MATERIA: PROGRAMACIÓN I

VISUAL BASIC

ESCUELA: ANÁLISIS DE SISTEMAS

ÚLTIMA ACTUALIZACIÓN: ENERO 2019

ELABORADO POR: ING. MARIA NEUS

Prefacio

El presente documento forma parte del programa de estudios de la materia Programación I ofrecido a sus estudiantes en el Instituto Universitario de Tecnología para la Informática – Iutepi. Sirve de apoyo complementario bajo la modalidad de autoaprendizaje publicado en su campus virtual, a todos los alumnos que cursan la materia.

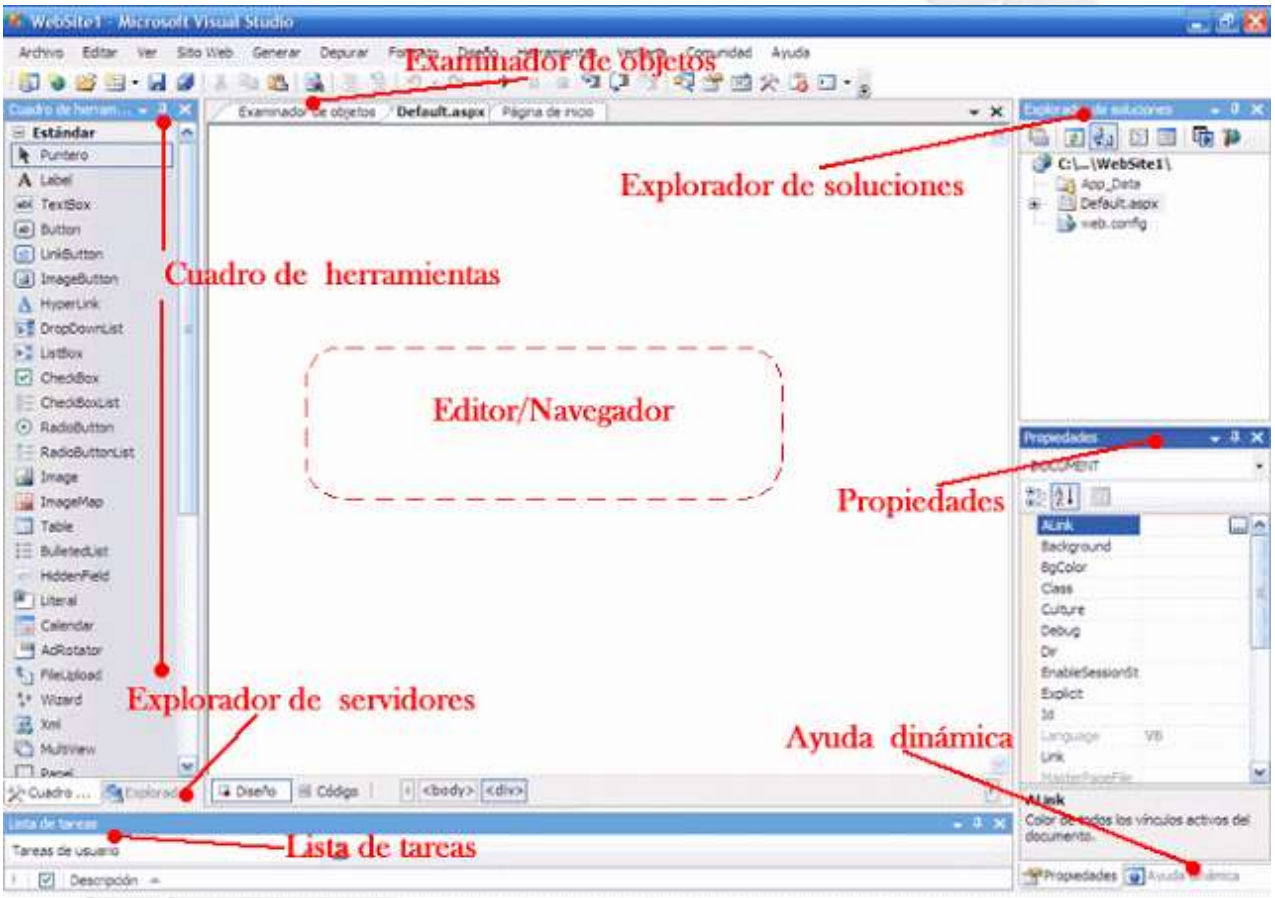
Contenido del programa de estudios

UNIDAD 1. AMBIENTE DE TRABAJO VISUAL BASIC
Área de trabajo
2. PROGRAMACIÓN WINDOWS
- Objetos y Eventos de Programación
3. VARIABLES
- Definición de variables.
- Tipos de variables
- Ámbito y alcance de una variable
4. OPERADORES
- Aritméticos
- Relacionales
- Lógicos
5. FUNCIÓN BÁSICA
- Función INPUTBOX
- Función y procedimiento MSGBOX
6. FORMAS, CONTROLES, PROPIEDADES Y MÉTODOS
- Propiedades
- Métodos.
- Propiedades y notación de punto.
- Métodos y notación de punto.
7. ESTRUCTURA DE DECISIÓN: RAMIFICACIONES Y CICLOS
- Ramificando con IF... END IF
- SELECT CASE... END SELECT.
- Estructura de Repetición
- Uso del SELECT CASE.
- Uso de Do UNTIL.
- Uso de FOR NEXT.
8. EXPLORACIÓN DE CONTROLES DE LA CAJA DE HERRAMIENTAS
- Cajas de textos y Etiquetas.
- Botones de opción.
- Casillas de verificación y marcos
9. FORMULARIO O PROCEDIMIENTO DE ARRANQUE
- Definición de formularios de arranque
- Creación de menús.
- Formulario MDI.
- Creación de Programa MDI.

10. MANEJO DE DATOS CON VISUAL BASIC
- El administrador y el control de datos.
- Data Manager.
- Creación de Tabla de Base de Datos.
- Propiedades de los DataControl
- Consulta de la base de datos con SQL

UNIDAD 1 AMBIENTE DE TRABAJO VISUAL BASIC

Área de trabajo



Los elementos que componen la pantalla de Visual Basic son:

Barra de menús. Visualiza las órdenes que se utilizan para desarrollar, probar y archivar una aplicación. Los menús que nos encontramos son:

Archivo: contiene las órdenes para trabajar con archivos.

Edición: contiene las herramientas que ayudan a escribir el código.

Ver: da acceso rápido a todas las partes del Programa.

Insertar: permite incluir nuevos módulos y formularios en la aplicación.

Ejecutar: permite verificar la aplicación mientras se desarrolla.

Herramientas: controla el aspecto y propiedades del entorno.

Complementos: contiene las utilidades para el manejo de bases de datos.

Ayuda: proporciona una valiosa, cómoda y potente ayuda, muy bien pensada y elaborada, con la cual, el principiante puede aprender muchas cosas y el programador experimentado puede auxiliarse cuando le sea necesario.

Caja de herramientas . Provee de un conjunto de herramientas que permiten colocar los controles en el formulario durante el diseño del proyecto. Ventana de proyecto. En esta ventana están especificados los ficheros (formularios, módulos, etc.) que forman la aplicación y, dónde se seleccionarán para crearlos o modificarlos. Esto se debe a que hay ficheros que pueden utilizarse en más de una aplicación. Además contiene dos botones: Ver Formulario que visualiza el formulario seleccionado y; Ver Código que visualiza el código del fichero seleccionado.

Ventana del formulario . Es la ventana que da lugar a la interfaz de usuario. Es la ventana que se personalizará. Los puntos que aparecen sobre el formulario, forman una rejilla que ayuda a la hora de alinear los controles que se sitúan sobre el mismo. Esta rejilla desaparece en tiempo de ejecución. Para eliminarla en tiempo de diseño se accederá a la opción Herramientas/Opciones/Ficha Entorno/Mostrar Cuadrícula.

