Universidad de Los Andes
Facultad de Ciencias Forestales y Ambientales
Escuela de Ingeniería Forestal

INTRODUCCIÓN A LA PROGRAMACIÓN
con aplicaciones en Visual Basic

María Alejandra Quintero M.

Mérida - Venezuela
Enero 2005

INDICE

Presentación

Este texto fue elaborado para estudiantes de un curso inicial de programación, específicamente para los alumnos de primer año de la Escuela de Ingeniería Forestal de la Facultad de Ciencias Forestales y Ambientales de la Universidad de Los Andes, Mérida – Venezuela. Sin embargo, puede ser utilizado por estudiantes de otras carreras o por cualquier persona que desee aprender a programar.

Se asume que el lector no tiene experiencia previa en programación, por lo que se hace especial énfasis en la lógica básica. El lenguaje de programación utilizado es Visual Basic, por ser un lenguaje de fácil aprendizaje y muy atractivo para programadores principiantes y expertos, debido a la sencillez con que permite hacer programas para Windows.

En general, los textos sobre Visual Basic están orientados a lectores que han programado en otros lenguajes, por lo cual se dedican a explicar detalladamente las características propias de este lenguaje, la sintaxis, controles, eventos, funciones, etc., sin profundizar en la lógica de programación. Por otra parte, en su gran mayoría la bibliografía sobre lógica de programación utiliza pseudo-código, o se orienta a otros lenguajes clásicos en la enseñanza de programación, tales como Pascal.

Por tal razón, en este texto se presta especial atención a las estructuras básicas de programación, mostrando sus aplicaciones en Visual Basic. En los primeros capítulos, los programas que se construyen siguen el enfoque tradicional, es decir, son programas secuenciales donde el usuario introduce los datos, el programa hace los cálculos y muestra los resultados. Por lo tanto, no se aprovechan las facilidades que Visual Basic brinda para la construcción de interfaces gráficas. Más adelante, en el capítulo 5 se explica cómo realizar programas con una interfaz gráfica, introduciendo los conceptos de formulario, controles y programación por eventos.

Es importante aclarar que este texto no es una referencia precisa de Visual Basic, pues sólo trata aspectos básicos del mismo; sobre este lenguaje existe abundante bibliografía y material en Internet, así como también la ayuda disponible en el programa. El objetivo es presentar algunos conceptos básicos de programación utilizando Visual Basic, así como también ejemplos y ejercicios propuestos que permitan una mejor compresión de la teoría.

1. Conceptos básicos de programación

1.1 ¿Qué es la programación?

Es la acción de escribir programas de computación con el fin de resolver un determinado problema.

El arte de programar implica escribir instrucciones para decirle a la computadora cómo procesar información específica.

Antes de comenzar a programar, es necesario conocer los conceptos de dato, constante, variable, algoritmo y diagrama de flujo. Asimismo, es conveniente definir los pasos que deben seguirse para construir un programa.

1.2 Datos

Un dato es la representación de un hecho, evento o elemento del mundo real. Por ejemplo, un empleado de una empresa puede ser representado por varios datos: nombre, cédula de identidad, cargo, edad, sexo, etc.

Puede decirse que los datos son todos aquellos objetos que la computadora puede procesar.

1.3 Tipos de datos

Los tipos de datos básicos utilizados en computación son los siguientes:

- Entero
- Real
- Carácter
- Cadena de caracteres
- Lógico

1.3.1 Datos de tipo entero: son números que no tienen componentes fraccionarios o decimales. Pueden ser negativos o positivos.

Ejemplos de datos tipo entero son:

-2 25000
30 -1250

1.3.2 Datos de tipo real: son números que tienen punto decimal y pueden ser positivos o negativos.

Ejemplos:

801.3 3550.5
-3.5 -100.1

1.3.3 Datos de tipo carácter: son símbolos que el computador reconoce. Un carácter puede ser una letra (A, B, ..., Z, a, b,.....z), un dígito (1, 2, ...,9) o un símbolo (! , @ , # , $, % , ^ , * , & , +, - , ...).

Un dato de este tipo sólo contiene un carácter, y debe estar entre comillas.

Ejemplos:

"M", "&", "9"

1.3.4 Datos de tipo cadena de caracteres*:* son datos que contienen una sucesión de caracteres delimitada por comillas.

Los siguientes son datos de tipo cadena de caracteres:
"Simón Bolivar"
"Lic. Mendoza"
"31 de diciembre de 1999"
"1000 $"

1.3.5 Datos de tipo lógico: son datos que sólo pueden tomar uno de dos valores, verdadero o falso. Se conocen también como datos de tipo booleano. Este tipo de datos se utiliza para representar las alternativas (si / no) a determinadas condiciones.

Ejemplo: se desea saber si una persona es soltera, en este caso la respuesta será verdadera o falsa y puede ser representada mediante un dato de tipo lógico.

1.4 Constantes y Variables

Los datos que maneja un programa pueden ser constantes o variables. A continuación se definen ambos términos.

1.4.1 Constante: es un valor o dato que no puede cambiar en la ejecución de un programa. Las constantes son valores fijos.

Una constante tiene dos atributos que la caracterizan: nombre y valor.

Ejemplos:

Pi = 3.1416
Mínimo = 20
Empresa = "Corporación M & M"
EdadMaxima= 50
Respuesta = Falso
Clase = "A"

El valor dado a una constante determinará su tipo. Así, Pi es una constante real ya que su valor es un número real. Las constantes Mínimo y EdadMáxima son de tipo entero. Empresa es una constante de tipo cadena de caracteres. La constante clase es tipo carácter y respuesta es de tipo lógico.

1.4.2 Variable: es un valor o dato que puede cambiar durante la ejecución de un programa. Una variable representa una dirección o posición de memoria donde se guarda un dato.

Todo dato que vaya a ser introducido en la computadora, y todo valor que se calcule a partir de otros datos en un programa, debe manejarse como una variable.

Una variable tiene dos atributos: un nombre que la identifica y el tipo de dato que describe su uso.

Los siguientes son ejemplos de variables:

NOMBRE	TIPO
Diámetro	Real
Nota	Entero
Ciudad	Cadena de caracteres

Una variable que es de cierto tipo solamente puede tomar valores de ese tipo. Por ejemplo, a la variable nota no podría dársele el valor 11.5 porque su tipo es entero y 11.5 es un número real; en este caso se originaría un error.

1.5 Pasos para la construcción de un programa

Elaborar un programa de computación implica llevar a cabo una serie de pasos que comienzan con la definición y análisis del problema, y conducen a la implantación de un programa que lo soluciona. Los pasos que generalmente sigue cualquier programador a la hora de construir un programa son los siguientes:

Análisis: tiene como finalidad conocer y comprender el problema. En esta fase se definen cuáles son los datos necesarios, qué debe hacer el programa y cuáles son los resultados que debe arrojar.

Una técnica que ayuda a realizar el análisis en forma ordenada es el análisis de entrada-proceso- salida, también llamado análisis E-P-S, el cual se describe en la sección 1.6.

Diseño: consiste en especificar cómo se resuelve el problema. Durante esta fase se establece la secuencia de pasos que debe seguirse para obtener la solución del problema. Esta secuencia de pasos es un esquema en base al cual se escribirá el código del programa.

Dos herramientas que se utilizan en el diseño del programa son los algoritmos y los diagramas de flujo, éstos se explican con detenimiento en las secciones 1.7 y 1.8 respectivamente.

Codificación: es la traducción de cada uno de los pasos especificados en el diseño a un lenguaje de programación, siguiendo las reglas de sintaxis del mismo. El resultado de esta fase será el programa escrito en el computador, llamado también **código fuente**.

Ejecución y pruebas: consiste en ejecutar (correr) el programa para observar su funcionamiento y detectar fallas. Durante esta fase se recomienda probar el programa con una amplia variedad de datos para encontrar y corregir todos los errores que puedan presentarse, de esta manera se evita que el programa produzca resultados erróneos en situaciones específicas. A la acción de encontrar y corregir errores se le conoce como **depuración** del programa.

El resultado esperado al finalizar los cuatro pasos antes descritos, es un programa de computación que funcione correctamente y que solucione el problema planteado.

A continuación se explican con detenimiento el análisis y diseño. Para llevar a cabo la codificación, ejecución y pruebas del programa, es necesario conocer un lenguaje de programación; razón por la cual en el capítulo 2 se presentan los fundamentos básicos del lenguaje Visual Basic.

1.6 Análisis de entrada – proceso - salida

Una manera fácil y ordenada de realizar el análisis del problema, es dividir dicho análisis en tres partes: entrada, proceso y salida.

Entrada: en esta parte se especifican cuáles son los datos necesarios para resolver el problema (datos de entrada) y de qué tipo son.

Proceso: se indican los procesos que se van a realizar con los datos de entrada, a través de fórmulas y expresiones escritas de la manera más sencilla posible.

Salida: aquí se explican cuáles son los resultados esperados.

Ejemplo 1: supóngase que se quiere realizar un programa para calcular el área de un triángulo. El análisis del problema usando la técnica de entrada – proceso – salida, es el siguiente:

Entrada
Los datos necesarios para resolver el problema son:

b: base del triángulo. Tipo: Real
h: altura del triángulo. Tipo: Real

Proceso
Calcular el área del triángulo usando la ecuación:

$$A = \frac{b \times h}{2}$$

Salida
A: área del triángulo. Tipo: Real.

1.7 Algoritmos

Un algoritmo es una secuencia ordenada de pasos que llevan a la solución de un problema o a la ejecución de una tarea. Los pasos deben ser simples, claros y exactos, seguir un orden lógico, y tener un principio y un fin. En la vida diaria se utilizan algoritmos; por ejemplo cuando se prepara una receta de cocina o cuando se siguen instrucciones para armar algún objeto (juguete, mueble, etc.); en cualquier caso el algoritmo indica cada paso en el orden apropiado.

Ejemplo 1: algoritmo para un cajero automático simple.

1. Obtener clave secreta del usuario
2. Si la clave no es válida, dar un mensaje de error e ir al paso 9.
3. Si la clave es válida, preguntar el usuario el tipo de transacción, depósito o retiro, y la cantidad.
4. Obtener del banco el saldo actual
5. Si el tipo de transacción es depósito, sumar la cantidad al saldo actual.
6. Si el tipo de transacción es retiro, consultar el saldo actual.
   6.1 Si la cantidad es mayor que el saldo actual, mostrar un mensaje de error e ir al paso 9.
   6.2 Si la cantidad es igual o menor que el saldo actual, restar la cantidad del saldo actual y entregar efectivo.
7. Mostrar el saldo actual
8. Preguntar al usuario si desea efectuar otra transacción. En caso afirmativo ir al paso 3.
9. Mostrar mensaje “Gracias por usar el cajero automático”
10. Fin del algoritmo

Para ser correcto, un algoritmo debe reunir las siguientes características:

- Debe ser claro y no ambiguo.
- Debe resolver el problema correctamente.
- Debe ejecutarse en un número finito de pasos.

Durante el desarrollo de algoritmos para computadora, es necesario idear los pasos que la máquina deberá seguir para resolver el problema planteado. Es importante especificar cada paso, aunque algunos parezcan demasiado obvios. El algoritmo puede incluir fórmulas o expresiones matemáticas.

En el siguiente ejemplo se muestra un algoritmo típico para computadora.

Ejemplo 2: algoritmo para calcular el área de un triángulo.

1. Obtener base del triángulo (b)
2. Obtener altura del triángulo (h)
3. $A = \dfrac{b \times h}{2}$
4. Escribir el área (A)
5. Fin

Obsérvese que este algoritmo comienza con la obtención de los datos necesarios para resolver el problema, llamados datos de entrada (pasos 1 y 2); se recomienda colocar entre paréntesis el nombre de las variables donde se almacenarán los datos. Luego, se describe el proceso, mediante el uso de la ecuación correspondiente (paso 3). Una vez efectuado el proceso está la instrucción *Escribir*, con la cual se indica que el resultado debe ser presentado al usuario del programa; además se especifica entre paréntesis el nombre de la variable que almacenará el resultado para facilitar la codificación.

Ejemplo 3: análisis E-P-S y algoritmo para calcular el salario de un trabajador, al cual se le paga de acuerdo a las horas trabajadas.

a) Análisis E-P-S
Entrada
Nom: nombre del trabajador. Tipo: cadena de caracteres
nh: número de horas trabajadas. Tipo: Real
T: tarifa por hora. Tipo: Real

Proceso
Calcular el salario del trabajador usando la ecuación:

S = nh x T

Salida
S: salario del trabajador. Tipo: Real.

b) Algoritmo

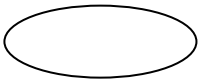
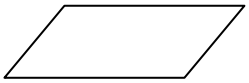
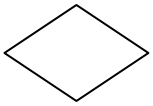
1. Obtener el nombre del trabajador (Nom)
2. Obtener el número de horas trabajadas (nh)
3. Obtener la tarifa por horas (T)
4. S = nh x T
5. Escribir nombre (nom) y salario del trabajador (S)
6. Fin

1.8 Diagramas de flujo

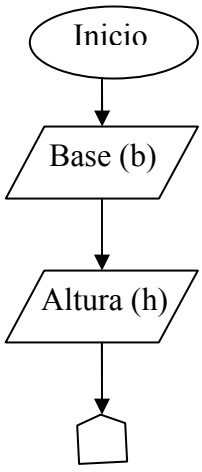
Un diagrama de flujo es una representación gráfica de un algoritmo, que utiliza símbolos para indicar acciones. Los símbolos se conectan a través de flechas que muestran el flujo o secuencia del programa.

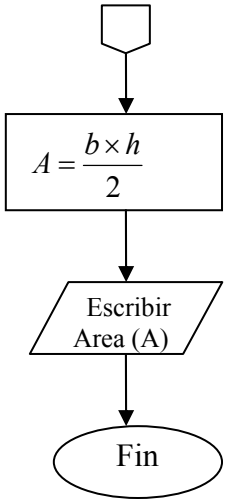
En el siguiente cuadro se muestran los símbolos más utilizados en la construcción de un diagrama de flujo.

Cuadro 1.1 Símbolos de los diagramas de flujo

SÍMBOLO	SIGNIFICADO
	Inicio / Fin del programa
	Entrada / Salida de datos
	Procesos
	Decisión
	Conector para una misma página
	Conector de página diferente

Ejemplo 1: diagrama de flujo para calcular el área de un triángulo.





1.9 Ejercicios propuestos

Realizar análisis E-P-S, algoritmo y diagrama de flujo para resolver los siguientes problemas:

1. Obtener la calificación promedio de un alumno que presenta tres exámenes.

2. Calcular el cuadrado de un número

3. Calcular la suma, resta y multiplicación de dos números reales.

4. Determinar la hipotenusa de un triángulo rectángulo, teniendo como datos las longitudes de sus catetos.

5. Calcular el precio total a pagar por un artículo si se tiene como dato el precio de venta y se sabe que el IVA es del 15%.

6. Conocido el radio de un círculo, calcular su longitud, el diámetro y el área de la circunferencia.

2. Introducción a Visual Basic

2.1. ¿Qué es Visual Basic?

Visual Basic es un lenguaje de programación que permite crear aplicaciones (programas) para Windows. Usando Visual Basic se pueden construir en forma fácil, programas con una interfaz gráfica que puede incorporar elementos como ventanas, botones, cuadros de diálogo, cuadros de texto, botones de opción y de selección, barras de desplazamiento, menús, etc., propios de cualquier aplicación bajo Windows.

En una aplicación Visual Basic, el programa está formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interfaz gráfica. Es por tanto, un término medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos. Combina ambas tendencias sin embargo, no puede decirse que Visual Basic pertenezca por completo a uno de esos dos tipos de programación.

