DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA, UNAM
CURSOS ABIERTOS

DIVISIÓN DE EDUCACIÓN CONTINUA

CURSO: ***CC047 Introducción a la Programación***
FECHA: ***15 al 17 de julio del 2002***

EVALUACIÓN DEL PERSONAL DOCENTE

(ESCALA DE EVALUACIÓN 1 A 10)

CONFERENCISTA	DOMINIO DEL TEMA	USO DE AYUDAS AUDIOVISUALES	COMUNICACIÓN CON EL ASISTENTE	PUNTUALIDAD
Ing. Claudia Zavala Díaz				

Promedio _____

EVALUACIÓN DE LA ENSEÑANZA

CONCEPTO	CALIF.
ORGANIZACION Y DESARROLLO DEL CURSO	
GRADO DE PROFUNDIDAD DEL CURSO	
ACTUALIZACIÓN DEL CURSO	
APLICACIÓN PRACTICA DEL CURSO	

Promedio _____

EVALUACIÓN DEL CURSO

CONCEPTO	CALIF
CUMPLIMIENTO DE LOS OBJETIVOS DEL CURSO	
CONTINUIDAD EN LOS TEMAS	
CALIDAD DEL MATERIAL DIDÁCTICO UTILIZADO	

Promedio _____

Evaluación total del curso_____

Continúa...2

1. ¿Le agradó su estancia en la División de Educación Continua?

SI ☐ NO ☐

Si indica que "NO" diga porqué: _____

2. Medio a través del cual se enteró del curso:

Periódico *La Jornada*	
Folleto anual	
Folleto del curso	
Gaceta UNAM	
Revistas técnicas	
Otro medio (Indique cuál)	

3. ¿Qué cambios sugeriría al curso para mejorarlo?

_____

_____

_____

_____

_____

_____

4. ¿Recomendaría el curso a otra(s) persona(s) ?

SI ☐ NO ☐

5.¿Qué cursos sugiere que imparta la División de Educación Continua?

_____

_____

_____

_____

_____

_____

6. Otras sugerencias:

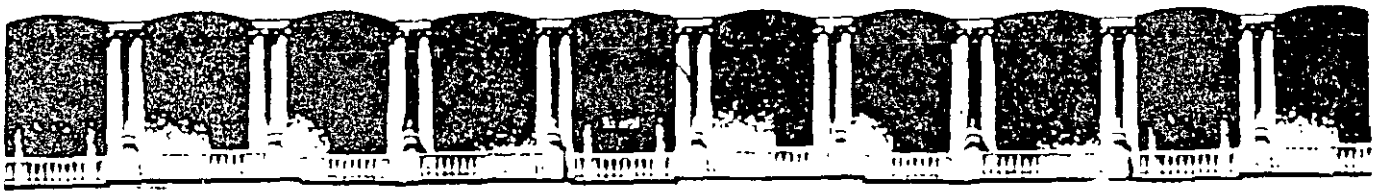
_____

_____

_____

_____

_____



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

MATERIAL DIDACTICO DEL CURSO

INTRODUCCION A LA PROGRAMACION

PROFESORA: ING. CLAUDIA ZAVALA DIAZ

JULIO, 2002

INDICE

1. ¿Qué es la programación?

Para poder definir lo que es la programación es necesario primero que es un programa

Un **programa** es un conjunto de instrucciones ordenadas y organizadas con fin determinado.

Entonces podemos decir que la **programación** es el conjunto de programas creados para resolver un problema.

Para la creación de programas debemos de tomar en cuenta los siguientes pasos:

- Poseer una idea clara del problema
- Definir las entradas y salidas que tendrá nuestro programa

Para ello nos ayudaremos de diversas herramientas que proporcionará el lenguaje (comandos, funciones, estructuras lógicas, etc).

2. La programación estructurada

La **Programación estructurada** es un conjunto de instrucciones ordenadas, modularizadas para resolver problemas o satisfacer necesidades de automatización de tareas.

En este tipo de programación, las instrucciones se van realizando de forma secuencial y con un orden determinado.

2.1 Tipos de datos

Los tipos de datos que se manejan son los siguientes:

Tipo de dato	Características	Almacén
Carácter (Character)	Alacena números, letras y símbolos	de 1 a 254 por campo
Numérico (Numeric)	Almacena números y su símbolo con precisión sencilla	de 1 a 26 incluyendo signo y punto decimal
Fecha (Date)	Almacena fechas en diferentes formatos	8 espacios
Lógico (Logical)	Almacena solo valores verdadero y falso (T o F)	1 espacio
Memoria (Memo)	Almacena texto teniendo como máximo la capacidad de disco de la máquina	10 espacios en su descripción
General	Alacena imágenes y sonido	10 espacios como definición

