_Click**, **cmdTotal_Click** y **cmdSalir_Click**.

Ejercicio 2: Agregar un control [StatusBar]

En este ejercicio, agregarás un control StatusBar a frmPrincipal en el cuál, enviaras mensajes sobre la barra de status en lugar de usar ventanas de mensaje [Message box].

I.- Agregar más controles Active X al proyecto usando Microsoft Windows Common Controls

1.- Sobre el menú proyecto, da clic en Componentes.

2.- De la lista de controles disponibles, selecciona Microsoft Windows Common Controls 6.0, y entonces da clic en Aceptar.

Observa que nuevos controles son agregados a la caja de herramientas de Visual Basic.

II.- Crea una barra de status

1.- De la barra de herramientas, agrega un control **StatusBar** en la parte inferior de frmPrincipal.
2.- Establece la propiedad **Nombre** de la barra de status a **sbrPrestamo**.
3.- Abre la caja de diálogo **Páginas de propiedades** del control **StatusBar**, seleccionando la propiedad **(Personalizado)** de la ventana de propiedades.
4.- Selecciona la etiqueta **Paneles**, y da clic al botón **Insertar panel** para insertar un segundo panel.
5.- Establece las propiedades para los dos paneles como sigue:

Paneles Index	Propiedad	Configuración deseada
1	Key	msg
	Style	0 – sbrText
	Autosize	1 – sbrSpring
2	Key	hora
	Alignment	1 – sbrCenter
	Style	5 – sbrTime
	Autosize	2 – sbrContents

6.- Da clic en **Aplicar**.
7.- Da clic en **Aceptar**.

III.- Envía mensajes a la barra de estado

1.- En el procedimiento **MuestraMensualmente**:
a.- Reemplaza el mensaje de error con un mensaje en la barra de estado:

```
sbrPrestamo.Panels("msg").Text = "Se necesita un valor numérico"
```

b.- Reemplaza el mensaje mostrando los resultados de la función **PagoMensual** con un mensaje en la barra de estado:

```
sbrPrestamo.Panels("msg").Text = "Pago mensual: " & _
                                 Format (dblMensual, "currency")
```

2.- En el procedimiento MuestraTotal, reemplaza cada uno de los mensajes, con un mensaje en la barra de estado.
3.- Guarda y prueba tu aplicación.

Ejercicio 3: Agregar un control [ToolBar]

En este ejercicio, agregarás un control ToolBar con dos botones a frmPrincipal.

I.- Agrega una lista de imágenes a frmPrincipal

1.- De la **Caja de herramientas**, agrega un control **ImageList** a frmPrincipal.
Nota: El control **ImageList** es invisible cuando la aplicación se ejecuta, así es que no te preocupes del lugar dónde lo coloques.
2.- En la ventana de propiedades establece su propiedad **Nombre** a **imlPrestamo**.
3.- Abre la caja de diálogo **Páginas de propiedades** del control **ImageList**, seleccionando la propiedad **(Personalizado)** de la ventana de propiedades.
4.- Da clic en el botón de opción de **16 x 16** sobre la etiqueta **General**.
5.- Desplázate a la etiqueta de **Imágenes** e inserta 2 imágenes, dando clic en el botón **Insertar imagen**.
Las imágenes se encuentran en la siguiente ruta: (Carpeta de Instalación)\Prácticas \Practica10\exit.bmp y en (Carpeta de Instalación)\Prácticas\Practica10\month.bmp
6.-Después de insertar las imágenes, da clic en **Aceptar**.

II.- Agrega un control ToolBar con 2 botones a frmPrincipal

1.- De la Caja de herramientas, agrega un control **ToolBar** en la parte superior de frmPrincipal.
2.- En la ventana de propiedades del control, establece la propiedad **Nombre** a **tlbPrestamo**.
3.- Abre la caja de diálogo **Páginas de propiedades** del control **ToolBar**, seleccionando la propiedad **(Personalizado)** de la ventana de propiedades.
4.- En la etiqueta **General**:
a.- Establece la propiedad ImageList a la lista de imágenes que acabas de crear. **imlPrestamo**.
b.- Establece la propiedad **Style** a **1 – tbrFlat**.
5.- En la etiqueta **Botones**, inserta 2 botones con las siguientes propiedades:

Botones Index	Propiedad	Configuración deseada
1	Description Key ToolTip Text Image	Botòn Salir salir Salir 1
2	Description Key ToolTip Text Image	Botón Pago Mensual mes Pago Mensual 2

6.- Después de insertar los botones da clic en Aceptar.