Ventana de propiedades. Especifica las propiedades de cada uno de los objetos. En cada momento mostrará las propiedades del objeto seleccionado en el formulario. Está formada por dos partes: la lista desplegable de objetos que visualiza el nombre del objeto seleccionado y, la lista de propiedades del objeto seleccionado.

Los elementos que componen la pantalla de Visual Basic son:

Barra de menús. Visualiza las órdenes que se utilizan para desarrollar, probar y archivar una aplicación. Los menús que nos encontramos son:

Archivo: contiene las órdenes para trabajar con archivos.

Edición: contiene las herramientas que ayudan a escribir el código.

Ver: da acceso rápido a todas las partes del Programa.

Insertar: permite incluir nuevos módulos y formularios en la aplicación.

Ejecutar: permite verificar la aplicación mientras se desarrolla.

Herramientas: controla el aspecto y propiedades del entorno.

Complementos: contiene las utilidades para el manejo de bases de datos.

Ayuda: proporciona una valiosa, cómoda y potente ayuda, muy bien pensada y elaborada, con la cual, el principiante puede aprender muchas cosas y el programador experimentado puede auxiliarse cuando le sea necesario.

Ventana o barra de herramientas. Facilita el uso a las órdenes más comunes. De izquierda a derecha los iconos que aparecen permiten:

Formulario: crear un nuevo Formulario.

Modulo: crear un nuevo módulo. Se utiliza para crear fragmentos de código independiente del formulario.

Abrir Proyecto: abrir un proyecto.

Guardar Proyecto: guardar un proyecto.

Bloquear Controles: impedir que se muevan los controles del formulario involuntariamente.

Editor de Menús: visualizar la ventana de diseño de menús.

Propiedades: visualizar la ventana de propiedades de los distintos objetos.

Examinador de Objetos: mostrar las clases, métodos, propiedades, etc. de los objetos disponibles en la aplicación.

Proyecto: visualizar la Ventana de proyecto.

Inicio: ejecutar la aplicación diseñada y así poder probar su funcionamiento.

Interrumpir: realizar una pausa en la ejecución de la aplicación bajo prueba, que puede reiniciarse cuando se desee con el botón continuar.

Terminar: terminar la ejecución de la aplicación bajo prueba para volver a la etapa de diseño.

Alternar Puntos de Ruptura:

Inspección Instantánea: visualizar el valor del elemento seleccionado en la ventana de código.

Llamadas: visualizar la estructura de llamadas activas.

Caja de herramientas. Provee de un conjunto de herramientas que permiten colocar los controles en el formulario durante el diseño del proyecto. Ventana de proyecto. En esta ventana están especificados los ficheros (formularios, módulos, etc.) que forman la aplicación y, dónde se seleccionarán para crearlos o modificarlos. Esto se debe a que hay ficheros que pueden utilizarse en más de una aplicación. Además contiene dos botones: Ver Formulario que visualiza el formulario seleccionado y; Ver Código que visualiza el código del fichero seleccionado.
Ventana del formulario. Es la ventana que da lugar a la interfaz de usuario. Es la ventana que se personalizará. Los puntos que aparecen sobre el formulario, forman una rejilla que ayuda a la hora de alinear los controles que se sitúan sobre el mismo. Esta rejilla desaparece en tiempo de ejecución. Para eliminarla en tiempo de diseño se accederá a la opción Herramientas/Opciones/Ficha Entorno/Mostrar Cuadrícula.
Ventana de propiedades. Especifica las propiedades de cada uno de los objetos. En cada momento mostrará las propiedades del objeto seleccionado en el formulario. Está formada por dos partes: la lista desplegable de objetos que visualiza el nombre del objeto seleccionado y, la lista de propiedades del objeto seleccionado.

EVALUACIÓN

Complete las siguientes afirmaciones

1. En _____ están especificados los ficheros
2. El menú que permite verificar la aplicación mientras se desarrolla se llama _____
3. La _____ facilita el uso a las órdenes más comunes
4. La opción para impedir que se muevan los controles del formulario involuntariamente se conoce como_____
5. Las opciones que se encuentran en la barra de menú son:_____

UNIDAD 2. PROGRAMACIÓN WINDOWS

Objetos y Eventos de Programación

En Microsoft Visual Basic, un evento es un mensaje enviado por un objeto dentro de un programa al bucle principal del mismo, informándole que algo ha sucedido. Ese "algo" puede variar ampliamente, desde un reloj corriendo a un clic del ratón en la parte del usuario. El programa puede capturar este evento y utilizar la información dentro de este para tomar decisiones sobre otras operaciones.

Visual Basic viene precargado con una gran cantidad de eventos pre-codificados, pero los usuarios también pueden escribir por su propia cuenta. Estos eventos abarcan casi todos los aspectos de la operación del programa y la interacción. Algunos eventos relacionados con la interacción entre el cursor del ratón y un control en un formulario (por ejemplo un botón) incluyen MouseClick, MouseDoubleClick, MouseEnter, MouseLeave, MouseWheel, MouseMove y MouseHover. Los eventos por lo general reciben nombres muy intuitivos para hacer la lectura del código de Visual Basic más fácil.

Los nuevos eventos se crean mediante un comando de declaración. El código para esto generalmente se ve algo como lo siguiente:

Public Event OptionChanged(ByVal Name As String, ByVal Number As Integer)

Este evento ejemplo tendría que ser manualmente programado para ser elevado al cambiar una opción, y enviaría dos valores al programa: el nombre de la opción cambiada y el valor al que se establece.

Se designa como objeto cualquier elemento, por ejemplo, un formulario, una imagen, un control, tal como una caja de texto; a su vez, los objetos tienen propiedades, que en el caso de la caja de texto una es la propiedad "text" que se encarga de contener el texto que aparecerá en la caja. A los objetos se les puede asociar eventos. Un evento es la ocurrencia de un suceso, comúnmente la acción que realiza el usuario sobre el objeto, que como resultado puede, por ejemplo, provocar un cambio en alguna propiedad de un objeto. Por ejemplo: Visual Basic tiene un evento llamado KeyPress, que ocurre cuando el usuario presiona una tecla; ese evento se puede asociar a la caja de texto, y en él definirá (por programación) qué acción se tomará cuando se oprima una tecla.

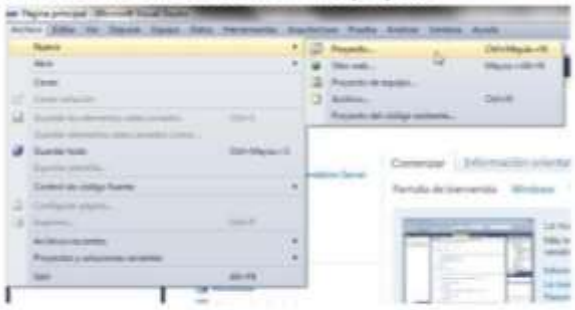
En síntesis, un objeto posee propiedades, responde a eventos y puede ejecutar métodos asociados a él.
Algunos eventos comunes definidos en Visual Basic son:
Click: ocurre cuando se presiona y suelta un botón del mouse sobre un objeto.
DblClick: ocurre cuando se presiona y suelta dos veces un botón del mouse sobre un objeto.
DragDrop: ocurre al arrastrar y soltar un determinado objeto con el mouse.
DragOver: ocurre si una operación de arrastrar y soltar está en curso.
GotFocus: ocurre cuando un objeto recibe el control o foco, ya sea mediante una acción del usuario como hacer click en un objeto ventana, o cambiando el foco de objeto desde el programa, mediante el método SetFocus.
LostFocus: contrario al anterior, este evento ocurre cuando el objeto pierde el enfoque, sea mediante acción del usuario o efectuado desde la aplicación.
KeyDown: ocurre cuando el usuario mantiene presionada una tecla.
KeyUp: ocurre cuando el usuario deja de presionar una tecla. Este evento sucede precisamente al terminar el evento KeyDown.
KeyPress: ocurre como cuando se presiona y suelta una tecla.