2.2 Operadores

Operadores matemáticos

+	Suma
-	Resta
*	Multiplicación
**	Exponenciación
/	División

Operadores relacionales

.and.	Conjunción
.or.	Disyunción
.not.	Negación

Operadores lógicos

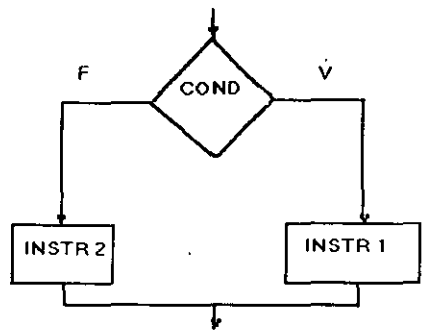
<	Menor que
<=	Menor o igual
>	Mayor que
>=	Mayor o igual
<> #	Diferente
=	Igual

2.3 Estructuras de control

IF- THEN- ELSE- ENDIF

La estructura lógica IF nos permite realizar una serie de instrucciones dependiendo del resultado de la evaluación de una condición dada Esto es, si la condición resulta verdadera realizará una o un grupo de instrucciones, si resulta falsa realizará otra u otro grupo.
La sintaxis es la siguiente:

```
IF <condición> THEN
   instrucciones 1
ELSE
   instrucciones 2
ENDIF
```



Las instrucciones 1 se realizarán cuando el valor de la condición sea verdadera de lo contrario realizará las instrucciones 2.
Es importante hacer notar que sólo se realizará un bloque de instrucciones, nunca realizará ambas, este tipo de estructuras no es cíclico, por lo que solo se realizará una sola vez.
Un ejemplo de ello es:

```
I=1
IF I<0  THEN
   LETRERO="NUMERO DECIMAL"
ELSE
   LETRERO="NUMERO ENTERO"
ENDIF
```

FOR-NEXT

La estructura For es una estructura cíclica dependiente de un contador que permitirá realizar una serie de instrucciones mientras el contador no llega a su límite, el desarrollador determinará el valor inicial del contador y el valor final que deberá alcanzar,

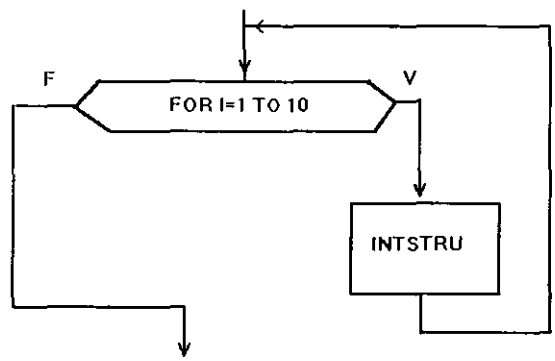
así como el valor de los incrementos (por default es 1). El ciclo por si mismo irá incrementando a la variable contador sin que el usuario se preocupe por ello. Para el caso de que en lugar de incrementar se decremente el valor de los incrementos deberá de ser con signo negativo. En el momento que el contador llegué a su límite el ciclo terminará.
La sintaxis es la siguiente:

```
FOR contador=valor_inic TO valor_fin STEP #no_incrementos
    instrucciones
NEXT
```

Un ejemplo es el siguiente:

```
FOR  I=1  to 1000  STEP 2
   suma=suma+I
NEXT
```

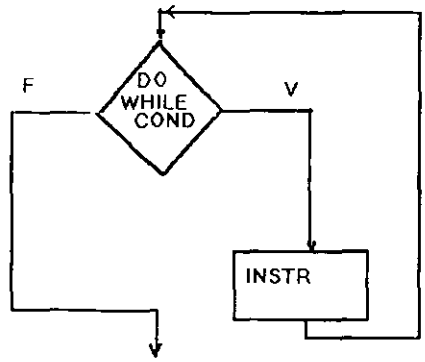
En este caso el ciclo se realizará 500 veces ya que los incrementos serán de 2 en 2.



DO WHILE - ENDDO

El Do while es una estructura cíclica que realizará una serie de instrucciones mientras la condición que evaluará resulte verdadera, en este caso el número de interaciones que se realizarán no poseen un límite como en el caso del FOR.
La sintaxis es la siguiente:

```
DO WHILE <condición>
   instrucciones
ENDDO
```



Un ejemplo de esta estructura se tiene

```
I=1
DO WHILE I<1000
      SUMA=SUMA+I
I=I+1
ENDDO
```

En este caso se utilizó el mismo ejemplo que en el For solo que aquí el número de iteraciones no está controlado y se tendrá que ir incrementando a la variable contador, es muy importante hacer notar que en la estructura WHILE se sufre el riesgo de nunca hacer falsa la condición por lo que estaríamos hablando de que el ciclo nunca terminaría y nuestro programa no pasaría de ahí, de tal forma que el programador debe de asegurarse que en todos los ciclos WHILE que maneje exista siempre una condición que rompa la estructura.