III.- Responder al evento ButtonClick del control ToolBar

Todos los botones sobre el control ToolBar comparten el mismo procedimiento de evento ButtonClick. Además, debes determinar que botón invocó el evento y llamar la función apropiada.

1.- Abre la ventana del editor de código para el evento **ButtonClick** del control **tlbPrestamo**.

2.- Agrega el código que determine que botón fue seleccionado y ejecuta el procedimiento apropiado:

```
Select Case Button.Key
        Case "mes"
                MuestraMensualmente
        Case "salir"
                Unload frmPrincipal
End Select
```

3.- Guarda y prueba tu aplicación.

PRÁCTICA 11: USAR EL ASISTENTE PARA EMPAQUETAMIENTO Y DISTRIBUCION

Cuando has finalizado una aplicación, tu desearías empaquetar esta para instalarla en cualquier computadora que trabaje son el sistema operativo Windows. El asistente para empaquetado y distribución automatiza el proceso de comprimir y copiar archivos hacia discos flexibles o hacia un fólder para su distribución. En esta práctica crearás un programa de instalación [Setup] para la aplicación que estima el pago de un préstamo.

Puedes continuar trabajando con los archivos que has creado, o puedes usar los archivos que son proveídos en (Carpeta de Instalación)\Prácticas\Practica11.

Objetivos

Después de completar está práctica serás capaz de:
- Usar el asistente para empaquetamiento y distribución.
- Instalar tu aplicación.
- Desinstalar tu aplicación

Ejercicios

Ejercicio 1: Crear un programa de instalación [Setup]
En este ejercicio, usarás el asistente para empaquetamiento y distribución, para crear un programa de Instalación [Setup] para distribuir tu aplicación.

Ejercicio 2: Instalar una aplicación
En este ejercicio, ejecutarás el programa de Instalación [Setup] para la aplicación que creaste en el ejercicio anterior.

Ejercicio 3: Desinstalar una aplicación
En este ejercicio, desinstalarás la aplicación que instalaste en el ejercicio anterior.

Ejercicio 1: Crear un programa de instalación [Setup]

En este ejercicio, usarás el asistente para empaquetamiento y distribución, para crear un programa de Instalación [Setup] para distribuir tu aplicación.

I.- Abre el proyecto Prestamo

1.- Si Visual Basic no se ejecuta, inicialo desde el menú **Inicio**.
2.- Abre el proyecto con el que has trabajado, o puedes usar los archivos que son proveídos en (Carpeta de Instalación)\Practicas\Practica11.

II.- Agrega los toques finales a la aplicación.

1.- Establece la propiedad **Icon** de frmPrincipal al icono FLGMEX localizado en (Carpeta de Instalación) \Prácticas\Practica11.
2.- Sobre el menú **Proyecto**, da clic en **Componentes** y elimina los controles que no son usados en el proyecto.

Nota: Visual Basic no te permitirá eliminar un control o referencia que este siendo usado por el proyecto.

.- Sobre el menú **Proyecto**, da clic en **Referencias**, y elimina todas las referencias que no son usadas en el proyecto.
4.- Guarda todos los cambios hechos al proyecto.
5.- Define el título de la aplicación y crea un archivo ejecutable.

Nota: El título de la aplicación es por omisión el texto desplegado en la barra de título de las ventanas de mensaje [MsgBox] y la ventana de entrada [InputBox], cuando no se especifica el parámetro título de estas ventanas.

a.- Sobre el menú **Archivo**, da clic en **Generar Prestamo.exe**.
b.- Da clic en el botón Opciones.
c.- Establece el **Título de la Aplicación** a **Pago estimado de un Préstamo**.
d.- Selecciona **frmPrincipal** como el icono que va ser mostrado, y da clic en **Aceptar**.
e.- Da clic en **Aceptar** para crear el archivo .exe.
6.- Guarda y cierra visual Basic. Prueba tu aplicación dando doble clic al archivo .exe que acabas de crear.

III.- Crea un programa de instalación

1.- Sobre el menú **Inicio**, selecciona **Programas**, selecciona **Microsoft Visual Studio 6.0**, selecciona **Herramientas de Microsoft Visual Studio 6.0** y da clic en **Asistente para empaquetado y distribución**.

2.- En el Asistente para empaquetado y distribución, da clic al botón **Examinar** y Abre el proyecto **Prestamo** con el que has estado trabajando o usa el que es proporcionado en (Carpeta de Instalación) \Prácticas\Practica11 y da clic en el botón **Empaquetar**.