MouseDown: ocurre cuando el usuario presiona un botón del mouse.
MouseUp: se produce cuando el usuario suelta el botón del mouse.
MouseMove: este evento ocurre mientras el usuario mueve o desplaza el puntero del mouse sobre un objeto.

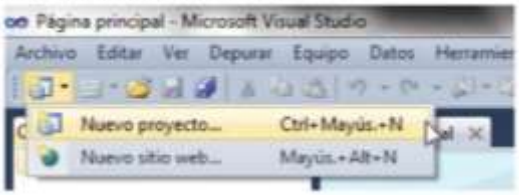
EVALUACIÓN

Práctica 1 (parte 1) Ejercicio Números primos
Abrir nuevo proyecto Visual Studio. NET

Desde el menú **archivo**, en el submenú **nuevo** seleccionamos **proyecto**

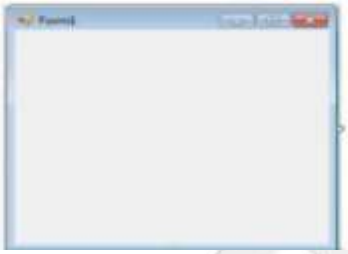


Desde la barra de herramientas Estándar de visual estudio



Creamos una aplicación para Windows Forms

Nos mostrara un formulario en blanco como este:



Los controles que necesitaremos son los siguientes:

Control	Propiedades	Valor
Form	*Text*	*Primo o no primo*
	BackColor	*(al gusto del usuario)*
Label1	*Text*	*Introduzca un numero*
TextBox	*(Name)*	*TxtNumero*
	Text	
Button1	*Text*	*Aceptar*
	(Name)	*BtnOk*
Button2	*Text*	*Borrar*
	(Name)	*BtnErase*
Button3	*Text*	*Salir*
	(Name)	*BtnExit*

NOTA: si no encuentra el cuadro de herramientas vaya al **menú ver → cuadro de herramientas,** si por el contrario no encuentra la ventana de propiedades de clic derecho sobre cualquier control y elija la opción **propiedades,** también vaya al **menú ver →ventana de propiedades.**

El formulario nos quedaría similar al siguiente:

UNIDAD 3. VARIABLES

Definición de variables

Declarar variables significa indicar el tipo de dato que almacenara las variables que participan en el programa.
Antes de usar una variable debemos declararla y es recomendable iniciarla. La primera vez que se usa debe declararse, el resto de veces no se debe hacer, es importante que el valor asignado sea compatible con el tipo de variable.
Por ejemplo, si declaramos una variable de tipo numérico, esta no puede contener una cadena de caracteres.
La sintaxis de una declaración de una variable es: Dim nombre_variable as tipo_variable. Por ejemplo, Dim num pero como hemos dicho antes es recomendable iniciar estas variables con un número genérico como uno o cero o según la ocasión.
Hay una serie de reglas a la hora de escribir el nombre de las variables:
Los nombres de las variables no pueden comenzar por números, pero si puede formar parte del nombre.
El nombre de una variable no puede tener espacios.
El nombre de la variable no puede estar formado por operadores como +, -, !, etc.
No puede haber dos variables que se llamen igual, aunque sean de distintos tipos, en el mismo ámbito

Tipos de variables

Los tipos más comunes de variables son:
Integer, representa un número entero de 32 bits con signo.
Long, representa un número entero de 64 bits con signo.
Single, representa un número de punto flotante de precisión simple.
Double, representa un número de punto flotante de precisión doble.
String, representa un texto.
Boolean, representa un valor booleano.

Ámbito y alcance de una variable

Variables Globales

Una variable global es una que es accesible a través del nivel entero de la película de destello. Usted puede fijarla en un marco, e inmóvil contendrá su contenido en otro marco.
Usted no necesita hacer cualquier cosa especial para crear una variable global. Apenas usarla, como en el ejemplo anterior, automáticamente hace la variable global.
En la mayoría de los lenguajes de programación, las variables globales están disponibles por todas partes. Sin embargo, las películas de destello utilizan un sistema de niveles. Las variables globales en el nivel de la raíz no son accesibles dentro de una película clip.at lo menos no directamente.

Variables Locales

Las variables locales, desemejante de globals, están solamente disponibles en la escritura actual. En el marco siguiente, la variable no existirá. Usted puede crear ciertamente una nueva variable con el mismo nombre, pero el contenido anterior del marco pasado no estará en él.
El punto de variables locales es crear código modular. Si una variable es local, se quita de memoria cuando se acaba la escritura. Si no, si es una variable global, la variable y su valor colgarán alrededor hasta que la película termina.

Por ejemplo, usted podría crear una variable local nombrada myLocal y poner el número 9 en él tenga gusto de esto:
var = 9 myLocal;
Después de que usted fije la variable con la palabra clave del var, usted no tiene que utilizar el var otra vez en ese pedazo local del código. Por ejemplo, el código siguiente crea la variable local, la fija a 9, cambia su valor a 11, y después lo envía a la ventana de la salida:
var = 9 myLocal; = 11 myLocal; trace(myLocal);
Al decidir a cuando utilizar variables locales y a cuando utilizar variables globales, la regla del pulgar es utilizar siempre variables locales a menos que haya una buena razón de utilizar un global. Utilizaremos sobre todo variables locales

EVALUACIÓN

Practica 1 (Parte 2)

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnOk.
    Dim Numero As Integer
    Dim msg As String
    Dim cont As Integer
    Dim x As Integer
    Numero = Me.TxtNumero.Text
    cont = 0
    For x = 1 To Numero
        If Numero Mod x = 0 Then
            cont = cont + 1
        End If
    Next
    If cont <= 2 Then
        msg = "el numero es primo"
    Else
        msg = "el numero no es primo"

    End If
    Me.Label2.Text = msg

End Sub
```

Ahora escribiremos el código del botón borrar:

```
Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    TxtNumero.Text = ""
    Label2.Text = ""
End Sub
```

Ahora codificaremos el botón cerrar