DO CASE - ENDCASE

En el caso de la estructura CASE ésta se podría sustituir por una serie de IFs anidados por lo que se respetan la mismas políticas que en el IF, solo realizará un solo bloque de instrucciones a la vez y no es una estructura cíclica, en este caso tendremos muchas opciones a elegir y dependiendo de una variable de control se realizará uno u otro bloque de instrucciones.
La sintaxis es la siguiente

```
DO CASE
    CASE <var_opcion>= valor_a
          instrucción 1
    CASE <var_opcion>=valor_b
          instrucción 2
    CASE  <var_opción>=valor_c
          instrucción 3
    OTHERWISE
   instrucción 4
ENDCASE
```

El Otherwise funciona cuando ninguna de las opciones leídas resultó existir en la lista de opciones.
Un ejemplo es:

```
@ 10, 01 say "Teclea lo opción deseada:"  get opción
read
DO CASE
    CASE opción=1
         @ 12,02 say "ALTAS"
    CASE opción=2
         @ 12,02 say "BAJAS"
    CASE opción=3
         @ 12,02 say "MODIFICACIONES"
     CASE opción=4
         @ 12,02 say "CONSULTAS"
     OTHERWISE
          @ 12,02 SAY "SALIR"
ENDDO
```

2.4 Tipos de variables

Manejaremos en este apartado los tipos de variables tomando en cuenta su duración o periodo de vida.

Públicas: Son válidas en todos los programas siendo modificables en cualquier parte del sistema. Se liberan cuando se libera la memoria. Se declaran en cualquier parte del programa.

Locales: Tienen vida mientras está vigente el programa en el que fueron creadas. Se declaran al principio del programa Cuando en un programa se declara como locales a una lista de variables y existe una llamada a alguna subrutina dentro de éste, en el momento de cambiar el control a la subrutinas, los valores locales se ocultarán, por lo que si en este programa existe una variable con el mismo nombre el valor anterior se sigue conservando ya que en el momento de salir del subprograma se liberan las variables definidas en éste y reaparecen las variables locales con sus valores.

2.5 Inicialización de variables

La inicialización de las variables consiste en darles un valor con el que iniciarán el programa, cabe aclarar que de no inicializarlas el programa marcará error al encontrarlas por primera vez. Dependiendo del tipo de la variable será la forma a inicializar.

Carácter

Se inicializa con un número determinado de espacios en blanco
Grabará en la variable carácter espacios en blanco, tantos como se le definan

Numéricas

Se inicializa con cero

Se almacena un cero en la variable

Lógicas

Solo acepta valores de verdadero o falso

Fecha

Es posible inicializar con la fecha del día
Se inicializa en blanco ayudándose de una función especial
Fecha nula

2.6 Funciones y procedimientos

Las funciones y procedimientos componen las UDFs, la diferencia principal entre ambas es el manejo de los valores de retorno. Las funciones por lo general envían parámetros de retorno, mientras que los procedimientos son procesos que se realizan repetidamente y a los cuáles bastará con llamarlos para realizar las operaciones.

Componentes
La estructura básica de una función es la siguiente:
FUNCTION <nombre de la función> (lista de parámetros)
o
PARAMETERS(lista de parámetros)
definición de variables
RETURN(valor de retorno)

Para el caso de los procedimientos

PROCEDURE <nombre del procedimiento> (lista de parámetros)
definición de variables
RETURN

Parámetros:

Los parámetros no son mas que variables de memoria utilizadas para recibir los datos o argumentos pasados hacia una función o procedure.

Ejemplo:

```
a=10
b=20
c=5
d=2
numcalc(a,b,c,d)

FUNCTION NUMCALC
PARAMETERS v1,v2,v3,v4
LOCAL nval
nval=v1*v2+(v3*1.3+v4)
RETURN(nval)
```

La manera más segura de trabajar con funciones es asegurar que no ocurrirán errores por falta de parámetros o por el paso de parámetros impropios de tal manera que aseguremos su cantidad y tipo.

Parámetros por referencia y por valor

Un argumento(parámetro) puede pasarse hacia una función de dos maneras: por referencia y por valor.
Cuando un argumento se pasa por valor (que es el modo normal de una función), su contenido original no se alterará, pues la función copia el contenido de la variable pasada hacia una variable local (dentro de la función). Esta variable se procesa y al final, el valor se retorna.

Ejemplo:

```
vr=60
?quad(vr)
?vr

FUNCTION quad
PARAMETERS num
num=num*num
RETURN (num)
```

En este ejemplo el programa principal visualizará una línea como resultado de la función y otra con el valor original de la variable vr. Notaremos que este valor se mantiene sin modificaciones después del procesamiento de la función, ya que pasó hacia la función una copia de su contenido y éste se asignó a una variable local.

Cuando se pasa una variable por referencia hacia una función, lo que pasa es su dirección de memoria en cuanto a su contenido, esto es , el comando PARAMETERS no creará una variable local dentro de la función, sino un alias para la variable utilizada como argumento, de esta forma la variable pasada sufrirá las modificaciones.