En este capítulo se especifican los fundamentos básicos del lenguaje Visual Basic y la sintaxis utilizada para crear el código de los programas. Además, se explican programas muy sencillos que siguen el enfoque de programación tradicional, es decir, programas secuenciales donde el usuario introduce los datos, el programa hace los cálculos y muestra los resultados. Más adelante en el capítulo 5, se explica cómo realizar programas con una interfaz gráfica.

2.2. Elementos del lenguaje Visual Basic

2.2.1 Tipos de datos

Los tipos de datos que maneja Visual Basic se especifican en el cuadro 2.1.

Cuadro 2.1. Tipos de datos que acepta Visual Basic

TIPO	DESCRIPCIÓN	RANGO
Byte	Entero corto	0 a 255
Integer	Entero (2 bytes)	-32768 a 32767
Long	Entero largo (4 bytes)	-2147483648 a 2147483647
Single	Real de simple precisión. (4 bytes)	-3.40 E+38 a 3.40E+38
Double	Real de doble precisión (8 bytes)	-1.79E+308 a 1.79E+308
Currency	Número con punto decimal fijo (8 bytes)	-9.22E+14 a 9.22E+14
String	Cadena de caracteres	0 a 65500 caracteres
Date	Fecha y hora (8 bytes)	Fecha: 01/01/100 a 31/12/9999 Hora: 0:00:00 a 23:59:59
Boolean	Lógico o booleano	True o False
Variant	Guarda información de cualquier tipo de dato: enteros, reales, caracteres, fecha y hora.	Mismo rango que el tipo de valor almacenado.
Object	Tipo especial de dato que contiene referencias a objetos como controles y formularios.	

2.2.2 Declaración de variables y constantes

La declaración de variables o constantes implica decirle a la computadora cuántas variables y/o constantes se utilizarán en el programa, cómo se llamarán y el tipo de datos que contendrán.

Declaración de constantes

Para declarar una constante en Visual Basic únicamente es necesario utilizar la palabra **CONST** seguida del nombre de la constante y su correspondiente valor. La sintaxis es:

```
Const   nombre_constante =  valor
```

Nombre_constante: es el nombre que el programador le da a la constante que se está declarando.

Valor: valor asignado a la constante.

Ejemplo de declaración de constantes:

```
Const  PI = 3.1416
Const Max = 350
Const  Saludo = "Hola"
```

Declaración de variables

En Visual Basic hay diferentes formas de declarar una variable. La **Sentencia DIM** es la forma más común. Puede emplearse en un Procedimiento, Función, Formulario o Módulo. La sintaxis es la siguiente:

```
Dim     nombre_variable As tipo
```

Nombre_variable: es el nombre que el programador le da a la variable que se está declarando.
Tipo: tipo de dato asociado a la variable.

Se le puede colocar cualquier nombre a una variable, siempre y cuando cumpla con las siguientes reglas:

1. El nombre de una variable tiene que comenzar siempre por una letra y puede contener hasta 255 caracteres.

2. El nombre sólo puede contener letras, números y el carácter de subrayado _ . No se aceptan espacios en blanco.

3. No pueden utilizarse como nombres de variables las palabras reservadas de Visual

Basic, como por ejemplo: if, next, for, val, caption, etc. Para saber cuáles son las palabras reservadas se puede consultar el *Help* de Visual Basic, en la referencia *Reserved Words*. Las palabras reservadas aparecen en color azul cuando se escribe el código del programa.

Ejemplos de declaraciones de variables:

Dim Edad as byte

Dim Nom_Estudiante as string

Dim salario as single

Dim area as double, saldo as single

Mediante estas declaraciones, el programa sabe de qué tipo de dato se trata y por tanto cómo debe trabajar con él.

Existen otras formas de declarar las variables en Visual Basic, aquí sólo se utilizará DIM.

Visual Basic no distingue entre mayúsculas y minúsculas. Por ejemplo, las variables SalarioTotal y salariototal son consideradas como una misma variable y no como dos.

Debe evitarse declaraciones de variables como la siguiente:

Dim X, Y as integer

En este caso no se declaran dos variables de tipo integer como pudiera pensarse a simple vista; se declara Y de tipo integer y X de tipo Variant.

En Visual Basic no es necesario que se declaren todas las variables que se van a utilizar. Cuando en elcódigo se escribe una variable nueva que no ha sido declarada , Visual Basic asume que es una variable y que su tipo es el adecuado para el valor que le está asignando en ese momento.

Por ejemplo, si Visual Basic encuentra estas instrucciones

Nombre ="Pedro González"
CI = "1234567"
Nota=18

entiende que *Nombre, CI* y *Nota* son variables, que *Nombre* y *CI* son cadenas de caracteres (string) porque su valor está entre comillas, y que *Nota* es un número (su valor no esta entre comillas).

Esta particularidad de no necesitar declarar las variables hace que sea sencillo introducir una variable nueva. Sin embargo, puede ser una fuente de errores. Supóngase que en un paso posterior del programa, se desea hacer una operación con la variable *Nota,* y que

por error el programador escribe *Nata* en vez de *Nota.* Visual Basic interpreta que *Nata* es una variable e irá a leer en memoria el valor que tiene. No tendrá ningún valor. Por lo tanto la operación no se hará en forma correcta y además no dará ningún aviso de que se ha cometido un error.

Para evitar este tipo de errores se recomienda declarar todas las variables que se van a utilizar en el programa. Para que Visual Basic dé un mensaje de error cuando se utiliza una variable no declarada previamente se puede utilizar la instrucción *Option Explicit,* la cual se explica a continuación.

OPTION EXPLICIT
Obliga a declarar previamente las variables que se vayan a usar. De no haberla declarado antes de usarla, el programa dará una comunicación de error.

2.2.3 Instrucción de asignación

Una vez que se elige un nombre y un tipo de dato para una variable, se le debe dar un valor. Existen tres métodos principales para dar valores a una variable:

1. Asignarle un valor dentro del programa.
2. Pedir al usuario que teclee un valor.
3. Leer un valor de un archivo de datos.

En esta sección se explicará la instrucción de asignación. La introducción de valores con el teclado se tratará en el apartado 2.5 de este tema. La lectura de archivos no se incluye en estos apuntes.

La sintaxis de una instrucción de asignación es:

Nombre_variable = valor o expresión

Ejemplos:

1) A = X+Y +2
El resultado de sumar las variables X y Y, se le asigna a la variable A.

2) Salario = 1250000
A la variable salario se le asigna el valor 1250000

3) Nombre_ alumno = "José Rodríguez"
A la variable nombre_alumno se le asigna el valor José Rodríguez. Obsérvese que cuando se asigna una cadena de caracteres, ésta debe ir entre comillas.

4) A=B

El contenido de la variable B se le asigna a la variable A.

Existen algunas reglas para instrucciones de asignación que hay que tener presente:

1. Sólo un nombre de variable puede ir a la izquierda del signo igual, porque indica la ubicación de memoria que cambiará.

2. El valor a la derecha del signo igual puede ser una constante (ejemplos b y c), otra variable (ejemplo d) o una fórmula o expresión que combine constantes y variables (ejemplo a).

3. La variable y su valor deben ser del mismo tipo de datos.

2.2.4 Operadores y expresiones aritméticas

Los operadores aritméticos que se utilizan para las operaciones básicas son:

+	Suma
-	Resta
*	Multiplicación
/	División
\	División sin decimales
Mod	Resto de una división
^	Exponenciación

Ejemplos:

1) Y = 12 + 10	→	Y = 22
2) Y = 12 - 10	→	Y = 2
3) Y= 12 * 10	→	Y = 120
4) Y = 12 / 10	→	Y = 1.2
5) Y= 12 \ 10	→	Y = 1
6) Y = 12 Mod 10	→	Y = 2
7)Y = 12 ^ 2	→	Y = 144

Muchas veces es necesario construir una expresión que incluya varios operadores aritméticos, es decir, una fórmula. En ese caso hay que considerar el orden en que se ejecutan las operaciones para escribir correctamente la expresión. En el cuadro 2.2 se indica el orden de precedencia de los operadores aritméticos.

Cuadro 2.2. Orden de precedencia de los operadores aritméticos.

OPERADOR	ORDEN DE PRECEDENCIA
( )	1
^	2
* /	3
\	4
Mod	5
+ -	6

Ejemplos:

1) Y = ((6 * 3) / 2) ^ 2 → Y= 81

2) Y = (7 * 8 * (16 mod 3) \ 5)* 3 – 28 → Y= 5

3) Y = 3 * 10 * (17 mod 3) \ 5 * 3 -28 → Y= -24

Algunas funciones numéricas

En el cuadro 2.3 se muestran algunas funciones matemáticas y trigonométricas que tiene Visual Basic, las cuales pueden ser útiles en la construcción de fórmulas.

Cuadro 2.3. Algunas funciones matemáticas y trigonométricas

FUNCIÓN	DESCRIPCIÓN
Abs ( )	Devuelve el valor absoluto de un número
Sqr ( )	Devuelve la raíz cuadrada de un número
Round (x, d)	Redondea un número real x a un número con d dígitos después del punto decimal. Si se omite la cantidad de decimales d, round aproximará el número al entero más cercano.
Exp ( )	Función exponencial. Devuelve e elevado al número indicado entre paréntesis.
Log ( )	Devuelve el logaritmo en base e de un número.
Sgn ( )	Devuelve 1 si el signo del argumento es positivo, y -1 si es negativo.
Sin ( )	Devuelve el seno de un ángulo expresado en radianes.
Cos ( )	Devuelve el coseno de un ángulo expresado en radianes.
Tan ( )	Devuelve la tangente de un ángulo expresado en radianes.
Atn( )	Devuelve el arco tangente de un ángulo expresado en radianes.

Ejemplo:

Considérese la siguiente fórmula matemática $Z = \frac{|x-3|+\sqrt{x}}{x^2}$.

La expresión aritmética equivalente en Visual Basic es Z = (abs(x -3) + srq(x)) / x^2

2.2.5 Operaciones de Entrada / Salida simple

Recuérdese que el flujo básico del procesamiento por computadora es entrada, proceso, salida. En los problemas que aquí se resuelven, se asume que hay un usuario sentado frente a la computadora introduciendo datos con el teclado, el programa acepta estos datos (entrada), los procesa y luego muestra el resultado en la pantalla (salida).

En Visual Basic existen varias formas de gestionar las entradas y salidas. En este capítulo se explicarán las más simples: cuadros de entrada (InputBox) y cuadros de Mensaje (MsgBox).

2.2.5.1 InputBox

Un InputBox es una ventana donde se le solicita información al usuario, tal como puede verse en el siguiente ejemplo:

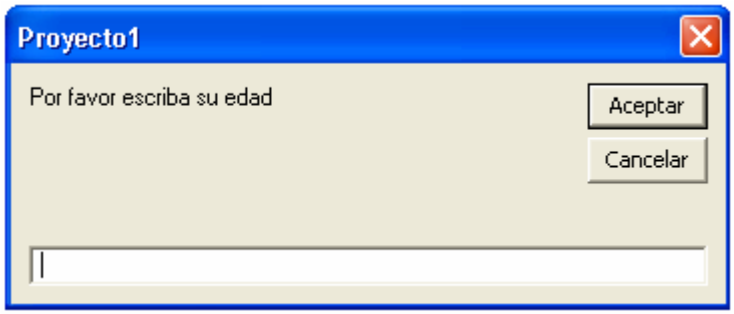


Figura 2.1. Cuadro de entrada (InputBox)

El InputBox escribe un mensaje que da información al usuario, en la figura 2.1 le está indicando que escriba la edad. Además presenta un cuadro donde el usuario puede escribir lo que se le está solicitando.

Sintaxis:

```
Nombre_variable = InputBox ("mensaje")
```

Esta es la expresión más sencilla de un InputBox, donde:

Nombre_variable: corresponde al nombre de la variable en la cual se almacenará el valor que escriba el usuario.

Mensaje: es la frase que aparecerá en el InputBox antes del cuadro donde el usuario puede escribir su respuesta. El mensaje debe ir entre comillas.

Ejemplo 1:
Para que aparezca el InputBox de la Figura 2.1, debe escribirse el siguiente código:

```
edad = InputBox("Por favor escriba su edad")
```

Nombre de la variable donde se almacenará el valor que el usuario escriba

Mensaje (escrito entre comillas)

Los cuadros de entrada o InputBox siempre incluyen los botones Aceptar y Cancelar. Si el usuario hace clic en Aceptar (o presiona la tecla Enter), lo que haya escrito se almacenará en la variable indicada (para el ejemplo, sería la variable edad). Si se presiona cancelar, a la variable se le asigna una cadena vacía " ".

Un InputBox tiene como título predeterminado el nombre dado al proyecto Visual Basic en el momento de guardar, por ejemplo, en la figura 2.1 se observa que el título es Proyecto1. Sin embargo, es posible colocarle un título diferente, así como también un valor predeterminado para la respuesta. En ese caso la sintaxis es:

Nombre_variable = InputBox ("mensaje", "título", valor predeterminado)

Ejemplo 2:

```
Pais_nac = InputBox("País de nacimiento", "Datos personales", "Venezuela")
```

Nombre de Variable

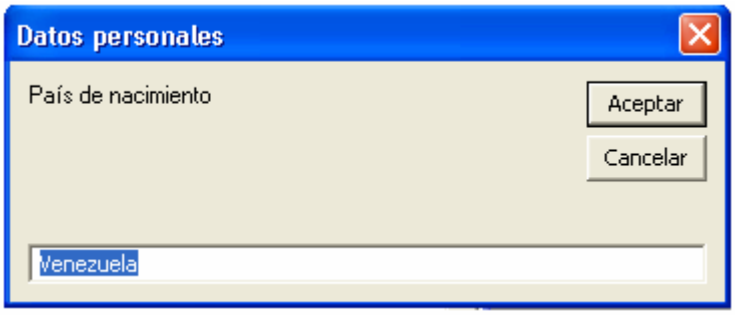
Mensaje

Título del cuadro de entrada

Valor predeterminado

Esta instrucción despliega un cuadro de entrada como el siguiente:

Figura 2.2. Cuadro de entrada del ejemplo 2 de InputBox



Observe que el valor por defecto es Venezuela, esto significa que el mismo aparecerá como respuesta predeterminada. Si el usuario está de acuerdo hace clic en Aceptar. Si no está de acuerdo puede escribir otro país.

2.2.5.2 MsgBox

Un MsgBox o cuadro de mensaje es una ventana donde se puede dar información al usuario. En la figura 2.3 se muestra un MsgBox.

Figura 2.3. Ejemplo de cuadro de mensaje (MsgBox)

Sintaxis.

La sintaxis del MsgBox en su forma más sencilla es la siguiente:

MsgBox (“mensaje”)

El MsgBox muestra el mensaje y luego espera que el usuario haga clic en Aceptar.

Ejemplo 1:

El código para generar el cuadro de mensaje de la figura 2.3 es el siguiente:

```
MsgBox ("Este es un ejemplo de MsgBox")
```

↓

Mensaje que aparece en el MsgBox

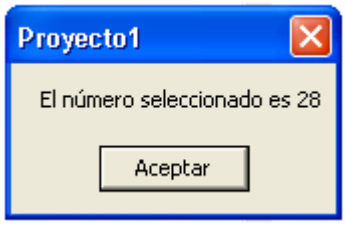
Si se desea colocar mensajes y variables en un MsgBox, éstos se concatenan con el signo &. A continuación se dan algunos ejemplos.

Ejemplo 2:
Supóngase que se tiene el siguiente código:

```
a=7
num = 4*a
MsgBox ("El número seleccionado es" & num)
```

Estas instrucciones hacen que un programa muestre el siguiente cuadro de mensaje:

Figura 2.4. Ejemplo 2 de MsgBox



Ejemplo 3:
La instrucción

```
MsgBox ("línea 1" & Chr(10) & "línea 2")
```

muestra el cuadro de mensaje de la figura 2.5.