```
Private Sub BtnExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles BtnExit.Click
    Me.Close()
End Sub
```

Al ejecutar la aplicación nos quedara así:

- En el caso de ser primo:

- En el caso de no ser primo:

UNIDAD 4. OPERADORES

Aritméticos

El operador más (+) se utiliza para añadir un valor a otro y también para concatenar cadenas.
El operador menos (-) se usa para hallar la diferencia entre dos o más números y también para mostrar un número como negativo.
El operador multiplicación (*) se utiliza para multiplicar un valor por otro.
El operador exponencial (^) se utiliza para elevar un número a la potencia de otro número.
División
La división se utiliza para obtener la fracción de un número en términos de otra, es decir, para dividir un número entre otro (evidente). En términos de VBA se pueden diferenciar dos tipos de división:
División entera: se utiliza cuando queremos que el resultado sea un número entero. Para ello se utiliza la barra invertida (\). Los operandos (que así es como se llama a los valores que intervienen) pueden ser cualquier tipo de número válido, con o sin decimales.
División decimal: Cuando queremos que el resultado de la operación sea un número decimales, debemos utilizar la barra normal (/).
Para hallar el resto de una división, se utiliza el operador Mod. Este operador divide un valor entre otro y muestra "sobrante" en forma de número decimal

Relacionales

Comparan valores y expresiones devolviendo siempre un resultado booleano: verdadero o falso.
Observa la siguiente tabla en la que te muestro algunos ejemplos de operadores de comparación.

Variable a	Variable b	Expresión	Resultado
27	3	a>b	Verdadero
35	7	a<b	Falso
12	12	a=b	Verdadero
12	12	a<>b	Falso

Lógicos

Muchas veces los operadores de comparación por sí solos se quedan algo cortos para algunas necesidades. Los operadores lógicos solucionan este problema al permitir la combinación de varias expresiones simples para formar una más compleja.
El operador And (y) combina dos condiciones simples y devuelve un valor que es verdadero si estas dos condiciones son verdaderas. Por ejemplo (a >=6) And (b>=7).
El operador Or (o) devuelve el valor Verdadero cuando una de las dos expresiones es verdadera.
El operador Not (no) se utiliza sobre una sola condición para negar su valor.

EVALUACIÓN

Indique si las siguientes líneas de código son correctas o incorrectas

1. If b MOD =0 V__ F__
2. Dim b As Integer = 1 To 10 V__ F__
3. If num1 > num3 & num3>4 then V__ F__
4. Descuento= total *0.06 V__ F__
5. Tp = Tp X 70 V__F__

UNIDAD 5. FUNCIÓN BÁSICA

Función INPUTBOX

Muestra un indicador en un diálogo en el que el usuario puede introducir texto. La entrada se asigna a una variable.

La instrucción InputBox es un método cómodo para introducir texto a través de un diálogo. Confirme la entrada pulsando Aceptar o la tecla Retorno. La entrada se devuelve como valor de retorno de la función. Si se cierra el diálogo con Cancelar, InputBox devuelve una cadena de longitud cero ("").

Función y procedimiento MSGBOX

msgbox: Las cajas de mensajes o Message Box, tienen una función clara, que es la de mostrar una determinada información, aviso, o pregunta para que el usuario tenga conocimiento de ella y actúe.

EVALUACIÓN

Práctica 2. Indicar si un número es negativo, positivo o cero a través de un mensaje

Que necesitaremos para esta aplicación:

Control	Propiedades	Valor
Form	Text	Positivo, negativo o cero
	BackColor	(al gusto del usuario)
Label1	Text	Digite un numero
TextBox	(Name)	TxtNumero
	Text	
Button1	Text	&Aceptar
	(Name)	BtnOk
Button2	Text	&Limpiar
	(Name)	BtnErase
Button3	Text	&Salir
	(Name)	BtnExit

Para la solución de este ejercicio partimos de la premisa que hay números negativos y positivos, los menores que cero son considerados negativos y los mayores que cero positivos.

Se usa la función **IsNumeric** para indicar si lo ingresado en la caja de texto es numero o letra, si lo acompañamos de una condicional **not IsNumeric** evaluamos que no sea numero.

El código en el botón **aceptar** nos quedaría así:

```
Private Sub BtnAceptar_Click(sender As System.Object, e As System.EventArgs) Handles BtnAceptar.Click
    Dim num As Integer
    If TextBox1.Text = "" Then
        MsgBox("esscriba un numero en la casilla", vbInformation)
    ElseIf Not IsNumeric(TextBox1.Text) Then
        MsgBox("esscriba un numero no texto", vbInformation)
        TextBox1.Text = ""
    ElseIf IsNumeric(num) Then
        num = Val(TextBox1.Text)
        If num = 0 Then
            MsgBox("el numero es cero", vbExclamation)
        ElseIf num > 0 Then
            MsgBox("el numero es Positivo", vbExclamation)
        ElseIf num < 0 Then
            MsgBox("el numero es Negativo", vbExclamation)
        End If
    End If
```

El código de borrar y salir nos quedara así:

```
Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles BtnLimpiar.Click
    Me.TextBox1.Text = ""
End Sub

Private Sub BtnSalir_Click(sender As System.Object, e As System.EventArgs) Handles BtnSalir.Click
    Me.Close()
End Sub
```

Al ejecutar la aplicación nos mostrara lo siguiente:

UNIDAD 6. FORMAS, CONTROLES, PROPIEDADES Y MÉTODOS

Propiedades

Las propiedades de las que dispone el control son las siguientes:(para obtener el cuadro de propiedades, seleccionar el control y pulsar F4 o pulsar con el boton derecho para obtener el menú contextual y marcar Propierties)
Text: Aqui indicamos el texto que aparecerá en el control. Podemos asignarle cualquier texto en tiempo de diseño o ejecución. También podemos tomar el texto que haya introducido el usuario para tratarlo durante la ejecución.
Name: Esta propiedad la tienen todos los controles, el nombre que viene por defecto en este caso Text1 y es el nombre con el que se conocerá el control cuando lo utilicemos en el código. En un mismo formulario no puede haber dos controles con el mismo nombre. Conviene poner un nombre que represente la función que tiene el control en la aplicación para que el código quede más claro. Ejemplo, si en el textbox vamos a introducir la dirección de una persona podemos asignarle a esta propiedad el valor Dirección.
MultiLine: Permite que introduzcamos varias líneas de texto en el control en lugar de sólo una.
Alignment: Alineación que tendrá el texto dentro del control: izquierda, centro o derecha. Para que funcione la propiedad MultiLine debe estar con el valor true.
Locked: Si esta con valor true bloquea el control, es decir, el usuario no puede introducir ni modificar el texto que contenga. Nos puede servir para utilizar el control como salida de datos sin que el usuario pueda modificarlos por error.
Otras propiedades que son comunes a la mayoria de los controles:
Backcolor: color de fondo.
Forecolor: color de letra.
Font: tipo y tamaño de letra.

Métodos.

Recordemos que por métodos se entienden los procedimientos o funciones asociados a un control, los cuales nos permiten realizar ciertas operaciones útiles sobre dicho control: Ej. ordenar sus elementos, buscar un dato, etc..

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

Los eventos son acciones que se pueden realizar en cualquier control: click, doble click, movimiento del ratón. A estos eventos se les puede asociar código para que se ejecute al producir el evento.
MouseMove: al mover el raton por encima del control.
Mousedown: al pulsar cualquier boton del raton
Change: al cambiar el contenido del control
Click: al hacer click con el botón izquierdo del ratón sobre el control
Doubleclick: al hacer doble click con el botón izquierdo del ratón sobre el control
Getfocus: este evento se activa cuando el control recibe el enfoque, es decir, cuando se activa el control en tiempo de ejecución para introducir datos en él o realizar alguna operación.

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

EVALUACIÓN

Práctica 3: Realizar una aplicación que permita cambiar los controles y propiedades de un formulario para diseñar una calculadora elemental

UNIDAD 7. ESTRUCTURA DE DECISIÓN: RAMIFICACIONES Y CICLOS

Ramificando con IF... END IF

If...Then...Else construcciones le permiten probar una o más condiciones y ejecutar una o varias instrucciones según cada condición. Puede probar las condiciones y realizar acciones en las siguientes maneras:
Ejecutar una o más instrucciones si una condición es True
Ejecutar una o más instrucciones si una condición es False
Ejecutar algunas instrucciones si una condición es True y otras personas si False
Probar una condición adicional si es un requisito previo False

Select Case... End Select.

Es una instrución que sirve para realizar diferentes cosas dependiendo del valor que tiene una variable. Es un tipo de bifurcación muy potente que nos permite abrir la línea del programa en muchos caminos. Es parecido a un if, pero más potente.

Uso de Do UNTIL.

Visual Basic admite además de la cláusula While, usar el término Until, como equivalente a "hasta que se cumpla que". Así Loop Until i >= 3 significaría "Repetir hasta que i sea mayor o igual que 3". En un bucle en el que i parte de un valor cero y se incrementa unitariamente Do While i < 3 sería equivalente a Do Until i >= 3, y Loop Until i >= 3 sería equivalente a Loop While i < 3. Dado que podemos valernos de equivalencias, puede evitar confundirnos el usar preferentemente un mismo tipo de expresión, sabiendo que disponemos de otra equivalente.

Uso de FOR NEXT.

INSTRUCCIÓN DESDE ... SIGUIENTE (FOR ... NEXT) Y CLÁUSULA PASO (STEP)

La sintaxis a emplear con Visual Basic es la siguiente:

```
For VAR = Vi  To Vf
        Instrucción 1
        Instrucción 2
.
.
.
    Instrucción n
Next VAR
```

El incremento que sufre el contador es, por defecto, unitario. Es decir, el primer valor que toma VAR en el bucle será Vi, el segundo Vi + 1, el tercero (Vi + 1) + 1, etc. La modificación de dicho valor de incremento la realizaremos a través de la cláusula Step después del valor Vf. Step 2 implicará que en cada repetición del bucle el contador se

incremente en dos unidades, Step 5 implicará que en cada repetición del bucle el contador se incremente en cinco unidades. Un paso negativo del tipo Step -1 supone que el contador decrece en vez de incrementarse. Si el paso es negativo, Vi necesariamente habrá de ser mayor que Vf, ya que en caso contrario no se producirá la entrada en el bucle.

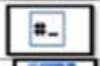
EVALUACIÓN

Práctica 4: Realizar una aplicación que permita realizar la solución para el siguiente caso, haciendo uso de las estructuras Condicionales y cíclicas explicadas anteriormente

1) En una tienda se realiza un descuento del 3% a los clientes que compren más de 10 productos, un descuento de 5% a los que compren entre 12 y 18 productos. Y se le ofrece un ticket para la próxima compra si compra más de 20 productos
2) En un estacionamiento se cobra por hora Bs.400 por minuto adicional Bs.5 determinar el costo que pagará una persona según su tiempo de parqueo

UNIDAD 8. EXPLORACIÓN DE CONTROLES DE LA CAJA DE HERRAMIENTAS

OBJETOS DE LA CAJA DE HERAMIENTAS DE VISUAL BASIC .NET

N°	Nombre	Descripción	Imagen
1.	Puntero	Sirve para seleccionar cualquier objeto.	
2.	Button	Genera un cuento cuando un usuario hace	
3.	Checkbox	Permite al usuario seleccionar o quitar la acción asociado.	
4.	CheckedListbox	Muestra una lista de elementos con una casilla a la izquierda de cada elemento.	
5.	ComboBox	Muestras con cuadros de texto editables con una lista desplegables de valores permitidos.	
6.	DataTimePicker	Permite al usuario seleccionar fecha y fecha, así como mostrar ambas en un formato específico.	
7.	Label	Proporciona información en tiempo de ejecución o texto descriptivo para un control.	
8.	LinkLabel	Muestra un control de etiquetas que admite funcionalidad de hipervínculo, formato y seguimiento	
9.	ListBox	Muestra una lista entre que los usuarios puedan elegir elementos.	
10.	ListView	Muestra una colección de elementos en una de cinco vistas diferentes.	
11.	MaskedTextBox	Utiliza una máscara para distinguir si los datos que especifica el usuario son apropiados o inapropiados.	
12.	MonthCalendar	Muestra un calendario mensual donde el usuario puede seleccionar una fecha.	
13.	NotifyIcon	Muestra un icono en el área de notificación, a la de la barra de tareas de Windows, en tiempo de ejecución.	
14.	NumericUpDown	Muestra un único valor numérico que el usuario pueda aumentar o reducir haciendo clic en los botones de arriba y abajo del control.	

Explicación de la unidad 8 en el siguiente enlace http://notyel-net.blogspot.com/2011/07/textbox-o-caja-de-texto.html

EVALUACIÓN

Práctica 5:

Realizar un formulario partiendo del siguiente código

```
Private Sub cmdIngresar_Click()
If Len(Trim(txtUsuario))=0 Then
txtUsuario.SetFocus
ElseIf Len(Trim(txtContraseña))=0 Then
```