Nota: Ya que tu guardaste los archivos del proyecto después de compilar el archivo ejecutable, el Asistente para empaquetado y distribución detectará una diferencia y te preguntará que si deseas recompilar. Da clic en el botón No para usar el archivo ejecutable creado en los pasos anteriores.

3.- Sobre la pantalla de **Tipo de empaquetado**, selecciona **Paquete de instalación estándar** en el Tipo de empaquetado y da clic en **Siguiente**.
4.- Sobre la pantalla **Carpeta de paquete**, selecciona (Carpeta de Instalación)\Prácticas \Practica11, da clic en el botón **Nueva Carpeta** y escribe **Instalación**, para la localización de los archivos de distribución, y da clic en **Siguiente**.
5.- Sobre la pantalla **Archivos incluidos**, da clic en **Siguiente**.
6.- Sobre la pantalla **Opciones de .cab**, selecciona **Un único archivo .cab** en **Opciones de .cab**, y da clic en **Siguiente**.
7.- Sobre la pantalla **Título de instalación**, deja el título como esta, y da clic en **Siguiente**.
8.- Sobre la pantalla **Elementos del menú Inicio**, observa la localización del menú por omisión, y da clic en **Siguiente**.
9.- Sobre la pantalla **Ubicaciones de Instalación** observa que puedes modificar la ubicación de la instalación para cada uno de los archivos que son listados. Sin hacer cambios, da clic en **Siguiente**.
10.- Sobre la pantalla **Archivos compartidos** da clic en **Siguiente**.
11.- Sobre la pantalla **Finalizado** nombra la secuencia de comandos como **Paquete de Instalación Prestamo 1**, y da clic en **Finalizar**.

Nota: El asistente de instalación, crea el programa [Setup] para tu aplicación. Da clic al botón **Cerrar** en la caja de diálogo **Informe de empaquetado**.

12.- Cuando el asistente para empaquetado y distribución sea finalizado, da clic en **Cerrar**.

Ejercicio 2: Instalar una aplicación

En este ejercicio, ejecutarás el programa de instalación [Setup] para la aplicación que creaste en el ejercicio anterior.

I.- Ejecuta el Programa de instalación (Setup)

1.- Inicia el Explorador de Windows.
2.- Abre la carpeta instalación ((Carpeta de Instalación)\Prácticas\Practica11\Instalación)
3.- Ejecuta el programa [Setup] dando doble clic sobre este, y sigue las instrucciones.

II.- Ejecuta tu aplicación

1.- Después de que la instalación termine ejecuta tu aplicación del grupo de programas del menú Inicio.

Ejercicio 3: Desinstalar una aplicación

En este ejercicio desinstalarás la aplicación que instalaste en el ejercicio anterior.

I.- Desinstala la aplicación

1.- Sobre el menú Inicio, localiza y da clic en el Panel de Control.
2.- Ejecuta el programa Agregar o quitar programas.
3.- Selecciona la aplicación que instalaste en el ejercicio anterior y da clic en el botón Cambiar o quitar
4.- Si te pregunta desinstalar archivos compartidos, selecciona no desinstalarlos.

II.- Intenta ejecutar tu aplicación

1.- Verifica el grupo de programas sobre el menú Inicio. ¿Tu aplicación ha sido desinstalada?

CONCLUSIONES.

Al finalizar la aplicación, el alumno tendrá la capacidad básica para inicializar proyectos, diseñar formularios, establecer las propiedades básicas de los controles ActiveX, agregar más controles ActiveX a la caja de herramientas, codificar procedimientos de eventos, procedimientos, funciones, compilar, crear archivos ejecutables, generar un programa de instalación de la aplicación, además de crear su primer aplicación usando una base de datos.

Dentro de todo esto, la parte más importante es que el alumno va a poder adquirir un poco más de capacidad en la parte lógica, mediante las restricciones para poder hacer uso de la aplicación, el manejo de los errores, la verificación de que los controles tengan un valor inicial para poder funcionar la aplicación y restringir los valores de los controles a sólo numéricos.

Actualmente, la mayoría del desarrollo de software a nivel empresarial esta enfocado en la programación orientada a objetos, por lo que, el contenido del presente material es un buen inicio en la introducción de los conceptos esenciales de la descripción de los objetos (propiedades), las acciones que realizan (métodos o funciones) y el momento en que se ejecutan (eventos).