Figura 2.5. Ejemplo 3 de MsgBox

Existen opciones que permiten cambiar la apariencia de un cuadro de mensaje, mostrando otros botones además del botón Aceptar, un título específico, un icono, etc. Así por ejemplo, si se desea colocar otros botones y/o un título específico en un MsgBox, debe utilizarse la siguiente sintaxis:

Nombre_variable = Msgbox (“mensaje”, tipos_de_botones, “título”)

Nombre_variable: se refiere a una variable donde se almacenará un número entero entre 1 y 7, el cual indica el botón sobre el cual el usuario hizo clic. En el cuadro 2.4 se muestra el significado de cada número.

Tipos_de_ botones: es un número entero entre 0 y 5, o una constante de Visual Basic que indica la combinación de botones que se desea para el MsgBox. En el cuadro 2.5 se muestran los posibles valores.

Título: texto que aparecerá en la barra de título del cuadro de mensaje.

Cuadro 2.4 Valores de retorno de un MsgBox

VALOR	CONSTANTE VISUAL BASIC	SIGNIFICADO
1	vbOK	El usuario hizo clic en el botón Aceptar
2	vbCancel	El usuario hizo clic en el botón Cancelar
3	vbAbort	El usuario hizo clic en el botón Abortar
4	vbRetry	El usuario hizo clic en el botón Reintentar
5	vbIgnorar	El usuario hizo clic en el botón Ignorar
6	vbYes	El usuario hizo clic en el botón Sí
7	vbNo	El usuario hizo clic en el botón No

Cuadro 2.5 Botones que pueden mostrarse en un MsgBox

VALOR	CONSTANTE VISUAL BASIC	BOTONES QUE APARECEN
0	vbOKonly	Aceptar
1	vbOKCancel	Aceptar y Cancelar
2	vbAbortRetryIgnore	Abortar, Reintentar, Ignorar
3	vbYesNoCancel	Sí, No, Cancelar
4	vbYesNo	Sí, No
5	vbRetryCancel	Reintentar, Cancelar

Ejemplo 4:
El MsgBox de la figura 2.6 se genera mediante la siguiente instrucción:

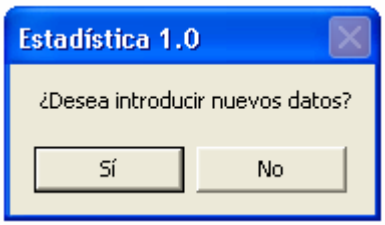
```
Resp = MsgBox("El inventario ha finalizado",0,"Sistema de Inventarios")
```

Figura 2.6. Ejemplo 4 de MsgBox

Ejemplo 5:
En el siguiente código se utiliza una constante Visual Basic para mostrar los botones Sí y No.

```
R = Msgbox ("¿Desea introducir nuevos datos?", vbYesNo, "Estadística 1.0")
```

Figura 2.7. Ejemplo 5 de MsgBox



Para incluir un icono en un cuadro de mensaje, se suma al parámetro tipos_de_botones el valor o constante Visual Basic que representa a un determinado icono. En el cuadro 2.6 se muestran los tipos de iconos disponibles.

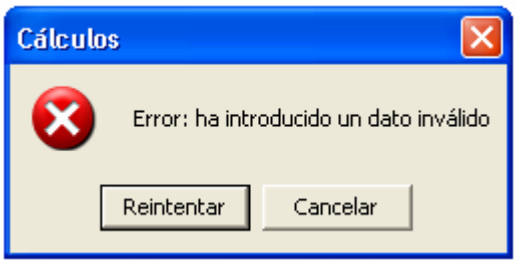
Cuadro 2.6 Tipos de iconos para un MsgBox

VALOR	CONSTANTE VISUAL BASIC	ICONO
16	vbcritical	
32	vbQuestion	
48	vbExclamation	
64	vbInformation	

Ejemplo 6:
El siguiente código muestra un mensaje de error con un icono.

```
Resp =  MsgBox ("Error: ha introducido un dato inválido", 5 + 16, "Cálculos")
```

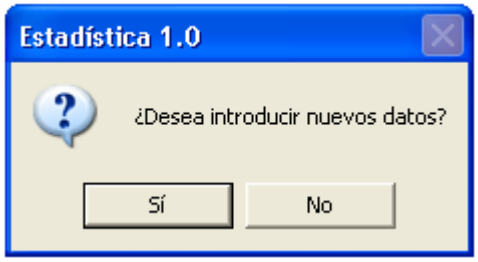
Figura 2.8. Ejemplo 6 de MsgBox



Ejemplo 7:
Para mostrar el cuadro de mensaje del ejemplo 5 con un icono de interrogación, se escribe la siguiente instrucción:

```
R = Msgbox ("¿Desea introducir nuevos datos?", vbYesNo + 32, "Estadística 1.0 ")
```

Figura 2.9. Ejemplo 7 de MsgBox

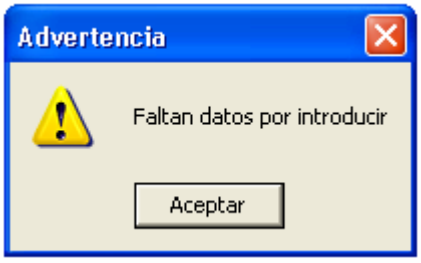


Ejemplo 8:

```
Resp = MsgBox (“Faltan datos por introducir “, vbexclamation, “Advertencia”)
```

Esta instrucción despliega el cuadro de mensaje de la figura 2.10. Obsérvese que el MsgBox sólo tiene el botón Aceptar, en ese caso no es necesario colocar 0 + vbexclamation ó vbOkonly + vbexclamation, en el segundo parámetro.

Figura 2.10. Ejemplo 8 de MsgBox



2.3. Escritura de programas

Ya se conocen los elementos básicos que permitirán escribir los primeros programas en Visual Basic. Lo que falta es colocar todo –entrada, proceso y salida – en un programa completo que la computadora pueda entender.

Ya que por ahora no se utilizan las herramientas visuales del lenguaje Visual Basic (formularios, controles, etc.), el código se coloca en una subrutina (sección de código) llamada Sub Main, la cual es el punto de partida del programa.

Ejemplo 1: programa que eleva un número entero al cuadrado (el número debe ser suministrado por el usuario).

Código del programa:

```
Option explicit
Sub main()
Dim num As Integer, cuadrado As Integer

num = InputBox("introduzca el número que desea elevar al cuadrado")
cuadrado = num * num
MsgBox (Str(num) + " elevado al cuadrado es" & cuadrado)

End Sub
```

Ejemplo 2: Programa que calcula el área de un triángulo.

```
Option explicit
Sub main()
Dim b As Single, h As Single, area As Single

b = InputBox("Introduzca la base")
h = InputBox("Introduzca la altura")
area = b * h / 2
MsgBox (" El área del triángulo es" & area )

End Sub
```

Ejemplo 3: programa que calcular el precio total a pagar por un artículo si se tiene como dato el precio de venta y se sabe que el IVA es del 16%.

```
Option explicit
Sub main()
Dim pv As Single, iva As Single, pt As Single

pv = InputBox("Precio de venta")
iva = 0.16 * pv
pt = pv + iva
MsgBox (" Precio total a pagar" & pt )

End Sub
```

2.4. Ejercicios propuestos

1. A continuación se da una lista de variables y una descripción de su contenido. Asigne a cada variable el tipo de dato de Visual Basic que considere más conveniente.

Variable	**Contenido**	**Tipo**
Suma	Resultado de sumar tres números enteros	
Saldo	Monto en bolívares disponible en una cuenta bancaria.	
Nombre	Nombre de una persona	
Enfermo	Indica si una persona presenta o no una determinada enfermedad	
NumArbol	Cantidad de árboles de una misma especie presentes en un área determinada	
Tipo	Tipo de una pintura (A, B, C o D)	

2. ¿Qué valor tiene la variable Z después de ejecutar las siguientes operaciones de asignación?

a) X = 3
   Y = 4
   Z = X - Y

b) Z= 5
   X = 2 + Z
   Z = 3
   Z = Z + X

3. ¿Qué valor tienen las variables Z y W después de ejecutar las siguientes operaciones de asignación?

a) Z = 8
   W = 4
   Y = 2
   W = W + 5
   Z = Z - Y + W

b) Z = 4
   W = 6
   Y = Z + W
   Z = W + Y
   W = Z + W

4. Obtener el valor de cada una de las siguientes expresiones aritméticas:

a) 69 \ 8
b) 69 mod 8
c) 12 \ 3
d) 12 mod 3
e) 7 * 10 – (5 mod 3) * 4 + 9
f) (7 * (10 – 5) mod 3) * 4+ 9
g) (12 + 3) + 8 * 3 mod 5 + 4 * 3
h) A * B / C * C -1 si A = 4, B = 3, C = 2

5. Escriba las siguientes fórmulas matemáticas como expresiones de Visual Basic

a) $y = \frac{a+b}{c-a}$

b) $w = \frac{x^2 + y^2}{z^2}$

c) $d = \sqrt{(x-y)^2 + (z-w)^2}$

d) $x = \ln(b+1) + seno(c)$

6. Escribir la fórmula matemática correspondiente a las siguientes expresiones de Visual Basic

a) Y = sqr (A^2 -abs(B))
b) Y = exp (x^3) – 18
c) Y =(1+sen(x)*cos(x)) / tan(x)+2

7. Escriba una instrucción en Visual Basic que permita mostrar:

a) un cuadro de entrada que solicite al usuario su nombre.
b) un cuadro de entrada que le pida al usuario que introduzca una clave y que tenga como título Sistema Automatizado,

8. ¿Qué acción realizan los siguientes fragmentos de código Visual Basic?

a)
```
num=55
Msgbox ("Número de alumnos presentes : " & num)
```

b)
```
nomb = "Juan Díaz"
MsgBox ("El mejor alumno es " & nomb)
```

c)
```
MsgBox ("Ingeniería Forestal " & Chr(10) & "Primer año")
```

d)
```
Resp = MsgBox ("¿Desea guardar los datos?", 3)
```

e)
```
Y = MsgBox (" La edad no puede ser mayor a 100", 48, "Error")
```

9. Realizar un programa en Visual Basic que resuelva los siguientes problemas:

a) Calcular el salario mensual de un trabajador si se tienen como datos el número de horas trabajadas, el precio de la hora y además se sabe que se le descuenta el 10% por concepto de caja de ahorros.

b) Convertir una temperatura dada en grados Celsius a grados Fahrerenheit y a grados Kelvin.
Las fórmulas de conversión son:
$F = (9/5) * C + 32$
$K = C + 273.1$

c) Intercambiar los valores de dos variables numéricas.

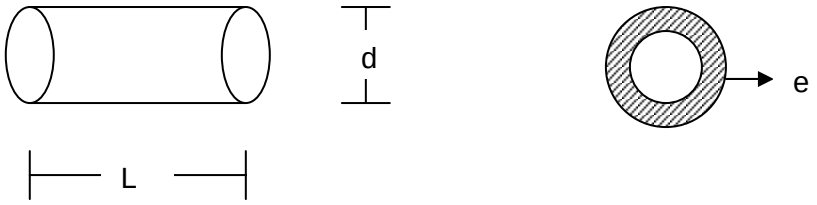
d) Dados dos números reales, calcular la suma, resta y multiplicación de dichos números.

e) En una gasolinera, los surtidores registran la cantidad de gasolina que le suministran a los clientes en galones, pero el precio de la gasolina está fijado en litros. Si se sabe que el precio del litro es de 180 Bs. y que un galón tiene 3.785 litros, calcule lo que hay que cobrarle al cliente.

f) Dados el radio y la altura de un cilindro, calcule el área y el volumen.
Las ecuaciones a utilizar son:
Area = $2 \times \pi \times$ radio x altura
Volumen = $\pi \times \text{radio}^2 \times$ altura

g) Calcular el volumen de acero de una tubería. Se tienen como datos el diámetro (d), el espesor (e) y la longitud de la tubería (L).

Volumen = superficie x espesor



3. Estructuras de Decisión

Las estructuras de decisión permiten a la computadora elegir los cursos de acción a seguir en un programa. En las estructuras de decisión se evalúa una condición y dependiendo del resultado se seleccionan las instrucciones a ejecutar. Se puede elegir entre dos o más alternativas posibles.

Antes de comenzar a estudiar los conceptos relacionados con las estructuras de decisión, es importante aprender a construir expresiones lógicas, usando operadores relacionales y operadores lógicos. Este tipo de expresiones permite definir las condiciones que utilizan las estructuras de decisión.

3.1 Operadores relacionales

Son operadores que permiten hacer comparaciones entre constantes y variables. En el cuadro 3.1 se muestran los operadores relacionales usados en Visual Basic, su significado y el equivalente en notación matemática.

Cuadro 3.1 Operadores relacionales.

OPERADOR	SIGNIFICADO	EQUIVALENTE MATEMÁTICO
>	Mayor que	<
<	Menor que	<
>=	Mayor o igual que	≥
<=	Menor o igual que	≤
=	Igual a	=
<>	Diferente a	≠

3.2 Operadores lógicos

Los operadores lógicos básicos son AND, OR y NOT. Estos operadores se aplican a operandos lógicos (booleanos), que son variables o constantes que pueden tener el valor verdadero o falso, pero no ambos a la vez.

3.2.1 Operador AND

Relaciona dos operandos booleanos. Da como resultado un valor verdadero (V), si los dos operandos son verdaderos (V); en caso contrario proporciona un resultado falso (F).

Sintaxis:

```
Operando 1  AND  Operando 2
```

Las posibles combinaciones de resultados se muestran en el cuadro 3.2.

Cuadro 3.2. Resultados de un operador AND.

Operando 1	Operando 2	Resultado
V	V	V
V	F	F
F	V	F
F	F	F

3.2.2 Operador OR

Al igual que AND, el operador OR relaciona dos operandos booleanos. Da como resultado un valor verdadero (V), si cualquiera de los dos operandos es verdadero (V); y proporciona un resultado falso (F) si los dos operandos son falsos (F).

Sintaxis:

```
Operando 1  OR  Operando 2
```

Los resultados que pueden obtenerse al aplicar un operador OR, se muestran en el cuadro 3.3.

Cuadro 3.3. Resultados de un operador OR.

Operando 1	Operando 2	Resultado
V	V	V
V	F	V
F	V	V
F	F	F

3.2.3 Operador NOT

Este operador se aplica a un operando lógico y da como resultado el valor opuesto al que tiene el operando. Esto es, si el operando es verdadero el resultado es falso, y si el operando es falso el resultado es verdadero.

Sintaxis:

```
NOT  Operando
```

Los posibles resultados se muestran en el cuadro 3.4.

Cuadro 3.4. Resultados de un operador NOT.

Operando	Resultado
V	F
F	V

3.3 Expresiones lógicas

Este tipo de expresiones se forma al combinar variables, constantes, operadores relacionales y operadores lógicos. Se llaman expresiones lógicas o booleanas porque al ser evaluadas, el resultado siempre será verdadero o falso.

Las expresiones lógicas más simples se forman al combinar variables y/o constantes con operadores relacionales. Ejemplos de este tipo de expresiones se muestran en el cuadro 3.5.

Cuadro 3.5 Ejemplos de expresiones lógicas simples

Expresión lógica	*Valor de la expresión si X=5 y Y=2*
X < 3	Falso
Y > X - 4	Verdadero
Y <= X	Verdadero
X = Y	Falso

Las variables de tipo cadena de caracteres también se pueden comparar. En este caso la computadora examina los valores de izquierda a derecha, carácter por carácter y compara el valor ASCII de cada letra. De acuerdo al código ASCII:

a < b < c < … < z
A < B < C < … < Z

Minúsculas > Mayúsculas

En el cuadro 3.6 se muestran algunos ejemplos.