```
txtContraseña.SetFocus
ElseIf txtContraseña = “AGPS” Then
MsgBox “La clave ingresada es correcta”
Unload Me
Else
MsgBox “La clave ingresada no es válida”
```

UNIDAD 9. FORMULARIO O PROCEDIMIENTO DE ARRANQUE

Cada aplicación de Visual Basic debe contener un procedimiento denominado Main. Este procedimiento sirve como punto de partida y control general de la aplicación. Las llamadas de .NET Framework su Main procedimiento cuando se haya cargado la aplicación y está listo para pasar el control a él. A menos que se va a crear una aplicación de formularios Windows Forms, debe escribir el Main procedimiento para las aplicaciones que se ejecutan en sus propios.

Main contiene el código que se ejecuta primero. En Main, puede determinar qué forma se cargarán en primer lugar, cuando se inicia el programa, averiguar si una copia de la aplicación ya se está ejecutando en el sistema, establecer un conjunto de variables de la aplicación o abra una base de datos que requiere la aplicación.

Requisitos para el procedimiento principal

Un archivo que se ejecuta en su propio (normalmente con la extensión .exe) debe contener un Main procedimiento. Una biblioteca (por ejemplo con la extensión .dll) no se ejecuta en su propio y no requiere un Main procedimiento. Los requisitos para los distintos tipos de proyectos que pueden crear son los siguientes:

Aplicaciones de consola ejecutan por sí solas, y debe proporcionar al menos una Main procedimiento. .

Aplicaciones de Windows Forms que se ejecutan por sí solas. Sin embargo, el compilador de Visual Basic genera automáticamente un Main procedimiento de tal aplicación y no es necesario escribir uno.

Bibliotecas de clases no requieren un Main procedimiento. Esto incluye bibliotecas de controles de Windows y bibliotecas de controles Web. Las aplicaciones Web se implementan como bibliotecas de clases.

Declarar el procedimiento Main

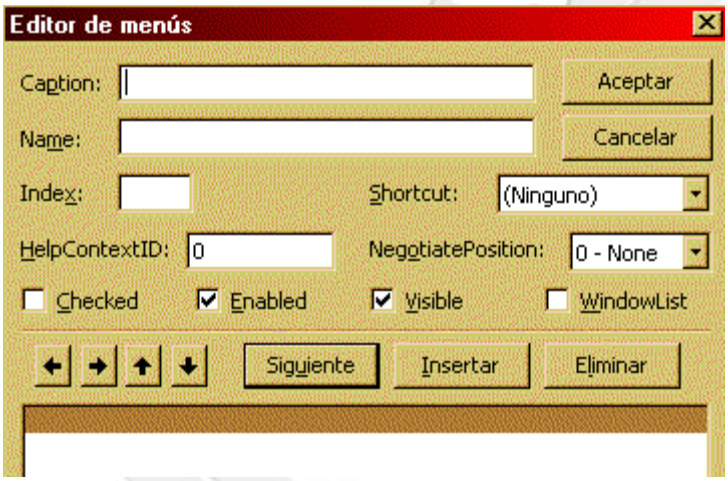
Este Puede tomar o no argumentos y puede devolver un valor o no.

Creación de formularios

Para crear el menú debemos completar los campos que aparecen en el editor:
Caption: Es el título del menú, por ejemplo en Visual Basic, el primer título de la barra de menus es "Archivo".
Name: Es el nombre con el que vamos a identificar el elemento del menú, generalmente se usa el propio caption precesido de mnu.
Index: Está relacionado con la posibilidad de crear arreglos.
HelpContextID: Identifica una ayuda sobre la opción de menú. Si el usuario utiliza F1 podrá utilizar la ayuda en pantalla correspondiente
ShortCut: Para el elemento del menú con el que estemos trabajando podemos elegir una combinación de teclas. En el formulario de ejemplo este se encuentra en (ninguno), pero abriendo Shortcut se puede elegir entre una cantidad de combinaciones.
NegotiatePosition: Selecciona la propiedad NegotiatePosition del menú.
Checked: Los elementos del menú pueden tener a su izquierda un tilde, esta propiedad también se puede activar desde el código.
Enabled: Tiene dos posibilidades True o False. Si no está tildado estará en False y el elemento del menú se verá grisado no pudiendo usarse.

Visible: Al igual que el anterior puede estar en True (cuando está tildado) o en False en cuyo caso no se verá.
WindowsList: Es para aplicaciones con varios formularios o ventanas. Estas pueden manejarse desde el menú. Si WindowsList está tildada (True), el menú puede presentar una lista con las ventanas abiertas
Flecha derecha: Pasa el menú seleccionado a un nivel jerárquico inferior. Crea cuatro niveles de submenús como máximo.
Flecha izquierda: Pasa el menú seleccionado a un nivel jerárquico superior. Crea cuatro niveles de submenús como máximo.