Para indicar el hecho de que una variable debe pasarse como referencia, se debe preceder del signo "@" y el nombre de la variable.

Ejemplo:

```
ml=50                  // Argumento pasado por valor retorna 2500
?quad(ml)              // Retorna 50
?ml
?
?"_____"
?quad(@ml)             // Argumento pasado por referencia, retorna 2500
?ml                    // Retorna 2500
```

3. Programación orientada a objetos

3.1 Conceptos fundamentales: clase, objeto, herencia

La programación orientada a objetos surge como una evolución natural de la programación estructurada. No es algo que rompa esquemas clásicos como podría suponerse. La Programación orientada a objetos formaliza técnicas utilizadas en dicha programación estructurada. Esta es la mejor manera de comprender en un primer momento su misión. Podemos decir que la Programación orientada a objetos aplica los objetos del mundo real a nuestras aplicaciones, nuestras aplicaciones las van a conformar entidades llamadas objetos.

Un objeto es algo intangible que tiene unas características y un comportamiento. En el mundo real podemos hablar del objeto silla, automáticamente construimos en nuestro cerebro un determinado tipo de silla, pero en cualquiera de los casos lo hacemos siempre con las características con las que nuestra percepción reconoce el objeto silla. Tendrá un respaldo, un asiento, un soporte, un color, etc Ese concepto de silla es lo que llamamos clase.

Una Clase es una abstracción de un objeto, es algo intangible, es el conjunto de ideas, no es algo real. A la acción de crear un objeto partiendo de una clase llamada instanciamiento.

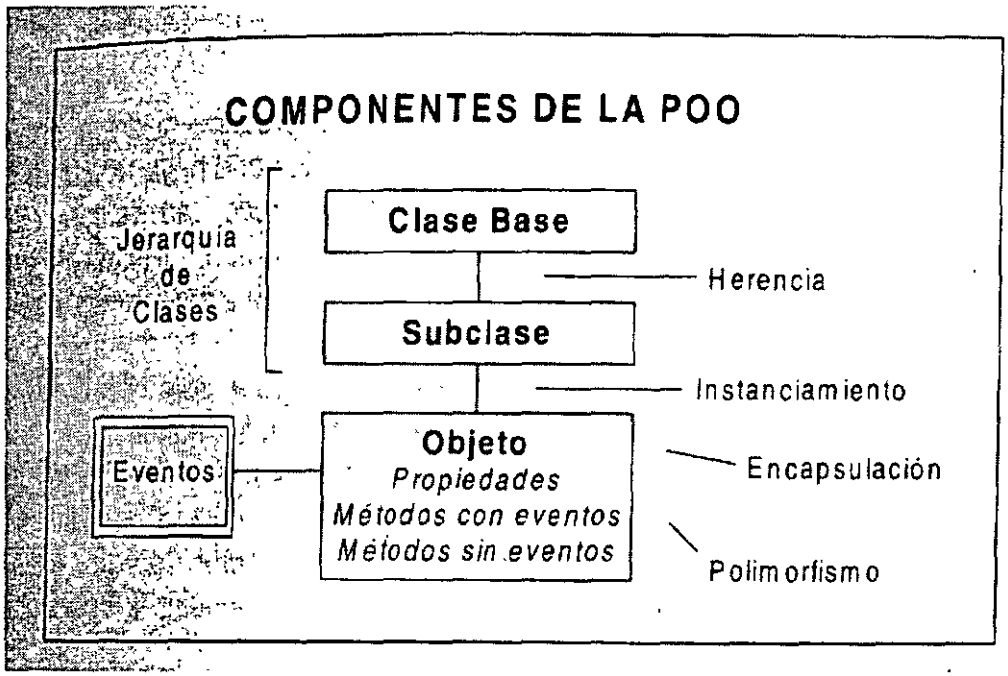
Ya tenemos dos conceptos que en la Programación orientada a objetos van a ser fundamentales, las clases y los objetos

Volviendo al ejemplo de la silla, si la abstracción, esto es, a la clase silla, le damos unas características específicas como pueden ser: tapicería de cuero, color negro, soporte giratorio y con ruedas, respaldo doble alto, tendríamos en la mente el nombre silla de ejecutivo. Es realmente un objeto, pero un objeto que tiene características tan definidas y que se repiten en muchos objetos silla, que podemos considerar a este tipo como una clase de silla. Ahora bien, a partir del concepto genérico de silla, estaremos hablando de una subclase. Por tanto, de una clase genérica o mejor llamada clase base, podemos crear subclases y éstas a su vez tener mas subclases, así formaremos jerarquía de clases.