Cuadro 3.6. Ejemplos de expresiones lógicas con cadenas de caracteres

Expresión lógica	*Valor de la expresión*
"Mari" < "Marianela	Verdadero
"Ana" > "José"	Falso
"Doris" > "Doria"	Verdadero

Es posible construir expresiones lógicas más complejas al utilizar los operadores lógicos AND, OR y NOT.

Ejemplos:

Sea X= Verdadero(V) y Y= Falso (F).

1. X AND Y = F

2. X OR Y = V

Se pueden utilizar varios operadores lógicos en una misma expresión. En este caso, para determinar el valor de la expresión es necesario conocer el orden de precedencia de los operadores lógicos, el cual se muestra en el cuadro 3.7

Cuadro 3.7. Orden de precedencia de los operadores lógicos

OPERADOR	ORDEN DE PRECEDENCIA
( )	1
NOT	2
AND	3
OR	4

Ejemplos:

Determinar el valor de las siguientes expresiones lógicas, suponiendo que A= V, B=V, C= F y D=F

1. NOT A AND B OR C
   NOT V AND V OR F
   F AND V OR F
   F OR F = F

   El valor de esta expresión es falso.

2. NOT (A AND C) OR B AND D
   NOT (V AND F) OR V AND F
   NOT F OR V AND F
   V OR V AND F
   V OR F = V

   El valor de esta expresión es verdadero.

Cuando se está programando, es bastante usual tener que construir expresiones lógicas que combinen operadores relacionales y operadores lógicos, como por ejemplo:

(A >= 5) AND (B< 8)

En expresiones de este tipo se evalúan primero los operadores relacionales y luego los operadores lógicos.

Si A = 8 y B = 2, el valor de la expresión anterior sería verdadero.

3.4 Estructuras de decisión simple

Este tipo de estructura evalúa una condición, que corresponde a una expresión lógica. Si la condición es verdadera, se ejecuta un conjunto de instrucciones. Si la condición es falsa se ignoran estas instrucciones.

Una estructura de decisión simple se utiliza cuando la ejecución de algunas instrucciones está condicionada, pero no hay instrucciones alternativas.

La figura 3.1 ilustra el diagrama de flujo de este tipo de instrucción.

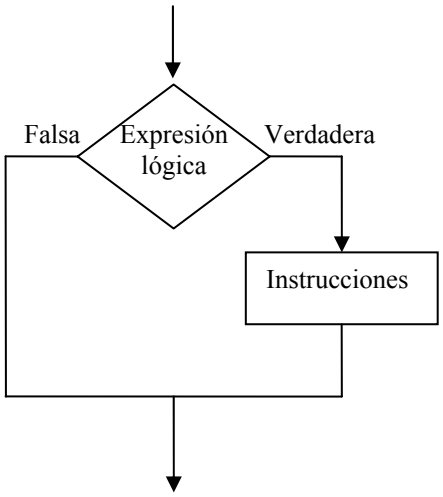


Figura 3.1. Diagrama de flujo de una estructura de decisión simple.

Si la expresión lógica es verdadera, se ejecutarán las instrucciones. Si es falsa, no sucede nada y la ejecución del programa continua en las instrucciones que van después de la estructura de decisión simple.

La sintaxis de una estructura de decisión simple en Visual Basic es la siguiente:

```
If   expresión lógica   Then

    Una o varias instrucciones

End  If
```

Ejemplo 1: programa que calcula el salario neto de un trabajador, teniendo como entrada su salario base y el número de hijos. Al trabajador se le descuenta el 5% de su salario base por concepto de seguro social, pero si tiene más de dos hijos se le pagan 50.000 Bs. adicionales.

```
Option Explicit
Sub main()
Dim sb As Single, nh As Byte, sn As Single

sb = InputBox("Salario base: ")
nh = InputBox("Número de hijos")
sn = sb - 0.05 * sb
If nh > 2 Then
    sn = sn + 50000
End If
MsgBox ("Salario Neto = " & sn)

End Sub
```

Ejemplo 2: programa que calcula el precio total a pagar por la compra de un artículo, considerando que el IVA es del 16%. Si el precio es mayor de 100.000 Bs. se realiza un des cuento del 1%.

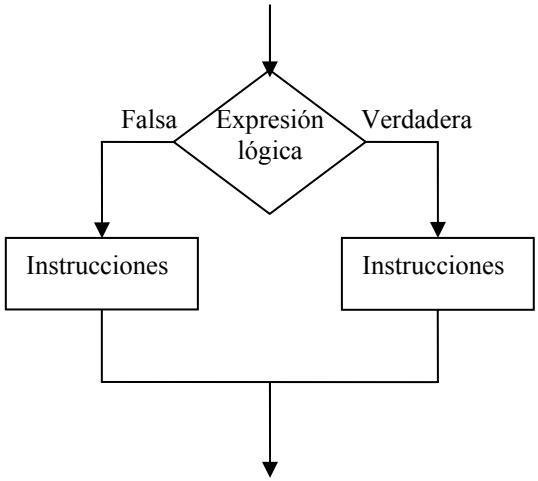
```
Option explicit

Sub main()

Dim Pv As Single, Pt As Single

Pv = InputBox("Introduzca el precio de venta del artículo")
If  Pv > 100000 then
    Pv  =  Pv  -  0.01 * Pv
End If
Pt = Pv  +  0.16 * Pv
MsgBox (" El precio total a pagar es " & pt)

End Sub
```

3.5 Estructuras de decisión doble

Se utilizan cuando la computadora debe elegir entre dos alternativas dependiendo de una condición. Una estructura de decisión doble evalúa una expresión lógica, si ésta es verdadera se ejecuta un conjunto de instrucciones, y si es falsa se ejecuta otro conjunto de instrucciones.

En la figura 3.2 se presenta el diagrama de flujo de este tipo de estructura de programación. En el mismo, se observa claramente como la computadora debe elegir entre dos cursos de acción.

Figura 3.2. Diagrama de flujo de una estructura de decisión doble.



En Visual Basic, una estructura de decisión doble se escribe de la siguiente manera:

```
If   expresión lógica   Then

    Una o varias instrucciones

Else

    Una o varias instrucciones

End  If
```

Si la expresión lógica es verdadera se ejecutan las instrucciones que están a continuación de **Then**, y si la expresión es falsa, se ejecutan las instrucciones que están después de **Else**.

Ejemplo 3: programa que calcula el promedio de un estudiante dadas tres calificaciones. Si el promedio es mayor o igual que 9.5 se debe mostrar un mensaje que indique "APROBADO". En caso contrario, el mensaje debe ser "REPROBADO".

```
Option explicit

Sub main()

Dim C1 As Single, C2 As Single, C3 As Single, prom As Single

C1 = InputBox("Introduzca la calificación 1")
C2 = InputBox("Introduzca la calificación 2")
C3 = InputBox("Introduzca la calificación 3")
Prom = (C1 + C2 + C3) / 3
If  Prom >= 9.5 then
 MsgBox (" Promedio=" & prom &  Chr(10) & Chr(10) & "APROBADO")
```

```
Else
 MsgBox (" Promedio=" & prom & Chr(10) & Chr(10) & "REPROBADO")
End If

End Sub
```

Ejemplo 4: Programa que calcula el valor de *Y*, el cual está dado por la siguiente ecuación $Y = \frac{x^3 + 5}{x}$. Si $x = 0$ debe dar un mensaje de error e indicar que no puede realizarse el cálculo.

```
Option Explicit
Sub main()
Dim X As Single, Y As Single, resp As Byte

X = InputBox("Escriba el valor de X")
If X = 0 Then
   resp = MsgBox("No puede realizarse el cálculo" & Chr(10) & "X debe
ser diferente de cero", vbCritical, "Error")
Else
   Y = (X ^ 3 + 5) / X
   MsgBox ("El valor de Y es " & y)
End If

End Sub
```

Ejemplo 5: Programa que calcula la comisión que le corresponde a un vendedor. Si vendió más de Bs. 1.000.000, la comisión es el 3% sobre las ventas. Si vendió Bs.1.000.000 o menos, la comisión es del 1% de las ventas.

```
Option Explicit

Sub main()

Dim Ventas As Single, Com As Single

Ventas = InputBox("Introduzca el monto vendido (Bs.)")

If Ventas > 1000000 Then
  Com = 0.03 * Ventas
Else
  Com = 0.01 * Ventas
End If

MsgBox (" Comisión del vendedor=" & Com )

End Sub
```

3.6 Estructuras de decisión anidadas

Se utilizan cuando hay más de dos alternativas entre las cuales se puede elegir. Cuando en una estructura de decisión, alguna de sus instrucciones es otra estructura de decisión, se dice que las estructuras están anidadas.

Hay diferentes maneras en que pueden anidarse las estructuras de decisión. En la figura 3.3 se presentan dos formas diferentes de anidamiento.

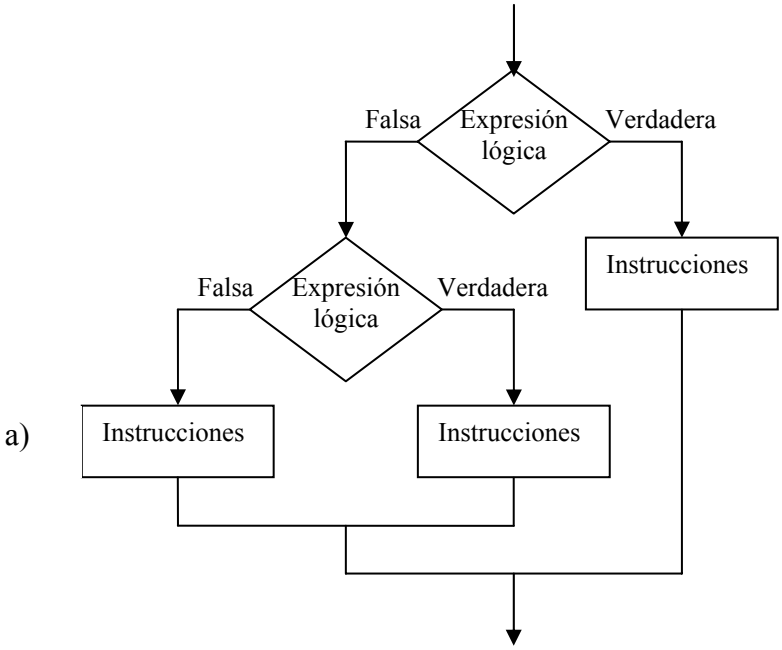
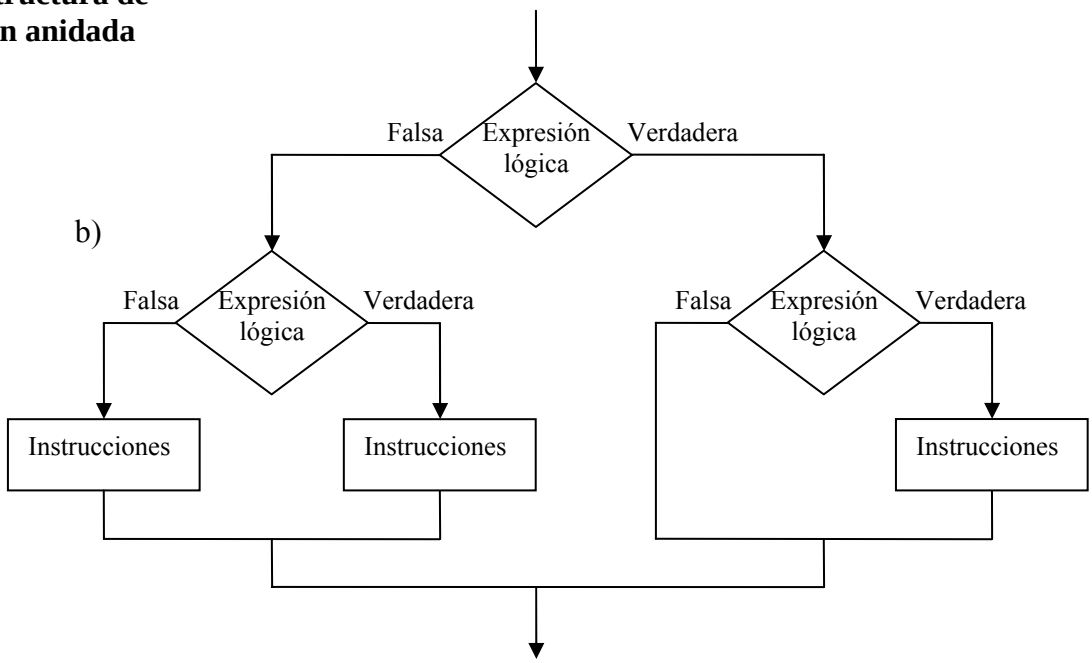


Figura 3.3. Diagrama de flujo de una estructura de decisión anidada



Visual Basic permite escribir las estructuras anidadas haciendo uso de estructuras If – then – else, tal como se muestra a continuación.

```
If    expresión lógica  then
  Una o más instrucciones
Else
      If  expresión lógica then
         Una o más instrucciones
      Else
         If expresión lógica then
             Una o más instrucciones
         End If
      End If
End If
```

Esta sintaxis corresponde a un esquema de anidamiento similar al mostrado en la figura 3.3 a). Se deja como ejercicio plantear la sintaxis correspondiente a la figura 3.3 b).

Ejemplo 6: programa para clasificar una especie forestal de acuerdo a su resistencia. El dato de entrada del programa es el porcentaje de pérdida de peso de la especie y la salida es uno de los siguientes mensajes.