Tomando en cuenta lo anterior, el alumno podría continuar desarrollando aplicaciones para seguir practicando con los conocimientos adquiridos, además de que va a tener las bases para aprender algún otro lenguaje de programación estructurada tal como Pascal y C++, los cuales son fundamentales para el entendimiento de la programación.

GLOSARIO.

Elementos del entorno integrado de desarrollo

El entorno integrado de desarrollo de Visual Basic (IDE) consta de los siguientes elementos.

Barra de menús

Presenta los comandos que se usan para trabajar con Visual Basic. Además de los menús estándar **Archivo, Edición, Ver, Ventana** y **Ayuda,** se proporcionan otros menús para tener acceso a funciones específicas de programación como **Proyecto, Formato** o **Depuración.**

Barras de herramientas

Proporcionan un rápido acceso a los comandos usados normalmente en el entorno de programación. Haga clic en un botón de la barra de herramientas para llevar a cabo la acción que representa ese botón. De forma predeterminada, al iniciar Visual Basic se presenta la barra de herramientas Estándar. Es posible activar o desactivar otras barras de herramientas adicionales para modificar, diseñar formularios desde el comando **Barras de herramientas** del menú **Ver.**

Las barras de herramientas se pueden acoplar debajo de la barra de menús o pueden "flotar" si selecciona la barra vertical del borde izquierdo y la arrastra fuera de la barra de menús.

Caja de herramientas

Proporciona un conjunto de herramientas que puede usar durante el diseño para colocar controles en un formulario. Además del diseño del cuadro de herramientas predeterminado, puede crear su propio diseño personalizado si selecciona **Agregar ficha** en el menú contextual y agrega controles a la ficha resultante.

Ventana Explorador de proyectos

Enumera los formularios y módulos del proyecto actual. Un *proyecto* es la colección de archivos que usa para generar una aplicación.

Ventana Propiedades

Enumera los valores de las propiedades del control o formulario seleccionado. Una *propiedad* es una característica de un objeto, como su tamaño, título o color.

Ventana Editor de código

Funciona como un editor para escribir el código de la aplicación. Se crea una ventana editor de código diferente para cada formulario o módulo del código de la aplicación.

Nota También puede agregar características a la interfaz de Visual Basic mediante un programa llamado *complemento*. Los complementos, disponibles en Microsoft y otros desarrolladores, pueden proporcionar características como el control de código fuente, que permite mantener proyectos de desarrollo en grupo.

Control ActiveX

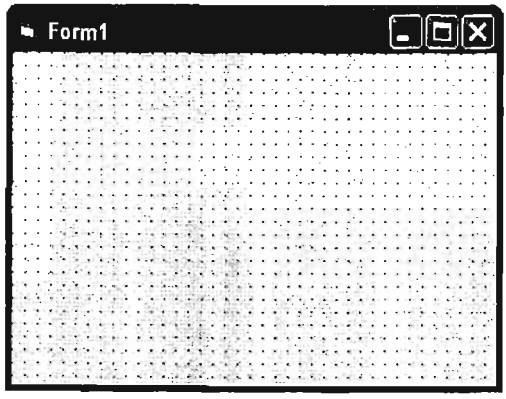
Un control ActiveX, como un control incorporado, es un objeto que se coloca en un formulario para permitir o mejorar la interacción de un usuario con una aplicación. Los controles ActiveX tienen eventos que se pueden incorporar en otros controles. Estos controles tienen la extensión .ocx.

Evento

Una acción, reconocida por un objeto, para la cual puede escribir código de respuesta. Los eventos pueden estar generados por una acción del usuario, como hacer clic con el *mouse* o

presionar una tecla, por código de programa o por el sistema, como ocurre con los cronómetros.

Formulario (Ventana)



Le permite crear ventanas, cuadros de diálogo y controles en la aplicación. Los controles se dibujan y se ven en un formulario.

Mientras está diseñando un formulario:

- Las ventanas Formulario tienen los botones **Maximizar**, **Minimizar** y **Cerrar**.
- Puede crear formularios fijos o móviles. El formulario que diseña tendrá las mismas características en tiempo de diseño y en tiempo de ejecución, a menos que especifique lo contrario en las propiedades del formulario.
- Utilice los botones del Cuadro de herramientas para dibujar los controles en el formulario.

Procedimientos de evento

Cuando un objeto en Visual Basic reconoce que se ha producido un evento, llama automáticamente al procedimiento de evento utilizando el nombre correspondiente al evento. Como el nombre establece una asociación entre el objeto y el código, se dice que los procedimientos de evento están adjuntos a formularios y controles.