3.2 Propiedades, eventos y métodos

Hasta ahora hemos halado de las características de los objetos. Bien, en la terminología de la Programación orientada a objetos, a éstas se le llaman propiedades Por tanto todos los objetos tienen unas propiedades que los distinguen de otros. Del mismo modo, el mundo el es algo dinámico y, por tanto, las cosas, esto es, los objetos, están expuestos a acciones externas y, a pesar de ser inanimados, pueden cambiar su estado dependiendo de estas acciones. Por ejemplo, si nos recostamos sobre el respaldo flexible de una silla, éste se echará hacia atrás. Se ha producido una acción, recostarse, y una reacción, echarse hacia atrás el respaldo.

En la Programación orientada a objetos, a las acciones les llamaremos eventos y a las reacciones métodos. Si esto lo aplicamos a la programación, los eventos serán acciones del usuario sobre nuestra aplicación y los métodos serán código asociado a cada evento.



3.3 Tipos de objetos

Existen los siguientes objetos:

- Etiquetas
- Texto
- Radio botones
- Botones de oprimir
- Listas
- Cajas de selección
- Ventanas
- Spinners

Se definirá y ejemplificará cada uno de los objetos en este capítulo.

En primera instancia veremos las semejanzas que tendrán los objetos entre sí a lo cuál llamaremos generalidades, estas se explican a continuación:

Label (A): El objeto texto es cualquier etiqueta o letrero que incluyamos dentro de la forma.

Cuadro de texto (ab): Lugar donde se almacenará información.

Botón de comando: Es un solo botón al cuál se le asignará código para realizar operaciones específicas.

Grupo de comandos: Es un objeto que reúne a varios botones comando, este caso tendrán comportamiento como grupo y en forma individual. Cualquier botón que sea seleccionado accionará a un proceso o llamará a otra aplicación.

Grupo de opciones: En este caso tendremos en lugar de botones de oprimir, radio botones. Al igual que el grupo de comandos el grupo de opciones estará formado por un conjunto de éstas.
En este caso el manejo del objeto es diferente ya que las opciones son mutuamente excluyentes, esto es no es posible seleccionar mas de una a la vez. Por ejemplo, una persona no puede ser sindicalizada y de confianza al mismo tiempo, por lo que sólo se podrá elegir una de las opciones Si una es seleccionada automáticamente deseleccionará a la otra.

Casilla de verificación: En este caso es un objeto individual el cuál se seleccionará o no, tan solo posee dos estados 1 o 0. En el momento de ser seleccionado aparecerá una cruz dentro de él.

Cuadro combinado: Lista de opciones en donde se podrá elegir la deseada. La lista aparecerá enrollada, bastará con oprimir la flecha con el mouse para poder desplegar el contenido total de ésta. Este objeto se utilizará para cuando se tienen opciones fijas de selección y no se desea que el usuario teclee la información como quiera, esto nos ayudará a tener un control más exacto de los datos. Por ejemplo, la delegación, el estado, el país, departamento, etc.

Cuadro de lista: La diferencia con la lista anterior es que esta se desplegará en un área mayor dentro de la pantalla pudiéndonos desplazar mediante las flechas de navegación a través del contenido de la lista En este caso es un objeto que podremos rellenar con campos de una tabla, con el contenido de un campo dentro de una tabla, etc.

Control numérico: Es el objeto que se utiliza para manejar valores numéricos. Por ejemplo la antigüedad de un empleado dentro de una empresa. Simplemente aumentará o disminuirá el usuario el número mediante las flechas que posee la caja en la que se encuentran los valores

Marco de página: Cuando poseemos demasiada información que se desea aparezca en una pantalla, es conveniente dividirla en forma razonable en una o dos pantallas. El marco de página nos permite hacer esto sin que cada pantalla sea una forma nueva. Dentro de este podremos manejar folders diferentes con información distinta. Por ejemplo, se manejarán datos de clientes, el primer folder serían los datos generales, en el segundo las compras y en el tercero las condiciones de pago.
Nuevamente tenemos que el objeto completo es el marco y se encuentra formado por páginas y cada página posee objetos.

3.4 Propiedades de los objetos

Llamaremos propiedades de los objetos a las características propias de cada uno.

Alignment Alineación del texto respecto al área definida por este.

Backcolor: Color que se utilizará como fondo en los objetos. Será una tercia de colores que nos dará la combinación deseada. Esto se asemeja al ColorRGB de la versión anterior. Al dar doble click en esta propiedad el sistema nos mostrará los colores que es posible seleccionar, al elegir uno Visual define la combinación correspondiente.

Caption: Título que se le asignará al objeto. Por ejemplo, en las formas será el nombre que aparecerá en el marco.

Disablebackcolor: Color del que se verá el fondo del objeto cuando se encuentra deshabilitado.

Disableforecolor. Color del que se verá la letra o contenido del objeto cuando éste se encuentre deshabilitado.

Fontbold: Para el caso de que se deseara que la letra estuviera en bold o remarcada.

Fontname: Nombre del font o tipo de letra a utilizar. Por default utiliza la letra Arial.