Flecha arriba: Mueve el elemento seleccionado hacia arriba.
Flecha abajo: Mueve el elemento seleccionado hacia abajo.
Lista Menús: Es un cuadro de lista que muestra en orden jerárquico todos los elementos del menú. Los elementos de los submenús aparecen corridos hacia la derecha y con unos puntitos para indicar su jerarquía.(menú, submenú, subsubmenú, subsubsubmenú, según el caso, cada uno un poco más hacia la derecha)

Siguiente: Selecciona la línea siguiente (hacia abajo) y sirve para seguir agregando elementos al menú.
Insertar: Inserta una línea en el cuadro de lista, pero a diferencia de "siguiente" lo hace encima de la línea seleccionada.
Eliminar: Borra la línea seleccionada.
Aceptar: Aplica los cambios efectuados de forma que desaparece el editor de menús y aparece el nuevo menú en el formulario que estamos usando.
Cancelar: Cierra el Editor de menús sin que se efectúe cambio alguno.



Formulario MDI.

Un formulario MDI es un contenedor de formularios. Como si estos fueran pestañas en el explorador. De forma similar los formularios se pueden diseñar para tener mas formularios dentro y manejarlos.

Visual Basic permite crear aplicaciones empleando una interfaz de múltiples documentos (MDI - Multiple Document Interface), o dicho en términos de Visual Basic, múltiples formularios. En una aplicación MDI, se pueden abrir varias ventanas hijas. Cada una de estas ventanas hijas (formularios) son iguales entre sí. En una aplicación MDI pueden

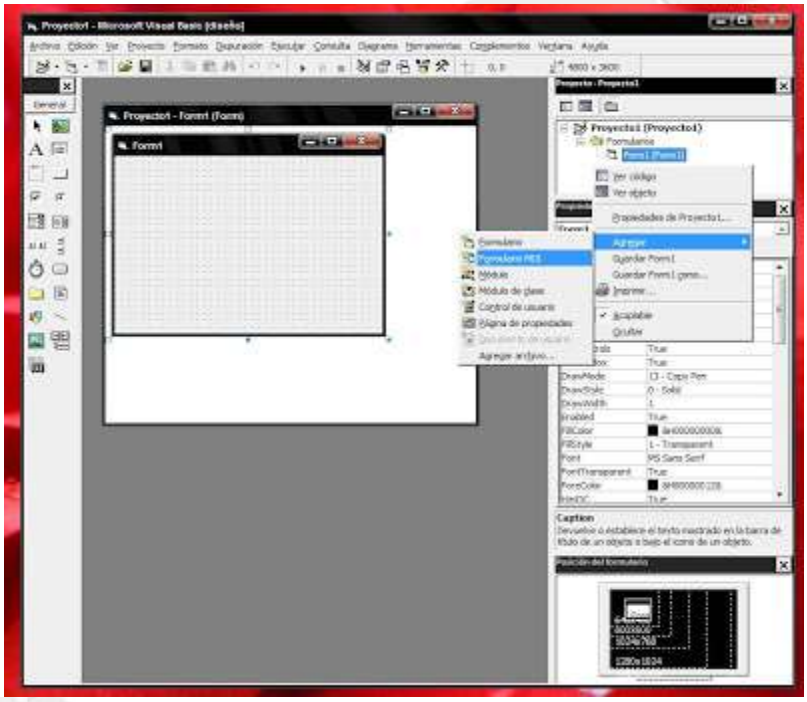
haber varias ventanas hijas, pero solo una ventana padre por aplicación. El formulario padre actúa como contenedor de los formularios hijo. Muchos procesadores de textos bajo Windows son buenos ejemplos de aplicaciones MDI.

Creación de Programa MDI.

Para crear una aplicación MDI se empezará a crear un nuevo proyecto y, se accede a Insert/MDI Form. El nuevo formulario será el formulario padre. Para que un formulario sea un formulario hijo, se deberá cambiar su propiedad MDIChild y establecerla a True.

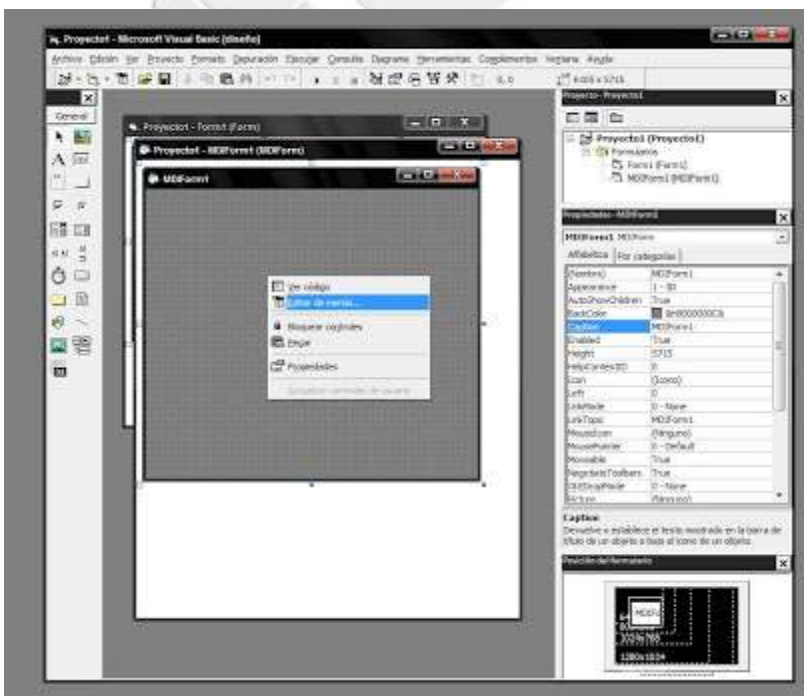
Cuando se visualizan varios formulario hijos, todos comparten el mismo código, pero cada uno de ellos guarda sus propios datos y reconoce sus propios sucesos. Según ésto, no se podrá utilizar el identificador del formulario para referirse a los controles o a sus propiedades, aunque sí se podrá utilizar la palabra clave Me.

1.- Una vez creada una aplicación ejecutable o EXE Estándar, clic derecho en la venta de proyectos... Agregar // Formulario MDI



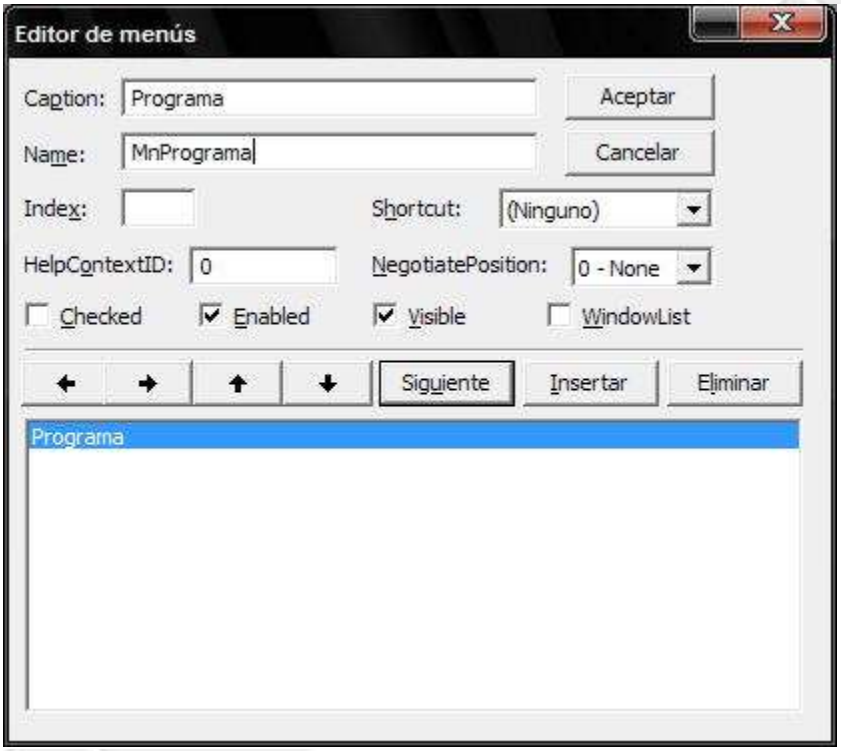
2.- Luego elegir Formulario MDI y clic al botón abrir.

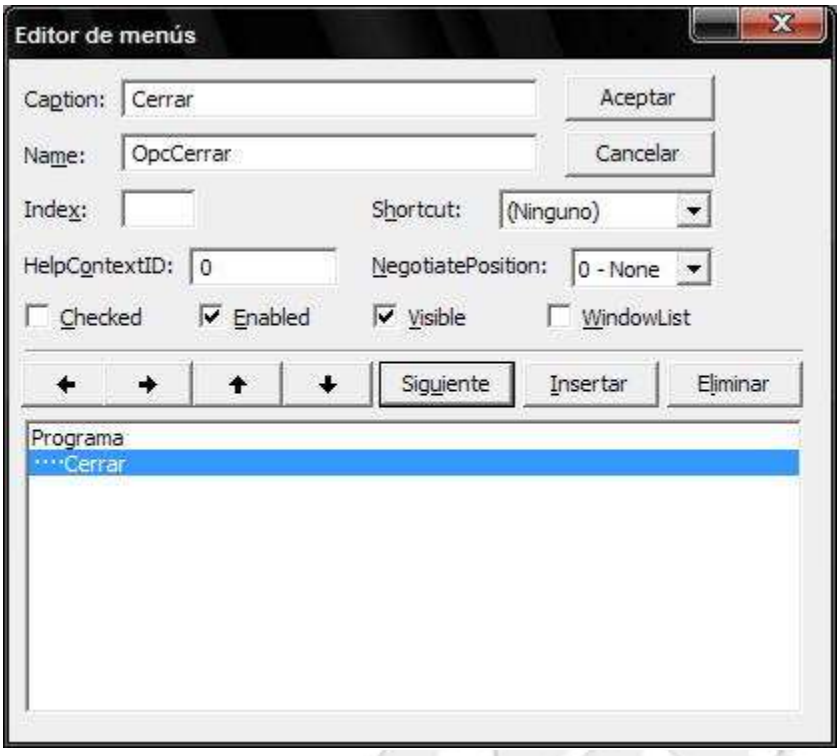
3.- Ya hemos creado el formulario MDI, ahora agreguemos el Menú, para hacerlo clic derecho en el interior del formulario MDI y escoger la opción Editor de menús.

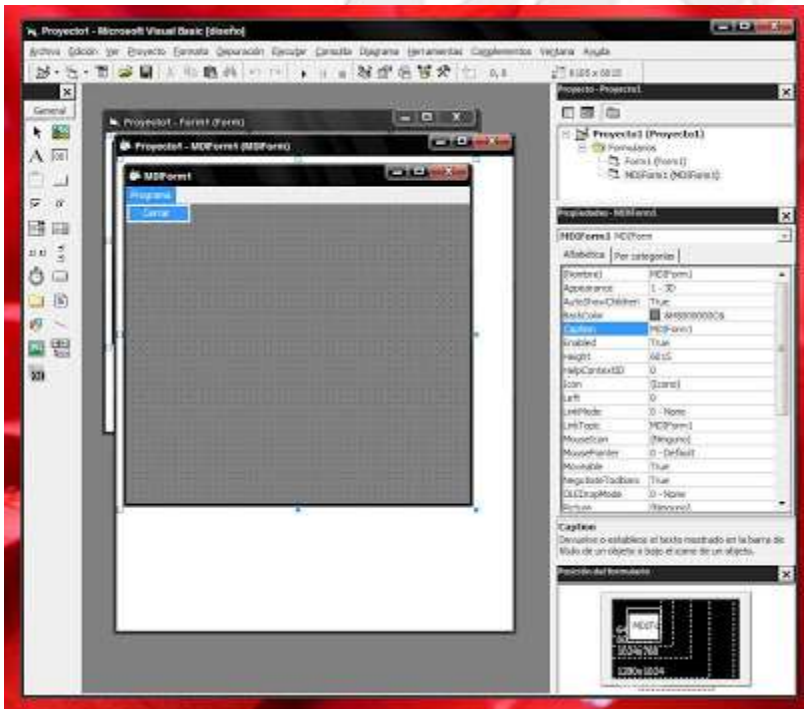


4.- En la ventana que aparecerá, empezar a crear los menú de la siguiente forma... Si deseas ubicar el nombre Raíz

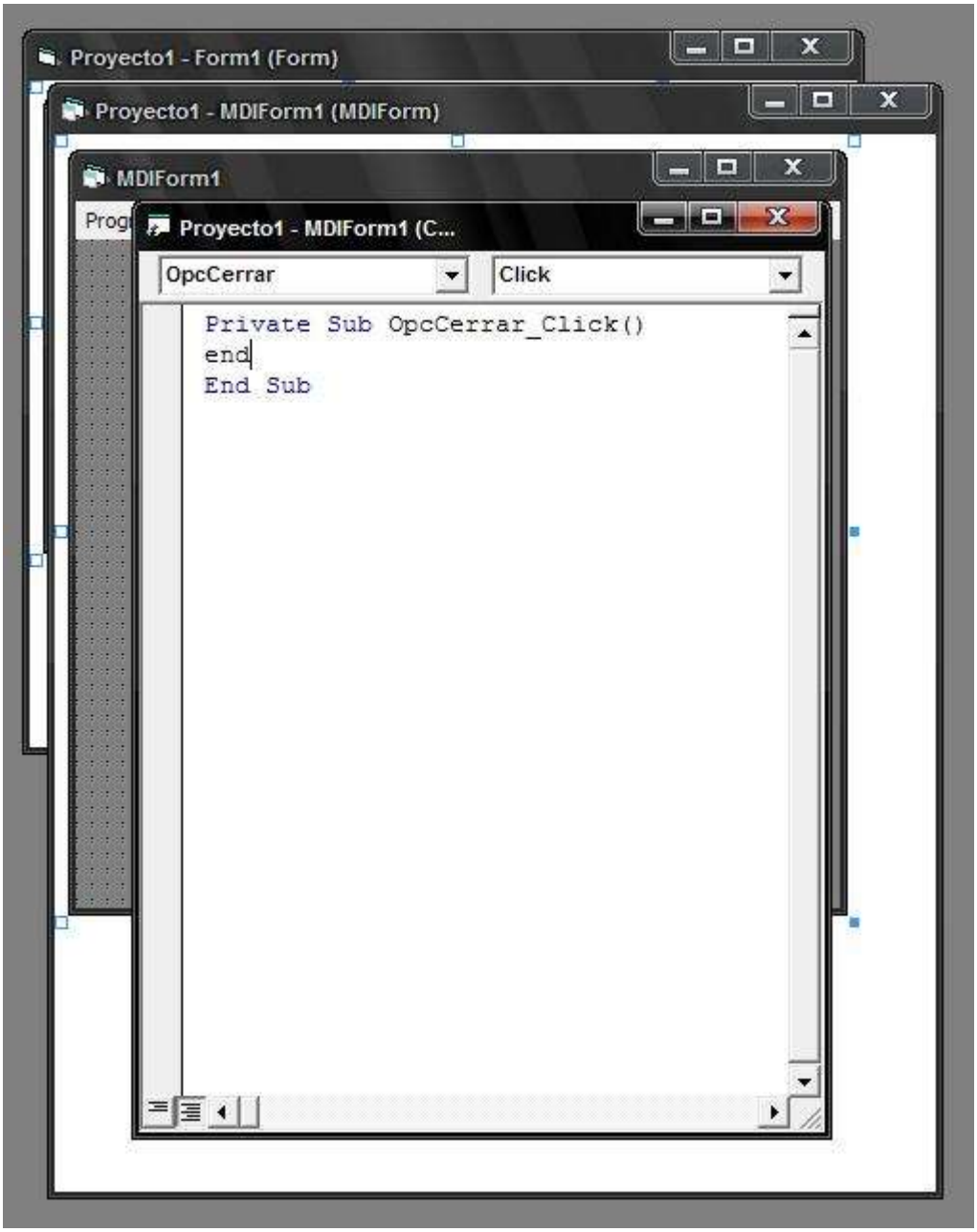
deberás poner en caption el Nombre del grupo, en este caso yo use Programa (por que en este grupo estarán todas las opciones para controlar el programa en forma general, ejemplo Cerrarlo) luego en el campo NAME tienes que poner el nombre del control (este nombre no se mostrara pero servirá para que visual basic pueda reconocer al menú como un elemento independiente y por tal motivo tiene que ser diferente del resto que vayas a poner) En el nombre o NAME tienes que utilizar las siguientes abreviaturas, para un nombre RAIZ o Grupo usas "Mn" (Menu) y para una opción que ira dentro de ese grupo usas la abreviatura "Opc" (Opción) ahora para asignar una opción basta con presionar la Flecha que señala hacia la Derecha y listo veras que es muy sencillo crear menus en VB6.