Mensaje	***% Pérdida de peso***
Altamente resistente	[0 – 1]
Resistente	(1 - 5]
Moderadamente resistente	(5 – 10]
Muy poco resistente	(10- 30]
No resistente	Más de 30

```
Option Explicit
Sub main()
Dim pp as Single

Pp = InputBox ("Introduzca el porcentaje de pérdida de peso")
If  pp <=1  then
     MsgBox ("Especie forestal altamente resistente")
Else
  If   pp <= 5 then
     MsgBox ("Especie forestal resistente")
  Else
    If  pp <= 10 then
        MsgBox ("Especie forestal moderadamente resistente")
    Else
      If pp<= 30 then
           MsgBox ("Especie forestal muy poco resistente")
      Else
           MsgBox ("Especie forestal resistente")
      End If
    End If
  End If
End If
End Sub
```

Ejemplo 7: Programa para calcular el precio a pagar por la compra de madera. Los datos de entrada son la cantidad de metros cúbicos a comprar, el precio por metro cúbico y el tipo de madera. La madera está clasificada en tres tipos (A, B y C). Si la cantidad a comprar es superior a 30 metros cúbicos, se aplica el siguiente esquema de descuento:

Tipo de madera	Descuento
A	4 %
B	8 %
C	10 %

Si la cantidad comprada es inferior a 30 metros cúbicos el descuento es del 2%, independientemente del tipo de madera.

```
Option Explicit
Sub main()
Dim cant as Single, pre as Single, pre_tot as Single, desc as Single
Dim TipoM as String

TipoM = InputBox ("Introduzca el tipo de madera")
cant = InputBox ("Introduzca la cantidad en metros cúbicos a comprar")
pre = InputBox ("Introduzca el precio por metro cúbico")

pre_tot = cant * pre

If cant > 30 then
  If TipoM= "A" then
     Desc = 0.04 * pre_tot
  Else
     If TipoM = "B" then
        Desc = 0.08 + pre_tot
     Else
        If tipoM = "C" then
           Desc = 0.1 * pre_tot
        End If
     End If
  End If

Else
   Desc= 0.02 * pre_tot

End If

Pre_tot = pre_tot – Desc
MsgBox (" El precio total a pagar es " & pre_tot)
End Sub
```

Ejemplo 8: programa que determina cuál es el mayor de tres números.

```
Option Explicit
Sub main()
Dim a As Single, b As Single, c As Single, mayor As Single

a = InputBox("Escriba el primer número")
b = InputBox("Escriba el segundo número")
```

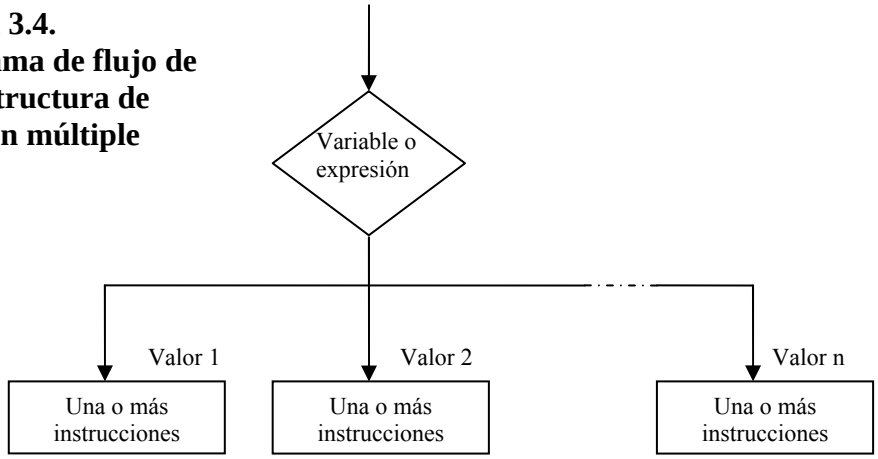
```
c = InputBox("Escriba el tercer número")
If a > b Then
   If a > c Then
        mayor = a
   Else
         mayor = c
   End If
Else
   If b > c Then
         mayor = b
   Else
         mayor = c
   End If
End If
MsgBox (" El número mayor es = " & mayor)
End Sub
```

3.7 Estructuras de decisión múltiple

Al igual que las estructuras de decisión anidadas, las estructuras múltiples se utilizan cuando se quiere elegir entre varias alternativas.

En una estructura de decisión múltiple se especifica una variable o expresión, la cual puede tomar diferentes valores, dependiendo del valor que tenga se ejecutarán las instrucciones pertinentes. En la figura 3.4 se observa el diagrama de flujo de este tipo de estructura.

Figura 3.4.
Diagrama de flujo de una estructura de decisión múltiple



Visual Basic tiene la instrucción *Select Case*, que corresponde a una estructura de decisión múltiple. Su sintaxis es la siguiente:

```
Select Case  Variable o Expresión
Case primer valor
     Una o más instrucciones (1)
Case segundo valor
     Una o más instrucciones (2)
Case tercer valor
     Una o más instrucciones (3)
.
.
.
Case Else
     Una o más instrucciones (4)

End Select
```

El *Select Case* funciona de la siguiente manera: si la variable o expresión toma el primer valor se ejecuta el bloque de instrucciones (1), si es igual al segundo valor se ejecutan las instrucciones (2), y así sucesivamente. En el caso de que la variable o expresión no sea igual a ninguno de los valores especificados, se ejecutan las instrucciones que están después del *Else*. Una vez que Visual Basic ejecuta el *Case* que coincida, ignorará los *Case* restantes y continuará con el código que esté después del *End Select*.

La parte *Case Else* de la estructura *Select Case* es opcional, sólo se coloca si es necesario. Si no se utiliza *Case Else* y la variable no toma ninguno de los valores especificados en los *Case*, se ejecuta el código que sigue a *End Select*.

Ejemplo 8: Programa que recibe como dato de entrada un número entero entre 1 y 7, y escribe el día de la semana correspondiente.

```
Option explicit
Sub main( )
Dim  número as Integer
número = InputBox (“Escriba un número entre 1 y 7 “)
Select Case número
Case 1
   MsgBox (“Domingo“)
Case 2
   MsgBox (“Lunes“)
Case 3
   MsgBox (“Martes“)
Case 4
   MsgBox (“Miércoles“)
Case 5
   MsgBox (“Jueves“)
Case 6
   MsgBox (“Viernes“)
Case 7
   MsgBox (“Sábado“)
Case Else
   MsgBox (“Número fuera de rango“)
End Select
End Sub
```

Formatos adicionales del Select Case

Existen dos formatos adicionales del Select Case, que permiten utilizar esta estructura cuando se desea hacer comparaciones o utilizar un rango de valores. A continuación se presenta el primer formato adicional:

```
Select Case Variable o Expresión
Case Is  Relación:
         Una o más instrucciones
Case Is  Relación:
         Una  o más instrucciones
Case Is Relación:
         Una o más instrucciones
.
.
.
Case Else:
        Una o más instrucciones

End Select
```

Donde relación puede ser cualquier prueba de comparación, por ejemplo >5, <25.

El segundo formato adicional permite considerar rangos de valores, su sintaxis es la siguiente:

```
Select Case Variable o Expresión
Case valor_inicial To valor_final:
     Una o más instrucciones
Case valor_inicial To valor_final:
         Una  o más instrucciones
Case valor_inicial To valor_final:
     Una o más instrucciones
.
.
.
Case Else:
        Una o más instrucciones

End Select
```

Ejemplo 9: Programa que clasifica a una persona de acuerdo a su edad, Las posibles clasificaciones son: bebé, niño, adolescente, adulto y anciano.

```
Option Explicit
Sub main()

Dim edad As Integer
Dim Tipo As String
edad = InputBox("Escriba la edad")
Select Case edad
Case Is < 2:
      Tipo = "Bebé"
```

```
Case 2 To 12:
      Tipo = "Niño"
Case 13 To 18:
      Tipo = "Adolescente"
Case 19 To 69:
      Tipo = "Adulto"
Case Is >= 70:
      Tipo = "Anciano"
End Select

MsgBox (Tipo)
End Sub
```

Como puede observarse en los ejemplos, un *Select Case* cumple la misma función que una estructura de decisión anidada con *If -Then -Else*, pero de una forma más ordenada, la única desventaja es que sólo es posible evaluar una expresión, mientras que con If pueden usarse tantas como sean necesarias. Por ejemplo, si en un programa se quiere hacer un descuento a los clientes tipo A que compren un monto superior a 200000 Bs, no puede usarse un *Select Case* ya que hay que considerar dos criterios para tomar la decisión, uno referente al tipo de cliente y el otro al monto de la compra. Es precisamente por eso que existen estas dos posibilidades, estructuras de decisión anidadas y múltiple, cuando una no es válida, se puede usar la otra.

3.8 Ejercicios propuestos

1. Resolver las siguientes expresiones lógicas

a) NOT (X AND Y OR X) si X= Verdadero y Y=Falso

b) NOT (A > B) OR (B > 4) AND (A < = 1) si A= 5 y B= 8

2. Construir una expresión lógica cuyo valor sea verdadero, si un número X se encuentra en el intervalo [3, 10). El resultado de la expresión debe ser falso si X se encuentra fuera de este intervalo.

3. Construir una expresión lógica cuyo valor sea verdadero, si la edad de una persona es mayor a 60 años o menor a 12 años. El resultado de la expresión debe ser falso si la edad no cumple con estas condiciones.

4. Construir una expresión lógica cuyo valor sea verdadero, si una persona es de sexo femenino y tiene más de 25 años. En otros casos, el resultado de la expresión debe ser falso.

5. Hacer un algoritmo y el programa correspondiente en Visual Basic, para resolver los siguientes problemas:

a) Determinar la cantidad total a pagar por una llamada telefónica, teniendo en cuenta lo siguiente:

- Toda llamada que dure tres minutos o menos tiene un costo de 300 Bs.

- Cada minuto adicional a partir de los tres primeros cuesta 150 Bs.

b) Se quiere un programa que permita calcular el precio a pagar por la compra de un artículo. El IVA a pagar es del 15% y si el precio bruto (precio de venta + IVA) es mayor de 150.000 Bs. se debe realizar un descuento del 1%.

c) Los empleados de una fábrica trabajan en uno de los siguientes turnos: diurno o nocturno. Se desea calcular el salario de un trabajador de acuerdo con lo siguiente:
- La tarifa diurna es de 1.000 Bs./hora
- La tarifa nocturna es de 1.500 Bs./hora

d) Dado un número entero N, determinar si es par.

e) Escribir un programa que clasifique a un entero x en una de las siguientes categorías y muestre un mensaje apropiado
- $x < 5$
- $5 <= x <= 50$
- $x > 50$

f) Hacer un programa que resuelva una ecuación de segundo grado $AX^2 + BX + C = 0$.

g) Calcular el salario total de un trabajador teniendo en cuenta su salario base y el número de horas extra trabajadas. Para el pago de las horas extra se debe considerar la categoría del trabajador, de acuerdo a la siguiente tabla:

Categoría	Precio de la hora extra
A	4000
B	3000
C	2500
D	1000

h) Escribir un programa que permita introducir el número de un mes (1 a 12) y muestre el número de días que tiene ese mes.

i) Dado un valor de X, calcular el valor de Y considerando lo siguiente:

$$Y = \begin{cases} 3X & \text{si} \quad X < 10 \\ X^2 - 3 & \text{si} \quad 10 \leq X < 20 \\ \sqrt{X} & \text{si} \quad X \geq 20 \end{cases}$$

j) Calcular la comisión que le corresponde a un vendedor de acuerdo a las ventas que hizo. Los criterios de cálculo son:

Ventas (Bs.)	Comisión (% sobre las ventas)
< 500000	0.5
[500000 - 1000000)	1
[1000000 – 1500000)	2
≥ 1500000	2.5

k) Una compañía consultora requiere un programa calcule el precio que debe cobrar por una asesoría.

El precio por hora viene dado por el tipo de asesoría, tal como se indica en la siguiente tabla:

Tipo de asesoría	Costo ($/hora)
A	3000
B	2000
C	1000
D	500

En estos momentos, la compañía desea aplicar un descuento especial que depende del número de horas de asesoría:

Número de horas	Descuento
< 5	No hay descuento
[5, 10)	5 %
[10, 15)	8%
>= 15	10%

l) Escribir un programa que tenga como dato de entrada una calificación alfabética A, B, C, D, E, o F, y muestre la correspondiente calificación numérica 20, 17, 14, 10, 5, 0, respectivamente.

m) Un negocio mayorista que vende tablas de madera de ciertas medidas y especie, tiene clasificado a sus clientes en tres tipos: 1 (si paga a tiempo), 2 (si se retrasa con los pagos) y 3 (si es un cliente nuevo). Este negocio necesita un programa que dado el número de tablas que un cliente compra, el precio unitario (precio de una tabla) y el tipo de cliente, calcule el precio total que dicho cliente debe pagar, considerando un descuento. Si el cliente es tipo 1 el descuento es del 15%, si es tipo 2 tiene un descuento del 5%, y si es tipo 3 tiene un descuento del 2%.

4. Estructuras de Repetición

Las estructuras de repetición permiten ejecutar un conjunto de instrucciones varias veces, tantas como sea necesario. También se conocen como "bucles" o "lazos".

En general, existen tres tipos de repetición: Para, Mientras y Hasta; en las próximas secciones se explica la lógica de cada una de estas estructuras y la forma de utilizarlas en Visual Basic.

4.1 Repetir Para

Se utiliza cuando de antemano es posible conocer el número exacto de repeticiones, el cual puede ser un dato de entrada o un valor dado en el planteamiento del problema.

En la figura 4.1 se muestra el diagrama de flujo correspondiente a una estructura "Repetir Para".

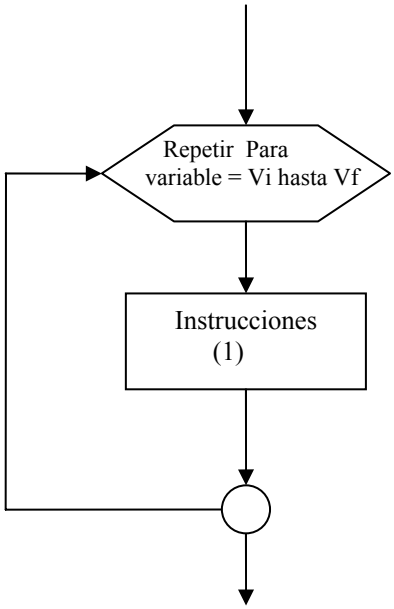


Figura 4.1
Diagrama de flujo de una estructura "Repetir Para"

La estructura "Repetir Para" ejecuta las instrucciones (1), un número determinado de veces. El número de repeticiones se especifica mediante una variable (índice), a la cual se le da un valor inicial (Vi) y un valor final (Vf). Así por ejemplo, si se quieren ejecutar 100 veces las instrucciones (1), el valor inicial es 1 y el valor final 100.

Cada vez que se ejecutan las instrucciones que están dentro de la estructura de repetición, la variable índice se incrementa en uno, hasta que su valor sea mayor al valor final, momento en el cual se deja de ejecutar el bucle. El incremento de la variable índice siempre es 1, a menos que se indique otra cosa.

La sintaxis de una estructura "Repetir Para" en Visual Basic es la siguiente:

```
For  variable = Valor_Inicial to Valor_Final  Step  Incremento
        Una o más instrucciones (1)
Next    variable
```

La instrucción *For* ejecutará las instrucciones (1) X veces, siendo X =Valor_final– Valor_inicial +1.

Step Incremento, permite especificar si se requiere un incremento diferente de uno. Si el incremento es 1, no es necesario escribir *Step*.

Los ejemplos 1 y 2 muestran dos programas muy sencillos que permiten comprender la sintaxis y el funcionamiento de la estructura For.

Ejemplo 1: programa que muestra 5 veces un mismo mensaje.

```
Option Explicit
Sub Main()
Dim i As Integer

For i = 1 To 5
      MsgBox ("Hola ¿Cómo estas?")
Next i
End Sub
```

Ejemplo 2: programa que escribe los números pares comprendidos entre 2 y 10 (ambos inclusive).

```
Option explicit
Sub Main ( )
Dim num as integer

For num = 2 to 10  step 2
     MsgBox(num)
Next num
End Sub
```

El siguiente ejemplo muestra un programa en el cual se utiliza una estructura de repetición para introducir los datos de entrada y calcular un resultado.

Ejemplo 3: programa que calcula el promedio general de un curso de n estudiantes.

```
Option Explicit
Sub Main()
Dim nota As Single, suma As Single, i As Integer
Dim n As Integer, promedio As Single

suma = 0
```

```
n = InputBox ("Introduzca el número de estudiantes a procesar:")

For i = 1 To n
    nota = InputBox("Nota del estudiante " & i)
    suma = suma + nota
Next i
promedio = suma / n
MsgBox("Promedio = " & promedio)

End Sub
```

La variable suma de este ejemplo es un **acumulador**. Una variable acumuladora se utiliza cuando se quiere sumar valores sucesivos de una variable, dentro de una estructura de repetición. Generalmente se inicializa en cero.

Ejemplo 4: programa que calcula el valor de la serie $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+.....+\frac{1}{n}$, donde n es un número especificado por el usuario.

```
Option Explicit
Sub Main()
Dim n As Integer
Dim i As Integer, suma As Single

n = InputBox ("Valor de n:")
suma = 0
For i = 1 To n
  suma = suma + 1 / i
Next i

MsgBox ("suma = " & suma)

End Sub
```

4.2 Repetir Mientras

Este tipo de estructura de programación, repite la ejecución de un conjunto de instrucciones mientras que una expresión lógica es verdadera. Si la expresión lógica es falsa, no se repiten las instrucciones.