Fontsize. El tamaño del font o tipo de letra El default es de 10 puntos pero algunas veces se requiere que sea más grande en el caso de títulos o mas chico cuando son etiquetas de otros objetos.

Forecolor: Color de la letra dentro de un objeto.

Height: Alto del objeto

Left: A que distancia se encuentra de la izquierda de la orilla de la forma el objeto.

StatusBarText: Texto que desplegará en la barra de estado. Este texto se utilizará como texto de ayuda para el usuario, donde el programador le dará una breve explicación del tipo de dato que se almacenará o las restricciones utilizadas. Este texto se desplegará en el momento que el usuario se encuentre en el objeto que lo posee.

Top: A que distancia del tope superior de la forma se encuentra el objeto Esta propiedad es muy útil para alinear los objetos.

Visible. Propiedad que nos permitirá jugar con los objetos en cuanto a que sean o no vistos. Por ejemplo, se desea que un cierto objeto aparezca sólo cuando una condición resultó verdadera, por lo que en el diseño se incluye este objeto, pero se le da la propiedad de visible igual a falso y por medio de código se cambiará ésta a verdadero.
Es muy importante mencionar que en el diseño de la forma deberán de incluirse todos aquellos objetos que se utilicen en algún momento, aún cuando aparezcan y desaparezcan como resultado de algún proceso o condición.

Width: Ancho del objeto.

Otras Propiedades

Enabled: Esta propiedad nos permite habilitar o deshabilitar a un objeto. Por ejemplo, el botón de grabar los datos de captura se encuentra deshabilitado hasta que no se termina de capturar lo mas importante del registro. Esta propiedad también es posible modificarla mediante código.

Name: Nombre que se le asigna al objeto. Anteriormente mencionamos que Visual le da por default el nombre de acuerdo a la clase a la que pertenece el objeto adicionándole un número secuencial. Este nombre es posible modificarlo.

Propiedades específicas

Forma

Autocenter: Cuando muestra la forma centrada respecto a la pantalla.

BorderStyle. Tipo de línea que utilizará para el borde de la forma (Ajustable, sin borde, doble y sencillo)

Closable: El que aparezca o no el botón de cerrar la forma. Esto es algo que deberá de definir el programador, ya que puede darse el caso de que el usuario cierre la forma sin que hubiera terminado de capturar.

Controlbox: Si se desea que aparezca o no el botón del extremo superior izquierdo de la forma donde nos permitiría cambiarnos entre ventanas, cerrarla, etc.

Desktop. La forma desktop es aquella que ocupa todo el espacio del escritorio de trabajo de Windows.

Icon: Es el icono que deseamos que aparezca cuando la forma se encuentra minimizada.

LockScreen: Candado que se le pone a la forma para que no sea visualiuzada.

Maxbutton: El que aparezca o no el botón de maximizar

Minbutton: El que aparezca o no el botón de minimizar.

Movable: El que la forma pueda o no ser movida del lugar en el que se encuentra.

Picture: Tapiz que utilizará la forma.

Windowtype: El tipo de ventana de que se trata (Sin modo, Modal). En este caso el que se encuentre sin modo nos permitirá que el usuario pueda cambiarse de ventana o de aplicación sin haber terminado con lo que estaba haciendo. Utilizando el tipo Modal no permitirá que el usuario se cambie de ventana hasta que no halla cerrado correctamente en la que está.

Alignment: Alineación de la etiqueta (Izquierda, derecha, centro) con respecto al área en la que fue definida.

Autosize: Que el área definida se ajuste o no al tamaño del texto.

Backstyle. El estilo del fondo (Opaco, Transparente). Cuando es opaco se verá el área de la etiqueta definida con otro color o con un marco, para el caso de transparente no se verá mas que el texto.

BorderStyle: Se le define si se requiere marco en la etiqueta o si se omite.

Cuadro de texto

Century: Activar o desactivar el manejo del siglo en campos tipo fecha.

Format: Formato de entrada y salida de los datos.
A continuación definimos los formatos que se utilizan:

Inputmask: Muestra como es posible introducir la información dentro de un text.
Es posible utilizar el siguiente código.

PasswordChar: Carácter que se utilizará para campos en los que no se quiere mostrar el contenido de lo capturado, por ejemplo: Password.
Readonly: Que el dato mostrado sea de solo lectura.

Value: El valor que posee inicialmente el campo o con el que lo inicializamos.

Cuadro de edición

MaxLength: Máxima longitud del texto a escribir en la región de edición

ScrollBars. Las barras de desplazamiento que mostrará en la ventana de edición. (Ninguna, Vertical)

Botón de comando

Picture: Será la imagen que le pegaremos al botón.

Cuadro de lista

MoverBars: Aparezcan un marcado de desplazamiento para los elementos de la lista, permitiendo al usuario reordenarlos.

Control numérico

Increment: Incrementos de cuanto irá aumentando el contador numérico.

KeyboardHighValue: El valor mas alto que se puede introducir por teclado.