- Un procedimiento de evento de un control combina el nombre real del control (especificado en la propiedad **Nombre**), un carácter de subrayado (_) y el nombre del evento. Por ejemplo, si desea que un botón de comando llamado cmdAceptar llame a un procedimiento de evento cuando se haga clic en él, utilice el procedimiento cmdAceptar_Click.
- Un procedimiento de evento de un formulario combina la palabra "Form", un carácter de subrayado y el nombre del evento. Si desea que un formulario llame a un procedimiento de evento cuando se hace clic en él, utilice el procedimiento Form_Click. (Como los controles, los formularios tienen nombres únicos, pero no se utilizan en los nombres de los procedimientos de evento.)

Todos los procedimientos de evento utilizan la misma sintaxis general.

Sintaxis de un evento de control	**Sintaxis de un evento de formulario**

Private Sub *nombrecontrol_Nombreevento* (*argumentos*)
  instrucciones
End Sub
	Private Sub Form_*nombreevento* (*argumentos*)
	  instrucciones
	End Sub

Aunque puede escribir procedimientos de evento nuevos, es más sencillo usar los procedimientos de código que facilita Visual Basic, que incluyen automáticamente los nombres correctos de procedimiento. Puede seleccionar una plantilla en la ventana Editor de código si selecciona un objeto en el cuadro **Objeto** y selecciona un procedimiento en el cuadro **Procedimiento**.

También es conveniente establecer la propiedad **Nombre** de los controles antes de empezar a escribir los procedimientos de evento para los mismos. Si cambia el nombre de un control tras vincularle un procedimiento, deberá cambiar también el nombre del procedimiento para que coincida con el nuevo nombre del control. De lo contrario, Visual Basic no será capaz de hacer coincidir el control con el procedimiento. Cuando el nombre de un procedimiento no coincide con el nombre de un control, se convierte en un procedimiento general.

Procedimiento

Una serie de instrucciones que funcionan como una unidad; una subrutina.

Shell (Función)

Ejecuta un programa ejecutable y devuelve un tipo **Variant (Double)** que representa la identificación de la tarea del programa si se ha ejecutado con éxito, en caso contrario devuelve cero.

Sintaxis

Shell(*pathname*[,*windowstyle*])

La sintaxis de la función **Shell** tiene estos argumentos con nombre:

Parte	Descripción
pathname	Requerido; **Variant (String)**. Nombre del programa a ejecutar y de cualesquiera argumentos necesarios o modificador de la línea de comandos; puede incluir directorio o carpeta y unidad de disco.
windowstyle	Opcional. **Variant (Integer)** correspondiente al estilo de la ventana en la cual se va a ejecutar el programa. Si se omite ***windowstyle***, el programa se inicia minimizado con enfoque.

El argumento con nombre ***windowstyle*** tiene estos valores:

Constante	Valor	Descripción
vbHide	0	Se oculta Windows y se pasa el foco a la ventana oculta.
vbNormalFocus	1	Windows recupera el foco y vuelve a su posición y tamaño original.
vbMinimizedFocus	2	Windows se muestra como un icono con foco.
vbMaximizedFocus	3	Windows se maximiza con foco.
vbNormalNoFocus	4	Windows vuelve al tamaño y posición más recientes. La ventana activa actual permanece activa.
vbMinimizedNoFocus	6	Windows se muestra como un icono. La ventana activa actual permanece activa.

BIBLIOGRAFIA.

Ceballos, Francisco. Microsoft Visual Basic 6 Curso de Programación. (Madrid. España: Editorial Alfaomega Ra – Ma, Segunda Edición).

Craig, John. 1998. Microsoft Visual Basic 6.0 Developers Workshop. Estados Unidos: Microsoft Press.

Davis, Harold.1999. Microsoft Visual Basic 6. Madrid, España; Anaya Multimedia.

Charte, Francisco. 1988. Programación en ActiveX con Visual Basic. Madrid, España: Anaya Multimedia.

Ramírez, José Felipe. 2001. Aprenda Visual Basic Practicando. México; Pearson Educación.

Brown, Kenyon. 1992. Como usar Visual Basic. México; Megabyte.

Ceballos Francisco. 2000. Enciclopedia de Visual Basic 6. México; Alfaomega.

Ayuda de MSDN Library Visual Studio 6.0. Estados Unidos. Microsoft