La estructura "Repetir Mientras" es adecuada cuando no se conoce de antemano el número de repeticiones. Por ejemplo, supóngase que se desea un programa que calcule el diámetro promedio de los árboles de una plantación, pero no se sabe exactamente cuántos árboles son. En casos como estos no es posible usar una estructura "Repetir Para".

También se puede utilizar un "Repetir Mientras" cuando se conoce el número de repeticiones o puede obtenerse como un dato de entrada, siendo una alternativa al "Repetir Para".

En la figura 4.2 se muestra el diagrama de flujo correspondiente a una estructura "Repetir Mientras".

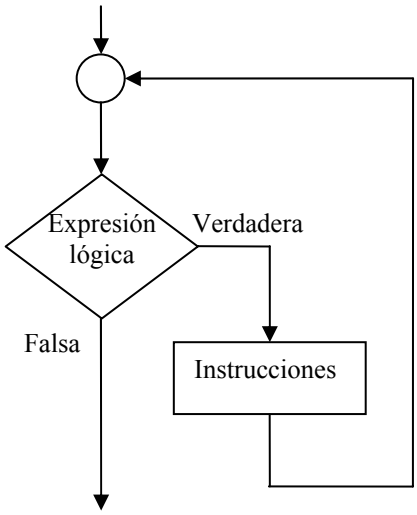


Figura 4.2.
Diagrama de flujo de una estructura "Repetir Mientras"

Visual Basic proporciona tres formas distintas para definir una estructura "Repetir Mientras", la sintaxis de cada una de ellas se indica a continuación.

a) Forma 1: expresión lógica al comienzo

```
Do While  expresión lógica

     Una o más instrucciones (1)

Loop
```

Esta es la manera tradicional de utilizar un "Repetir Mientras"; se evalúa inicialmente una expresión lógica, y si ésta es verdadera se ejecuta el conjunto de instrucciones (1), si es falsa no se ejecuta. El diagrama de flujo de la figura 4.5 corresponde a la forma 1 de "Repetir Mientras" en Visual Basic.

b) Forma 2: expresión lógica al final

```
Do
    Una o más instrucciones (1)

Loop While  expresión lógica
```

La forma 2 es una variante del "Repetir Mientras", aquí la expresión lógica se evalúa al final de la estructura de repetición, por lo tanto, las instrucciones dentro del bucle siempre se ejecutarán al menos una vez. Después de la primera ejecución del bucle, se evalúa la expresión lógica y si ésta es verdadera se repite la ejecución de las instrucciones correspondientes, si es falsa no se repite.

c) Forma 3

```
While   expresión lógica

     Una o más instrucciones (1)

Wend
```

Esta manera de escribir un "Repetir Mientras" tiene la misma lógica de la forma 1, es simplemente otra alternativa.

Los siguientes ejemplos hacen uso de la forma 1 de la estructura "Repetir Mientras". Sin embargo, es fácil resolverlos usando cualquiera de las otras dos formas

Ejemplo 5: programa que calcula el promedio general de un curso de n estudiantes. Éste es igual al ejemplo 3 pero en su solución se utiliza "Repetir Mientras".

```
Option Explicit
Sub Main()
Dim nota As Single, suma As Single, i As Integer
Dim n As Integer, promedio As Single

suma = 0
n = InputBox ("Introduzca el número de estudiantes a procesar:")
i=1
Do While i<= n
    nota = InputBox("Nota del estudiante " & i)
    suma = suma + nota
    i=i+1
Loop
promedio = suma / n
MsgBox( "Promedio = " & promedio)

End Sub
```

Ejemplo 6: programa que recibe como datos de entrada las edades de los 50 empleados de una empresa y calcula: a) el número de empleados que tienen menos de 30 años, b) el número de empleados que tienen entre 30 y 50 años, c) el número de empleados que tienen más de 50 años, y d) la edad promedio de los empleados.

```
Option Explicit
Sub Main()
Dim i As Integer, edad As Byte, e1 As Byte, e2 As Byte, e3 As Byte
```

```
Dim prom As Single, sumaedad As Integer

i = 1
sumaedad = 0
Do While i <= 5
   edad = InputBox("Edad " & i)
   sumaedad = sumaedad + edad
   If edad < 30 Then
     e1 = e1 + 1
   Else
     If edad <= 50 Then
       e2 = e2 + 1
     Else
       e3 = e3 + 1
     End If
   End If
   i = i + 1
   Loop
prom = sumaedad / 5
MsgBox ("Promedio = " & prom)
MsgBox ("Empleados con menos de 30 años= " & e1)
MsgBox ("Empleados con que tienen entre 30 y 50 años " & e2)
MsgBox ("Empleados con más de 50 años= " & e3)

End Sub
```

Las variables e1, e2 y e3 son contadores. Un **contador** es una variable que se utiliza para determinar la cantidad de elementos que cumplen cierta condición. En este ejemplo, las variables e1, e2 y e3 almacenan la cantidad de empleados que tienen menos de 30 años, entre 30 y 50 años, y más de 50 años, respectivamente.

Ejemplo 7. programa que recibe como entrada el volumen de los árboles de una plantación y calcula el volumen total, asumiendo que se desconoce cuantos árboles se van a procesar. Además del volumen total, el programa muestra como salida el no. de árboles que fueron procesados.

Este ejemplo corresponde a una situación donde el número de repeticiones es desconocido. Una forma de resolverlo es usar un valor centinela, esto es, cuando el usuario no desee introducir más datos escribirá como volumen el valor -1 y el programa debe dejar de recibir más datos de entrada y mostrar la salida. En este caso el valor centinela es -1, podría usarse otro valor, siempre y cuando no sea un valor posible para el volumen.

```
Option explicit
Sub Main()
Dim na As Integer     'número de árboles
Dim vol As Single
Dim sumavol As Single  'acumulador utilizado para sumar los volúmenes
na = 0
vol = 0
Do While vol <> -1
  vol = InputBox("Volumen " &(na + 1))
  If vol <> -1 Then
  sumavol = sumavol + vol
```

```
  na = na + 1
  End If
Loop
MsgBox ("Volumen total= " + & sumavol)
MsgBox ("Número de árboles procesados= " & na)

End Sub
```

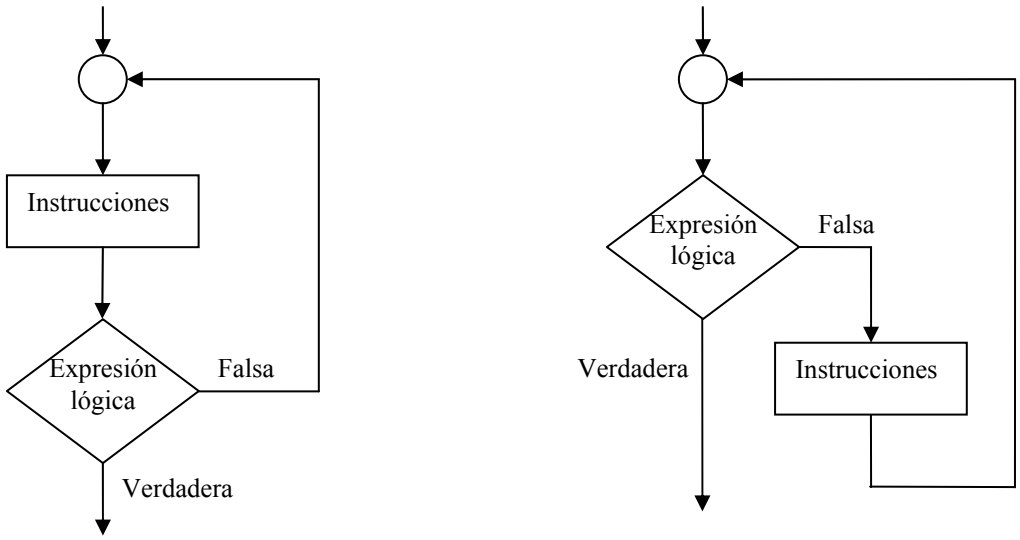
4.3 Repetir Hasta

Esta estructura permite repetir un conjunto de instrucciones hasta que una expresión lógica sea verdadera, o lo que es igual, repite mientras una expresión lógica es falsa. Cuando la expresión lógica es verdadera, el bucle deja de ejecutarse y el programa continúa en la instrucción siguiente.

Un "Repetir Hasta" se puede utilizar en los mismos casos que se usa "Repetir Mientras". La lógica del problema indicará cuál de las dos estructuras es más conveniente emplear, o si es indiferente utilizar cualquiera de ellas.

La expresión lógica que determina la ejecución de un "Repetir Hasta" puede ir al comienzo o al final de la estructura. En la figura 4.3 se muestra el diagrama de flujo correspondiente a las dos situaciones.

Figura 4.3. Diagrama de flujo de una estructura "Repetir Hasta"



a) Expresión lógica al final **b) Expresión lógica al comienzo**

La sintaxis en Visual Basic de un "Repetir Hasta"es la siguiente:

a) Expresión lógica al final

```
Do
    Una o más instrucciones (1)
Loop Until expresión lógica
```

Como la expresión lógica se evalúa al final de la estructura de repetición, las instrucciones (1) siempre se ejecutarán al menos una vez. Después de la primera ejecución del bucle, se evalúa la expresión lógica y si ésta es falsa se repite la ejecución de las instrucciones, si es verdadera no se repite.

b) Expresión lógica al comienzo

```
Do Until  expresión lógica
    Una o más instrucciones (1)
Loop
```

Al comienzo del bucle se evalúa la expresión lógica y si ésta es falsa se ejecutan las instrucciones (1), si es verdadera no se ejecuta el "Repetir Hasta".

Ejemplo 8: programa que calcula el promedio general de un curso de n estudiantes. Éste es el mismo programa de los ejemplos 3 y 5, pero resuelto con la estructura "Repetir Hasta".

```
Option Explicit

Sub Main()
Dim nota As Single, suma As Single, i As Integer
Dim n As Integer, promedio As Single

suma = 0
n = InputBox ("Introduzca el número de estudiantes a procesar:")
i=1
Do Until i> n
    nota = InputBox("Nota del estudiante " & i)
    suma = suma + nota
    i=i+1
Loop
promedio = suma / n
MsgBox ("Promedio = " & promedio)

End Sub
```

Ejemplo 9: Programa que permite obtener la siguiente información acerca de los estudiantes de una Facultad
- Número de estudiantes del sexo masculino
- Número de estudiantes del sexo femenino
- Edad promedio
- Promedio de notas general de la Facultad

Para resolver este problema utilizando una estructura de repetición, se requiere como primer dato de entrada el número de estudiantes de la Facultad. Luego, para cada estudiante se solicitan los siguientes datos: sexo, edad y promedio de notas.

```
Option Explicit
Sub Main()
Dim n As Integer, sexo As String, edad As Byte, prom As Single
Dim i As Integer, nm As Integer, nf As Integer, sumae As Single
Dim sumap As Single, prome As Single, promg As Single

n = InputBox("Número de estudiantes")
i = 1
nm = 0
Do Until i > n
  sexo = InputBox("Sexo:", "Estudiante " & i)
  edad = InputBox("Edad:", "Estudiante " & i)
  prom = InputBox("Promedio:", "Estudiante " & i)
  If sexo = "M" Or sexo = "m" Then
     nm = nm + 1
  Else
    If sexo = "F" Or sexo = "f" Then
    nf = nf + 1
    End If
  End If
  sumae = sumae + edad
  sumap = sumap + prom
  i = i + 1
Loop
prome = sumae / n
promg = sumap / n
MsgBox("Edad Promedio=" & prome & Chr(10) & "Promedio General= " & promg)
MsgBox ("No. de hombres= " & nm & Chr(10) & "No. de mujeres= " & nf)

End Sub
```

Este programa utiliza variables contadoras para determinar el número de estudiantes del sexo masculino (nm) y el número de estudiantes del sexo femenino (nf). Se asume que el sexo masculino debe ser ingresado al programa con una letra M (mayúscula o minúscula), y el sexo femenino con una F (mayúscula o minúscula).

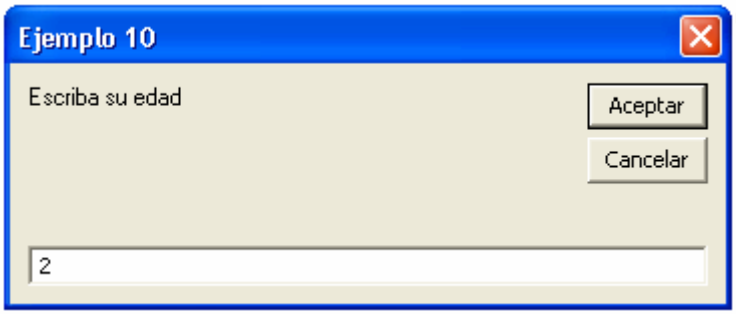
Además se usaron acumuladores (sumae y sumap) para representar la sumatoria de las edades y la sumatoria de los promedios de notas.

Ejemplo 10: este programa tiene un conjunto de instrucciones que permiten validar la entrada de datos haciendo uso de una estructura "Repetir Hasta". En el ejemplo el usuario debe introducir la edad, pero se requiere que la edad esté entre 5 y 100, es decir que una edad menor que 5 se considera inválida, al igual que una edad superior a 100. El objetivo es que el programa detecte el error y no acepte las edades que son inválidas.

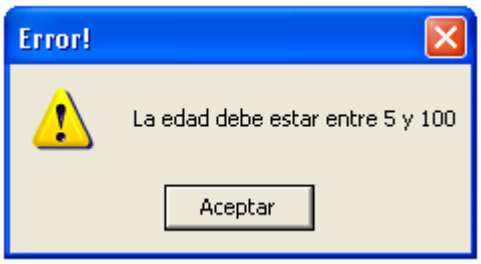
```
Option Explicit
Sub main()
Dim edad As Byte, Resp As Integer

Do
edad = InputBox("Escriba su edad")

If edad < 5 Or edad > 100 Then
  Resp = MsgBox("La edad debe estar entre 5 y 100", vbExclamation, "Error!")
End If
Loop Until edad >= 5 And edad <= 100

End Sub
```

El programa funciona de la siguiente manera:

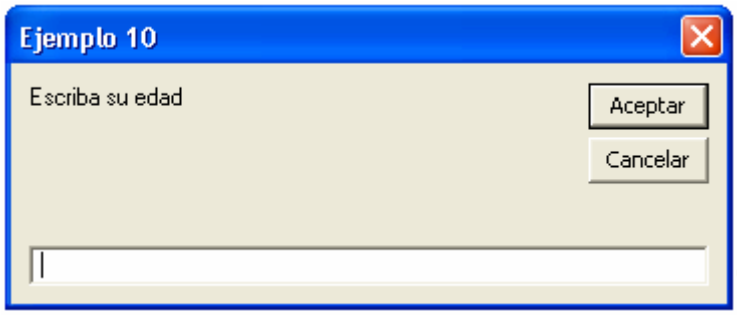
1. Muestra un InputBox solicitando la edad



2. Si la edad no se encuentra dentro del rango especificado muestra un mensaje de error



3. Luego vuelve a mostrar el InputBox, para que se introduzca un nuevo valor. Esto lo repite hasta que el valor introducido para la edad sea válido.



4. Si la edad introducida se considera válida (entre 5 y 100), entonces se acepta el valor escrito por el usuario en el InputBox y se asigna a la variable edad.

Ejemplo 11: programa que calcula la suma y la media aritmética de un conjunto de números. Cada vez que se introduce un número se muestra un MsgBox donde se pregunta si se desea introducir un nuevo número, si la respuesta es sí, aparece un InputBox para escribir el siguiente número, si es no, entonces se muestran la suma, la media aritmética y la cantidad de números procesados.