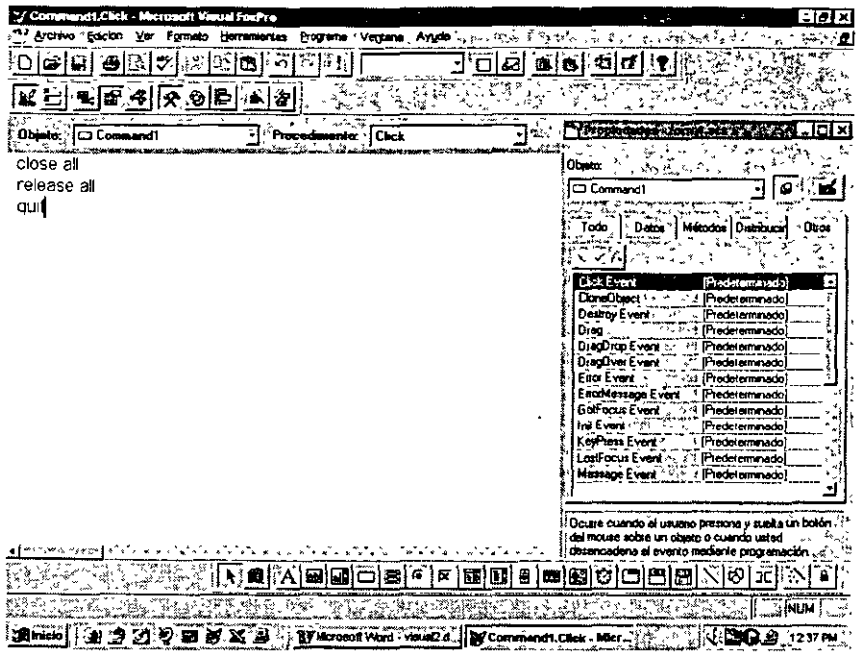
KeyboardLowValue: El valor mas pequeño que se puede introducir por teclado

SpinnerHighValue: El valor mas alto que se puede obtener mediante las flechas de desplazamiento.

SpinnerLowValue: El valor mas bajo que se puede obtener mediante las flechas de desplazamiento.

3.5 Manipulación y programación de Clases y eventos

Click: Cuando sensibiliza el sistema un click del mouse sobre el objeto. Pensemos que se tiene un Command Button o botón de comandos, él cuál al oprimirse cerrará tablas, desactivará y eliminará de memoria las ventanas activas. Estas acciones deseamos sucedan en el momento de darle click con el mouse.
Primer paso. Llamaremos al evento Click del botón de comandos, esto mediante un doble click sobre el método dentro de la ventana de propiedades. Al hacerlo se nos mostrará la pantalla siguiente:



Analizando la pantalla veremos que en la parte superior izquierda nos indicará el nombre del objeto y del lado derecho el procedimiento. Aquí es donde debemos de incluir el texto correspondiente a los procesos que deseamos realice.

Dobleclick· En este caso es cuando sensibiliza un doble click el objeto.

RightClick· En este caso es cuando sensibiliza que el botón derecho del mouse es el que está realizando la llamada al método

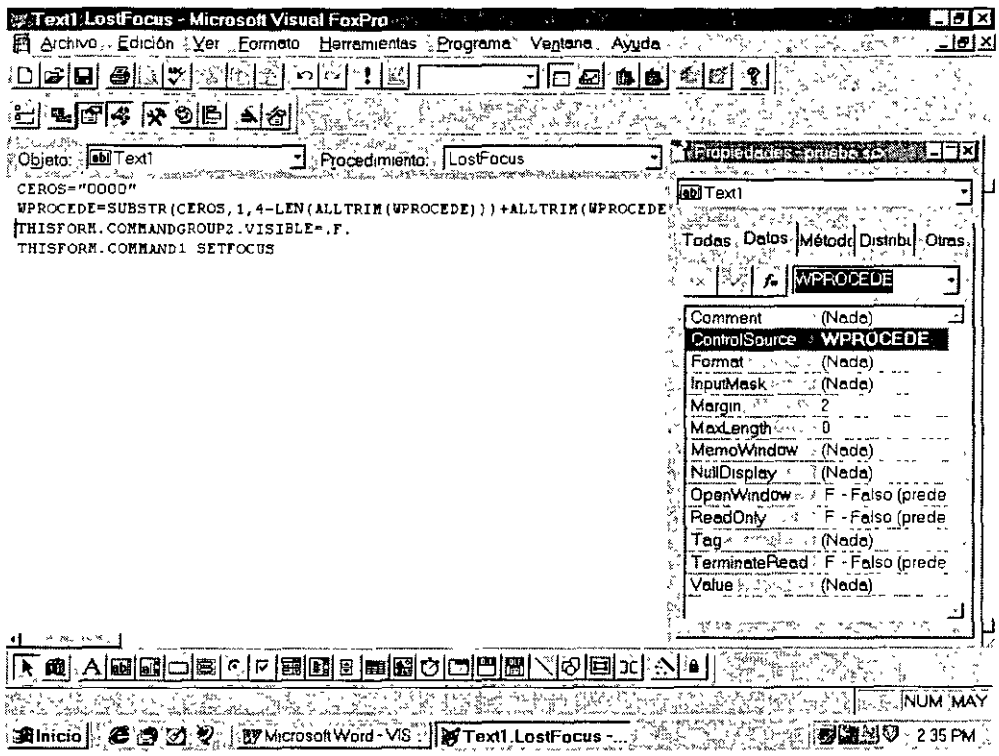
MiddleClick: Llama al método al sensibilizar el botón medio del mouse.