Para comenzar la codificación de cada Menú basta con hacerle clic encima y aplicar el código respectivo. Como puedes ver en la siguiente imagen, hice clic al Menu programas luego a la opción Cerrar y se abrió la ventana de código. Para cerrar el programa el código es simplemente poner la palabra END.



EVALUACIÓN

Práctica 6:
Cargar un formulario en memoria, pero Use la sentencia **Load**, o haga referencia a una sin mostrarlo propiedad o control sobre el formulario.
Cargar o mostrar el formulario. Use el método **Show**.
Mostrar un formulario cargado. Use el método **Show.**
Ocultar u formulario Use el método **Hide.**
Ocultar un formulario y descargarlo de Use la sentencia **Unload**

UNIDAD 10. MANEJO DE DATOS CON VISUAL BASIC

El administrador y el control de datos.

- Data Manager.

Durante los últimos años hemos visto como los volúmenes de datos han sido incrementados en las organizaciones, hoy en día todos los movimientos que hacemos son registrados por nuestros teléfonos inteligentes, sensores, logs de páginas web, coordenadas de posicionamiento y hábitos de compras. La competencia entre empresas está empezando a enfocarse más en la experiencia de los usuarios que en los productos en si. Ante este panorama abrumador de volúmenes de datos, distintas fuentes, velocidad y formas surge el Data Management.

Esta nueva disciplina es también conocida como un proceso de negocios de alto nivel. Consiste en diferentes sub-procesos:

· Planificación y ejecución de

· Políticas, prácticas y proyectos que

· Adquieren, controlan, protegen, entregan y refinan el valor de

· Los activos de datos e información

La misión de esta función es dar respuesta a las necesidades de información de todos los sectores de la empresa en términos de disponibilidad, seguridad y calidad de información.

Los objetivos estratégicos de la función de Data Management son:

· Entender las necesidades de información de la empresa.

· Capturar, almacenar, proteger y asegurar la integridad los activos de información.

· Continuamente mejorar la calidad de los datos e información incluyendo:

· Exactitud de los datos.

· Integridad de los datos

· Integración de la información

· El tiempo de captura y presentación de los datos

· La definición de los datos.

· Asegurar la confidencialidad

· Maximizar el valor de los activos de datos.

- Creación de Tabla de Base de Datos.

Ver siguiente enlace http://anaccimides.weebly.com/informaacutetica.html

- Propiedades de los DataControl

Explicación y ejercicios en el siguiente enlace https://lnoussanl.org/toi/archivos/TOI_LNL%20UNIDAD%203.pdf

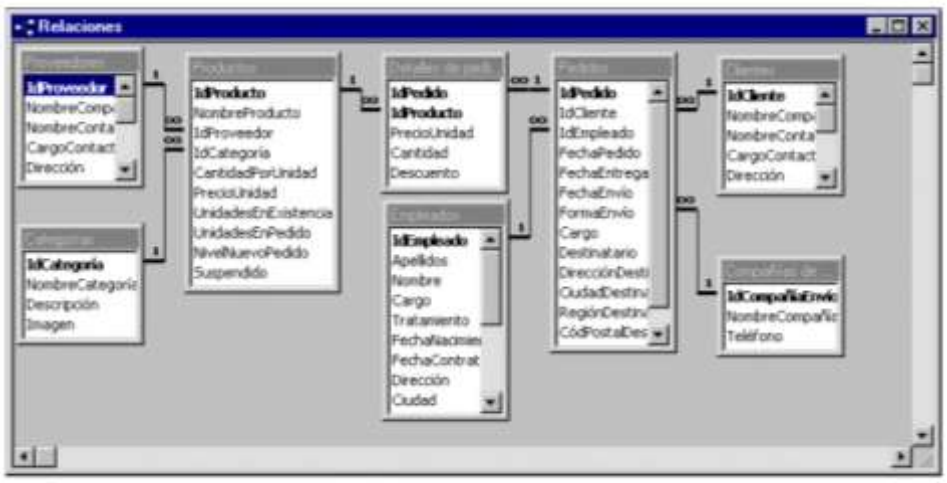
- Consulta de la base de datos con SQL

Video explicativo en el siguiente enlace https://www.youtube.com/watch?v=nFzEEpDrFNc

EVALUACIÓN

Práctica 7

Realice la siguiente tabla de datos



Referencias

https://ayudaexcel.com/los-operadores-vba/

https://docs.microsoft.com/es-es/dotnet/visual-basic/programming-guide/language-features/control-flow/decision-structures

https://docs.microsoft.com/es-es/dotnet/visual-basic/programming-guide/program-structure/main-procedure

https://help.libreoffice.org/3.3/Basic/InputBox_Function_Runtime/es

https://support.microsoft.com/es-bo/help/141693/scope-of-variables-in-visual-basic-for-applications

https://techlandia.com/definicion-eventos-visual-basic-sobre_105198/

http://visualbasic.ar.tripod.com/menu.html

https://www.discoduroderoer.es/variables-en-visual-basic-net/