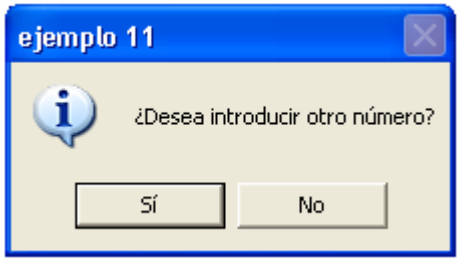
```
Option Explicit
Sub main()
Dim num As Single, suma As Single, media As Single, can_num As Integer
Dim resp As Byte

suma = 0
can_num = 0
Do
num = InputBox("Escriba un número:")
suma = suma + num
can_num = can_num + 1
resp =MsgBox("¿Desea introducir otro número?", vbYesNo + vbInformation, _
"ejemplo 11")
Loop Until resp = 7

media = suma / can_num
MsgBox ("Números procesados = " &  can_num  &  Chr(10) & "Suma = " & _
suma &  Chr(10) & "Media = " & media)
End Sub
```

Nota: cuando una instrucción no cabe en una sola línea, se coloca al final de ésta el símbolo _, y se continúa en la siguiente línea.

Obsérvese que la instrucción

```
resp =MsgBox("¿Desea introducir otro número?", vbYesNo + vbInformation, _
"ejemplo 11")
```

muestra el siguiente cuadro de mensaje:



Si el usuario hace clic en el botón Sí, la variable resp toma el valor 6 (ver cuadro 2.4 del capítulo 2). Si hace clic en el botón No resp vale 7, en ese caso la expresión lógica de la estructura Do – Loop – Until es verdadera, por lo tanto se deja de ejecutar el bucle e inmediatamente se muestran los resultados en un Msgox.

4.4 Ejercicios propuestos

Hacer un algoritmo y el programa correspondiente en Visual Basic, para resolver los siguientes problemas:

1. Hacer de tres maneras diferentes: usando "Repetir Para", "Repetir Mientras" y "Repetir Hasta".
   a) Obtener la suma de n números introducidos por el usuario.
   b) Calcular la suma de los cuadrados de los 50 primeros números naturales (enteros positivos).

2. Calcular independientemente la suma de los números pares e impares comprendidos entre 1 y 50.

3. Se tienen las calificaciones de los alumnos de un curso de informática, el cual consta de tres materias: Visual Basic, Excel y Word. Calcular la nota definitiva de cada alumno, número de estudiantes aplazados, número de estudiantes aprobados y el promedio general del curso.

5. Se tienen como datos de entrada los salarios de 50 empleados de una empresa. Hacer un programa para determinar: a) el número de empleados que ganan menos de 350000 Bs., b) el número de empleados que ganan entre 350000 y 600000 Bs. c) el número de empleados que ganan más de 600000 Bs. d) El valor de la nómina (suma de todos los salarios).

6. Escribir un programa que calcule la suma:

$1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\frac{1}{5^2}+.....+\frac{1}{n^2}$ donde n es un número especificado por el usuario

7. En un centro meteorológico se llevan las precipitaciones mensuales caídas en tres zonas del país: Occidente, Centro y Oriente. Se desea un programa que reciba como datos de entrada las precipitaciones registradas en los 12 meses de un año para cada región y determine: a) precipitación anual en cada región y b) región con mayor precipitación anual.

8. Se quiere un programa que contabilice una cuenta de ahorros. Al inicio se le introduce el nombre del titular de la cuenta y el saldo inicial. A continuación se permite hacer depósitos y retiros sucesivos, el usuario debe escribir una "d" si desea depositar o una "r" si desea retirar. Cuando es depósito se incrementa al saldo y cuando es retiro se resta, luego de cada operación debe mostrarse el saldo. El programa finalizará cuando ya no se desee hacer más movimientos. Al terminar, el programa debe mostrar el saldo final.
Sugerencia: utilizar una estructura de repetición con una condición evaluada al final del bucle.

9. Calcular el promedio de un estudiante que presenta tres exámenes. El programa debe validar los datos de entrada, es decir, sólo debe aceptar notas entre 0 y 20.

5. Uso de formularios y controles

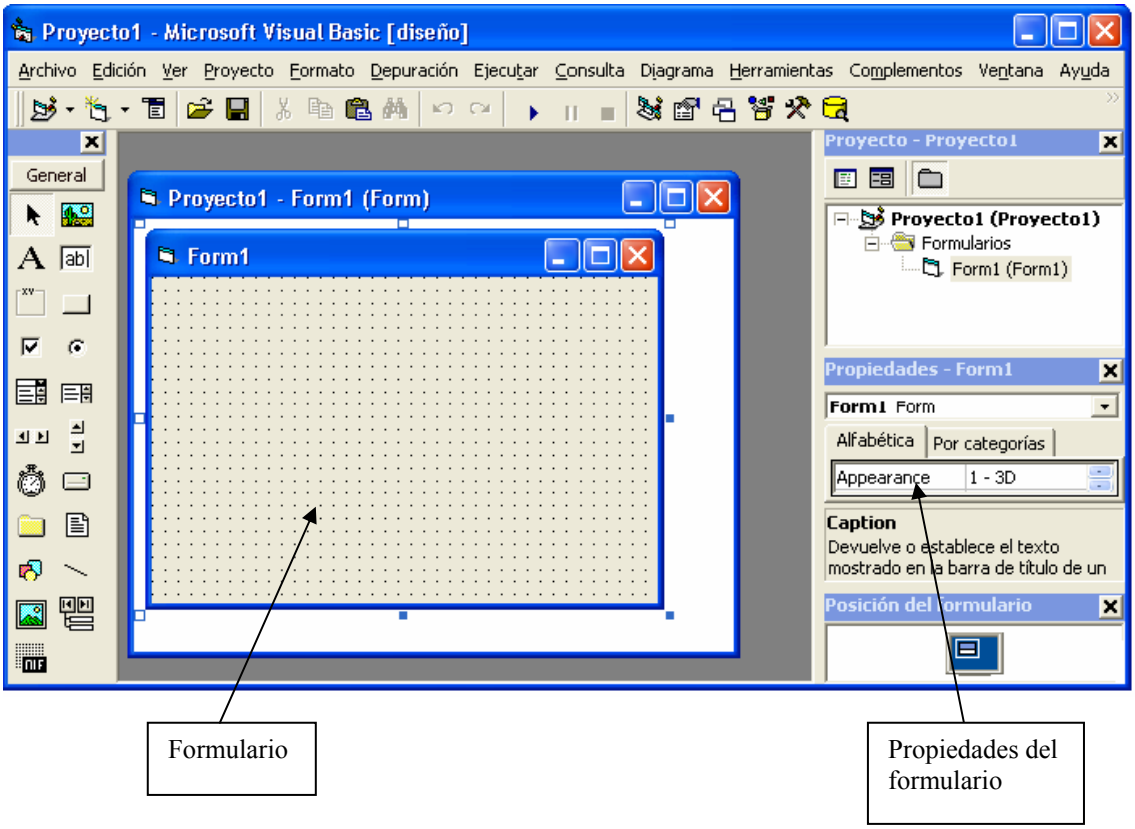
Hasta ahora no se han utilizado las capacidades que tiene Visual Basic para crear programas con una interfaz gráfica propia del ambiente Windows, que incluya botones de comando, cuadros de texto, cuadros de diálogo, botones de opción y de selección, listas desplegables, etc. En los capítulos anteriores los programas se han limitado al uso de InputBox y MsgBox para la entrada y salida de datos, pues se hace énfasis en lógica de programación. En este capítulo se muestran otras formas de gestionar las entradas y salidas, introduciendo la programación por eventos.

5.1. Formularios (Form)

Un formulario es una ventana donde se colocan todos los controles del programa (botones de comando, cuadros de texto, etiquetas, botones de opción y de selección, listas, imágenes, etc.). Un programa puede tener varios formularios, pero en programas sencillos un solo formulario debería ser suficiente.

Cuando se inicia Visual Basic se abre en pantalla un formulario, que tiene por defecto el nombre de Form1. En la figura 5.1 se observa la ventana inicial de Visual Basic; en el centro se encuentra el formulario.

Figura 5.1. Ventana inicial de Visual Basic.



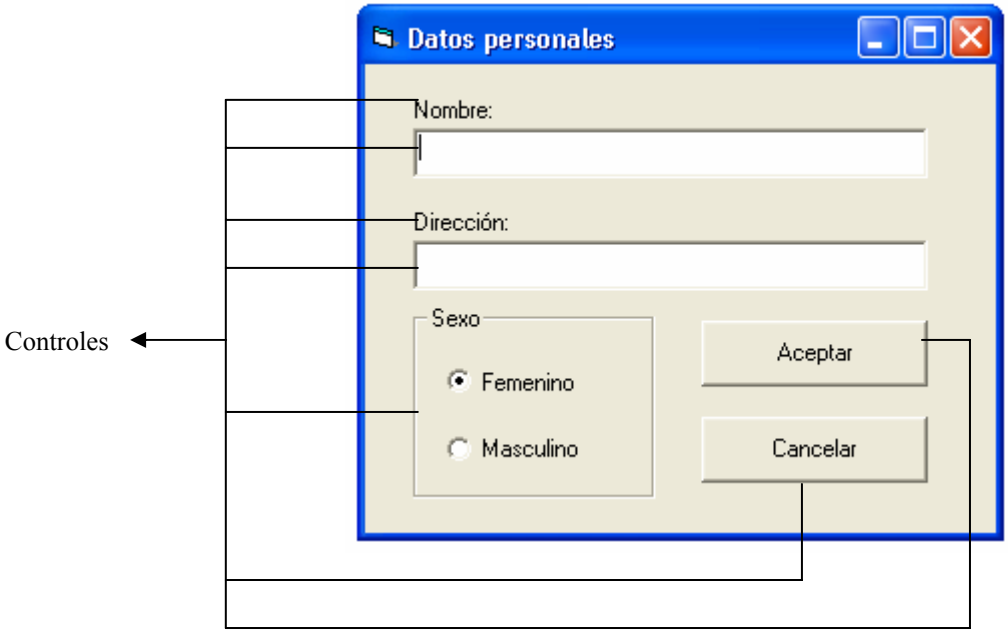
Este formulario (Form1) será la interfaz gráfica del programa, a través de la cual se solicitarán los datos necesarios y se escribirán los resultados.

Una vez que se abre la ventana inicial de Visual Basic para comenzar a construir un programa, el primer paso es colocar en el formulario los controles necesarios para que el usuario introduzca los datos.

5.1.1 Ejemplo de un formulario

En la figura 5.2 se muestra un ejemplo de un formulario donde se piden algunos datos personales. El programador que creó este formulario, lo hizo agregando controles al formulario que aparece en la ventana inicial de Visual Basic (figura 5.1).

Figura 5.2. Ejemplo de formulario



5.1.2 Algunas propiedades de los formularios

Un formulario tiene propiedades que definen su apariencia (forma, tamaño, título, color, etc.) y su forma de responder a las acciones del usuario. Si se hace clic en las barras de desplazamiento del recuadro *propiedades* que aparece en la ventana de Visual Basic (ver figura 5.1), se puede observar una lista de todas las propiedades de un formulario y además se puede cambiar el valor de las mismas.

Algunas propiedades son:

Nombre: permite identificar al formulario. Se introduce en tiempo de diseño y no se puede variar durante la ejecución. Nombre por defecto: Form1 (Form2 y sucesivos, si hay más de un formulario).

Caption: título del formulario. Es el texto que se mostrará en la barra de título cada vez que aparezca en pantalla el formulario, no tiene otra función dentro del programa. El programa no accede a un formulario por el título sino por el nombre.

No debe confundirse el Nombre con el Título (*Caption*). Por ejemplo, en el formulario de la figura 5.2, en la propiedad *Caption* se escribió: Datos Personales; y en la propiedad *Nombre* se le colocó formdatos.

MinButton: Es una propiedad booleana que admite el valor *True* o *False*. Si está en *True*, aparecerá el botón de minimizar el formulario  , si está en *False*, no aparecerá. Debe configurarse de una u otra forma, dependiendo si se quiere o no minimizar el formulario durante la ejecución. Valor por defecto: *True*.

MaxButton: funciona en forma similar a la propiedad MinButton pero con el botón maximizar  .

Backcolor: Establece el color de fondo del formulario.

Picture: permite seleccionar una imagen para que aparezca como fondo del formulario.

Se puede consultar el *Help* de Visual Basic para conocer las demás propiedades que tienen los formularios.

5.2. Controles

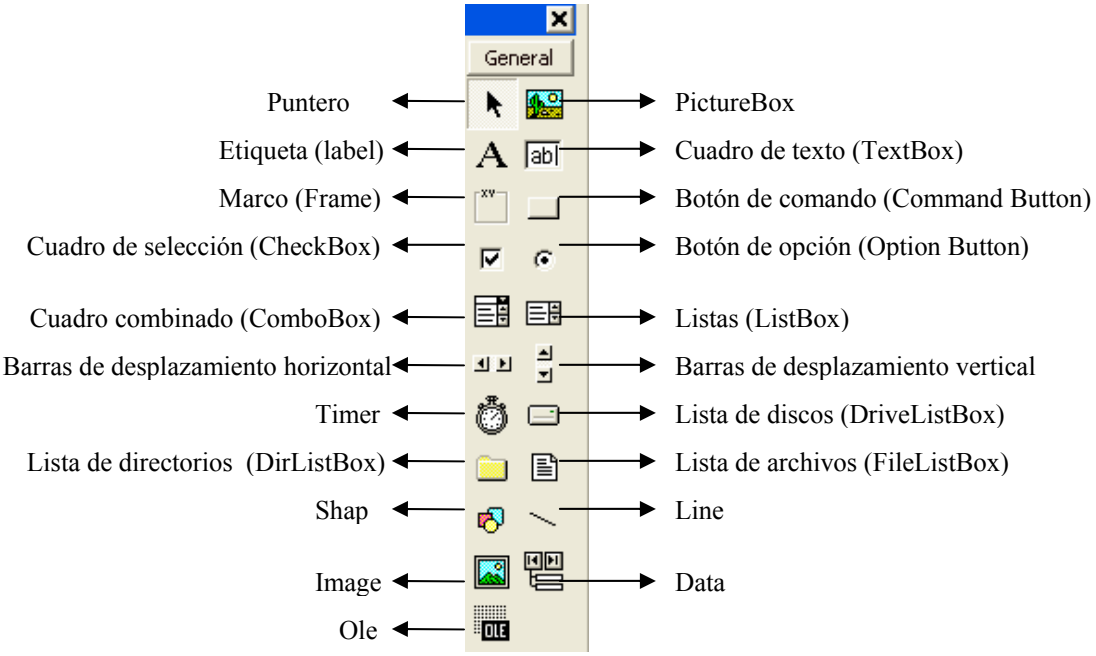
Un control es un elemento gráfico que puede incluirse en un formulario, con la finalidad de mostrar información o aceptar los datos introducidos por el usuario. En el formulario de la figura 5.2 se pueden observar algunos controles.

Los controles al igual que los formularios, tienen propiedades que definen sus características. Las propiedades dependerán del tipo de control que se utilice.

Para colocar un control en un formulario, basta con hacer clic en el cuadro de herramientas que se encuentra en la parte izquierda de la ventana de Visual Basic, específicamente sobre el control que se quiere agregar. Luego se coloca el cursor del ratón en el sitio del formulario donde se va a insertar el control y se arrastra el ratón para especificar el tamaño deseado.

En la figura 5.3 se muestra el cuadro de herramientas con los nombres de los diferentes controles.