KeyPress. Cuando el objeto detecta que se ha oprimido una tecla siempre recorre al evento Keypress para verificar si no es una tecla programada. Veamos que sucedería si en el caso de oprimir ESC. Quisiéramos que enviara un mensaje a pantalla preguntando si el usuario desea salir del proceso que se está realizando.
En algunos casos los procesos nos proporcionarán parámetros que el Visual los hace viajar a éste para comodidad en la manipulación de la información.
Para el evento Keypress tenemos dos parámetros: nKeyCode, nShiftAltCtrl El primero nos proporcionará el código de la tecla que se oprimió (ASCII) y el segundo el valor para el caso de que utilicemos teclas combinadas con las de control.

Init: Son todos aquellos procesos, definiciones e inicializaciones de variables que se requiere se procesen en el momento que se abre la forma. Es aquí donde declararemos variables públicas, para el caso de las formas.
Los siguientes eventos son de los mas importantes en cuanto a su manejo:

GotFocus: Es el momento en que el objeto se encuentra en foco esto es que, la atención de Visual se encuentra puesta en él Este método es útil para realizar funciones que requerimos cuando nos encontramos dentro de un objeto.

LostFocus: A diferencia del anterior el Lostfocus es cuando el objeto acaba de perder el foco esto es, ahora la atención de Visual se encuentra puesta en otro objeto Este método lo utilizaremos para programar acciones que dependerán del valor del objeto que acabamos de abandonar. Por ejemplo, en la forma de captura existe un área donde le determinamos si el cliente es local o foráneo. Para el caso de foráneo necesitamos se llene un dato correspondiente al estado y país de origen, pero si es local no. Entonces cuando el objeto donde le determinamos el origen del cliente pierde el foco le se validará la condición anterior, teniendo lo siguiente.



Métodos Específicos

Forma

Activate: Es aquel método que se ejecuta cuando la forma se abre, se ejecutará cuantas veces se vuelva a activar a la forma, a diferencia del Init que sólo se realiza una vez (la primera vez que se abre)

Release: Cuando deseamos abandonar la forma y eliminarla de memoria.

Refresh: Refresca a la forma, esto es renueva los valores de las variables.

Cuadro de texto

ProgrammaticChange: Este evento se programa cuando el valor del objeto está en constante cambio, por lo que el evento se dispara al sensibilizarse esto.

SetFocus: Es cuando cualquier objeto envía el foco a otro. Ahora veamos a los tres eventos relacionados con el foco.

FOCO

SE TIENE (GOT) PIERDE (LOST) ADQUIERE (SET)

Estos son tres momentos importantes de un objeto ya que dependiendo es lo que podemos hacer o manipular.

Valid: Es el método que nos permitirá realizar la validación de una entrada, esta puede ser tan compleja como lo requiera el desarrollo

When: Este método es una pregunta donde podremos condicionar salidas, búsquedas despliegues de información, etc. Podríamos decir que traduciéndolo sería Cuando suceda esto haz esto otro.

4. Programación orientada a eventos

4.1 Programación orientada a eventos

Los lenguajes visuales orientados al evento y con manejo de componentes dan al usuario que no cuenta con mucha experiencia en desarrollo, la posibilidad de construir sus propias aplicaciones utilizando interfaces gráficas sobre la base de ocurrencia de eventos.
Para soportar este tipo de desarrollo interactúan dos tipos de herramientas, una que permite realizar diseños gráficos y, un lenguaje de alto nivel que permite codificar los eventos. Con dichas herramientas es posible desarrollar cualquier tipo de aplicaciones basadas en el entorno.

Comparación de programaciones

1. Programación Estructurada:
    a. Diseño modular
    b. Diseño descendente
    c. Utilización de estructuras de control básicas (no goto): secuencia, decisión, repetición
2. Programación Orientada a Objetos: encapsulación de datos y funciones en paquetes llamados objetos con la propiedad de ocultar información.
3. Programación Orientada a Eventos: consiste en la ejecución de código cuando se lleva a cabo una acción sobre un control. Las porciones de código a ejecutarse no son controladas por la aplicación en si sino por el código asociado a los eventos de un control.