Figura 5.3. Cuadro de herramientas

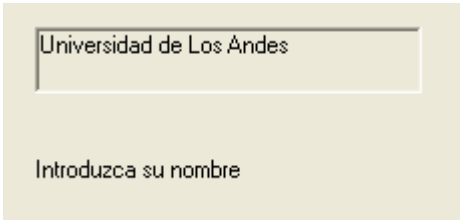


En las siguientes secciones se explican los controles básicos.

5.2.1 Etiquetas (Label)

Estos controles permiten mostrar mensajes en un formulario. Con frecuencia se utilizan etiquetas para colocar títulos y para escribir los resultados generados por el programa. En la figura 5.4 se muestran dos etiquetas.

Figura 5.4. Etiquetas.



Las principales propiedades de las etiquetas son:

Nombre: permite identificar la etiqueta y acceder a ésta mediante código.
Para facilitar la lectura y análisis de los programas escritos en Visual Basic, se acostumbra colocar prefijos a los nombres de los controles (de acuerdo a su tipo). En el caso de la etiquetas se usa el prefijo lbl. Un ejemplo de nombre de una etiqueta puede ser: lbltitulo.

Caption: es la propiedad más importante ya que contiene el texto que aparece sobre la etiqueta. Esta propiedad puede cambiarse en la ventana propiedades y en el código del programa.

Alignment: permite alinear el texto de la etiqueta a la izquierda, a la derecha o centrarlo.

Font: se refiere al tipo, tamaño y estilo de letra del texto de la etiqueta.

Forecolor: permite definir el color del texto de la etiqueta.

BackColor: especifica el color de fondo.

BorderStyle: si esta propiedad vale 0 la etiqueta no tiene borde, y si es igual a 1 presenta un borde como el de la primera etiqueta de la figura 5.4.

Autosize: cuando está en *True* ajusta el tamaño de la etiqueta al del texto.

WordWrap: si vale *True* hace que el texto se distribuya en varias líneas cuando no cabe en una sola.

5.2.2 Cuadros de texto (TextBox)

Los cuadros de texto reciben la información que el usuario introduce. Aunque hay otros controles que captan los datos de entrada, los cuadros de texto son la forma más sencilla de hacerlo. También se puede utilizar un TextBox para mostrar resultados del programa, en este caso es importante tener presente que el usuario puede modificar lo que está escrito en el cuadro de texto, incluso si es un valor generado por el programa.

Figura 5.5. Cuadros de texto.

Frecuentemente se utilizan etiquetas para colocar mensajes antes de los cuadros de texto, con la finalidad de que los usuarios sepan qué deben escribir en ellos.

Algunas propiedades de los cuadros de texto son:

Nombre: permite identificar al cuadro de texto. Se acostumbra anteponer las letras txt al nombre de un TextBox.

Text: es la propiedad más importante de los TextBox, ya que almacena el texto que aparece en el cuadro.

MultiLine: si su valor es True, el cuadro de texto puede tener varias líneas. Si está en False el texto se maneja en una sola línea.

ScrollBars: permite agregar barras de desplazamiento horizontales y/o verticales para que el usuario pueda moverse por las líneas del texto. Para observar estas barras la propiedad Multiline debe tener el valor True.

Un cuadro de texto también tiene las propiedades font, forecolor y backcolor, las cuales funcionan de la misma forma que en las etiquetas.

5.2.3 Botones de comando (Command Button)

Estos botones determinan el momento en que el usuario desea hacer algo, como salir del programa, hacer algún cálculo, comenzar a imprimir, etc. Aparecen en casi todas las ventanas de las aplicaciones de Windows.

En la siguiente figura pueden verse dos botones de comando. El tamaño puede cambiarse a voluntad, pero la forma siempre es rectangular.

Figura 5.6. Botones de comando.

El botón de comando tiene más de 30 propiedades, seguidamente se mencionan las más importantes:

Nombre: permite hacer referencia al botón. Se acostumbra colocar el prefijo cmd a los nombres de los botones de comando.

Caption: define el texto que aparecerá en el botón. Por ejemplo, en el primer botón de la figura 5.4 la propiedad *Caption* está definida como Aceptar.

Enabled: cuando está en False el botón no puede ser pulsado y se presenta en forma atenuada (no está activado el botón).

Style: si su valor es 0 (Standard) el botón tendrá la forma tradicional (ver figura 5.6). Si su valor es 1 (Graphical) el botón podrá tener algún color o una imagen.

Backcolor: especifica el color de fondo. Para que el botón aparezca con color, éste debe ser tipo gráfico (Style=1).

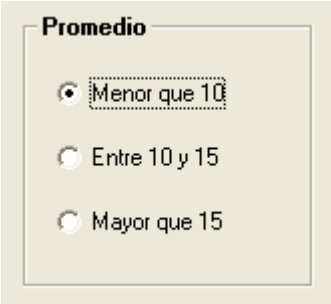
Picture: permite seleccionar una imagen para que aparezca en el botón. Para poder observar la imagen, la propiedad Style debe ser igual a 1.

5.2.4 Botones de opción (OptionButton)

Un botón de opción permite al usuario realizar una selección. Este control muestra una opción que se puede activar o desactivar. La opción que se activa o selecciona se

presentará con un punto negro que la distingue. En la siguiente figura se muestra un ejemplo de botones de opción.

Figura 5.7. Botones de Opción.



Generalmente los controles OptionButton se utilizan en grupo, para mostrar opciones de las cuales el usuario sólo puede seleccionar una. Pueden agruparse colocándolos dentro de un control Frame (Marco)  .

Para agrupar botones de opción se coloca en primer lugar el Frame y luego se sitúan los botones, tantos como se desee. Todos los controles OptionButton que están dentro de un marco actúan como un solo grupo. En un mismo formulario se pueden colocar cuantos grupos de botones de opción se quiera, cada uno de ellos dentro de un marco. Si no se coloca un Frame todos los botones de un mismo formulario forman un único grupo.

En el ejemplo de la figura 5.7, se observa un grupo de botones de opción, en el cual sólo uno puede ser seleccionado.

Algunas de las propiedades más utilizadas de los botones de opción son:

Nombre: define un identificador para el botón de opción. Comúnmente se empieza el nombre de un OptionButton con las letras opt.

Caption: texto que aparece al lado del botón.

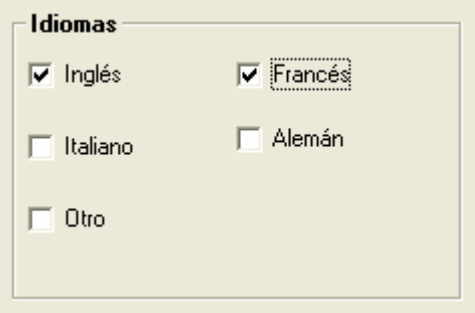
Value: es la propiedad más importante e indica si el botón está seleccionado en tiempo de ejecución. Si Value es igual a True el botón está seleccionado. Si es False, el botón no estará seleccionado.

Además de muchas otras, tiene las propiedades backcolor, font y forecolor antes mencionadas.

5.2.5 Casillas de selección (CheckBox) 

El control CheckBox funciona en forma similar al OptionButton, con la siguiente diferencia: las casillas de selección no son mutuamente excluyentes, es decir, el usuario puede seleccionar una o varias casillas, incluso si se encuentran dentro del mismo marco o formulario. En la figura 5.8 se muestra un ejemplo de casillas de selección dentro de un marco; en el mismo se puede observar que más de una casilla está seleccionada.

Figura 5.8. Casilla de selección



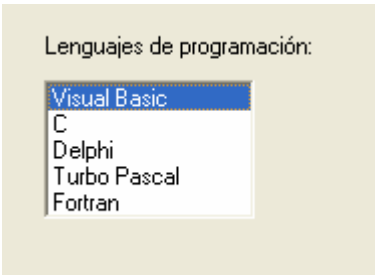
Las casillas de verificación tienen las mismas propiedades básicas de los botones de opción, pero la propiedad Value vale 0 si la casilla no está seleccionada, 1 si está seleccionada o 2 si está atenuada.

El prefijo que se acostumbra utilizar en el nombre de un CheckBox es chk.

5.2.6 Listas (ListBox)

Es un control que permite mostrar una lista de elementos, de la cual el usuario podrá seleccionar uno o varios de ellos. En la figura 5.9 se muestra un ejemplo de una lista, en la que se encuentra seleccionado un elemento.

Figura 5.9. Lista



Si la lista no puede verse completa porque el control no es lo suficientemente grande, aparecerán en forma automática barras de desplazamiento que le permitirán al usuario observar todos los elementos.

Algunas propiedades de las listas son:

Nombre: permite identificar al ListBox. El prefijo que comúnmente se utiliza en los nombres de lista es Lst.

List: se usa para definir el contenido de la lista. Para establecer los elementos de la lista, se escribe un elemento o valor en la propiedad *List* en la ventana propiedades, luego se pulsa la combinación de teclas Ctrl – Enter, y a continuación se escribe el siguiente valor. Cuando no hay más elementos que definir se oprime Enter.

Multiselect: indica las formas de selección. Si es 0-None (valor por defecto), el usuario podrá seleccionar sólo un elemento haciendo clic sobre él. Si es 1-Simple, podrá seleccionar más de un elemento haciendo clic sobre ellos. Si Multiselect tiene el valor 2-Extended la selección múltiple se hará usando las teclas Ctrl y Shift.

Listindex: permite determinar el elemento de la lista que el usuario seleccionó. Los elementos están numerados a partir de 0. Listindex almacena el índice (número) del elemento seleccionado. Por ejemplo, supóngase que en la lista de la figura 5.9 se escoge Fortran, en ese caso la propiedad Listindex toma el valor 4. Si el usuario no selecciona algún elemento de la lista, Listindex vale -1. Cuando el ListBox permite seleccionar varios ítems (Multiselect = 1 o 2), Linstindex guarda el índice del último elemento seleccionado.

ListCount: almacena la cantidad de elementos que tiene la lista.

El control ListBox tiene asociados algunos **métodos** que permiten realizar operaciones específicas mediante código, tales como añadir y eliminar elementos de la lista. Es importante aclarar que un **método** es una función pre-programada que viene con el lenguaje Visual Basic y que ejecuta tareas comunes a distintos programas. Los diferentes tipos de objetos y controles tienen sus propios métodos.

Algunos métodos de las listas son:

AddItem: permite agregar elementos a la lista.

Sintaxis

```
Nombrelista.AddItem  "elemento"
```

Ejemplo: la siguiente instrucción agrega el ítem Visual C++ a la lista de la figura 5.9, cuyo nombre es lstLenguajes.

```
lstLenguajes.AddItem "Visual C++"
```

RemoveItem: se utiliza para eliminar un elemento de la lista.

Sintaxis

```
Nombrelista.RemoveItem  posición_del_elemento_en_la_lista
```

Por ejemplo, para eliminar de lstLenguajes el elemento Turbo Pascal el código correspondiente es:

```
lstLenguajes.RemoveItem 3
```

Clear: elimina todos los elementos de la lista.

Sintaxis

```
Nombrelista.clear
```

Asignación usando elementos de la lista
Si se quiere asignar un valor a un elemento determinado de la lista, mediante código, se utiliza la propiedad List con la siguiente sintaxis:

Sintaxis

```
Nombredelista.List(posición del elemento) = valor
```

Supóngase que para el ejemplo de la figura 5.9, se quiere cambiar a través de código el elemento Visual Basic por Cobol, entonces la instrucción correspondiente es:

```
lstLenguajes.List(0) = "Cobol"
```

También se puede asignar un ítem de la lista a una variable. Por ejemplo, la siguiente instrucción asigna el elemento Delphi (No. 2 de lista de la figura 5.9) a la variable LengProg de tipo String:

```
LengProg = lstLenguajes.List(2)
```

5.2.7 Cuadros combinados (ComboBox) 

Son controles que combinan un cuadro de texto y una lista. Sus propiedades más importantes son:

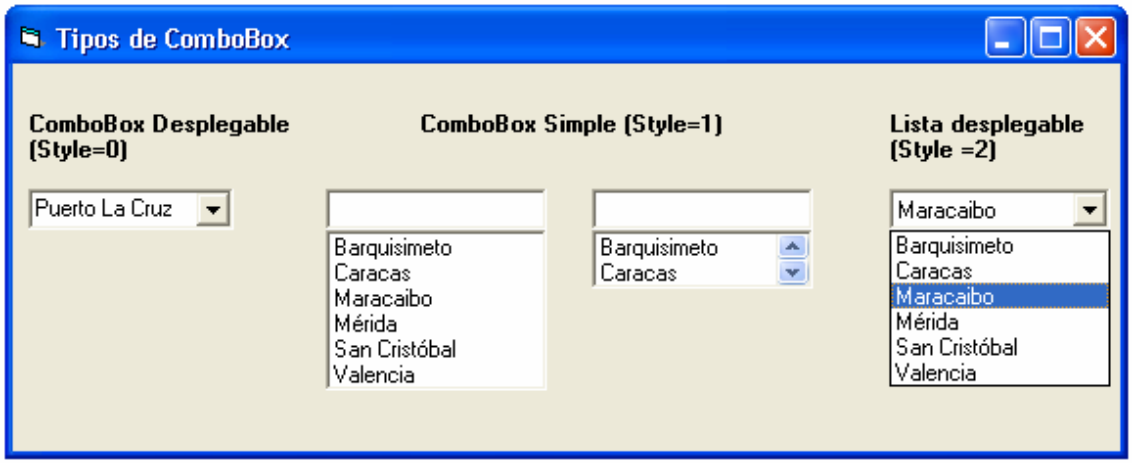
Nombre: se utiliza para identificar el ComboBox. El prefijo que usualmente se coloca en el nombre es cbo.

Style: permite indicar el tipo de cuadro combinado que se desea utilizar. Hay tres tipos:
1. Cuadro combinado desplegable (*Style*=0): ocupa sólo una línea del formulario, para ver los elementos de la lista el usuario debe abrir el cuadro haciendo clic sobre una flecha que está en el lado derecho del ComboBox. El usuario puede seleccionar uno de

los elementos de la lista o puede escribir un nuevo valor en el cuadro de texto disponible.
2. Cuadro combinado simple (*Style*=1): funciona en forma similar a un cuadro combinado despleglable, con la diferencia que en el formulario se pueden observar varias líneas de la lista de elementos.
3. Lista despleglable (*Style*=2): no permite al usuario escribir sobre el cuadro de texto, sólo puede seleccionar un valor de la lista. Se presenta en una línea del formulario y para ver la lista el usuario debe hacer clic en la flecha ubicada al lado derecho del ComboBox.

En la siguiente figura se pueden observar los tres tipos de cuadros combinados disponibles. Nótese que el ComboBox desplegable (*Style=0*) ocupa una línea del formulario y allí el usuario escribió una ciudad que no está en la lista. Obsérvese también que el ComboBox tipo lista (*Style=2*) se encuentra desplegado, esto es porque el usuario hizo clic en la flecha para ver las opciones disponibles, de no ser así se mostraría en una sola línea. Aquí el usuario no puede escribir en el cuadro de texto.

Figura 5.10. Tipos de cuadros combinados (ComboBox)



Text: almacena lo que se encuentra en el cuadro de texto de un ComboBox.

Los cuadros combinados también tienen las propiedades **List**, **ListIndex**, **ListCount** y los métodos **AddItem**, **RemoveItem** y **Clear** explicados en las listas.

5.2.8 Control Image

Este tipo de control se utiliza para colocar imágenes en un formulario. Los tipos de archivos que soporta el control Image son: mapas de bits (.bmp, .dib), imágenes GIF (.gif), imágenes JPEG (.jpg), iconos (.ico, .cur) y metarchivos (.wmf, .emf).

En la figura 5.11 se muestra un formulario en tiempo de diseño con un control Image.

Figura 5.11. Formulario con un control Image

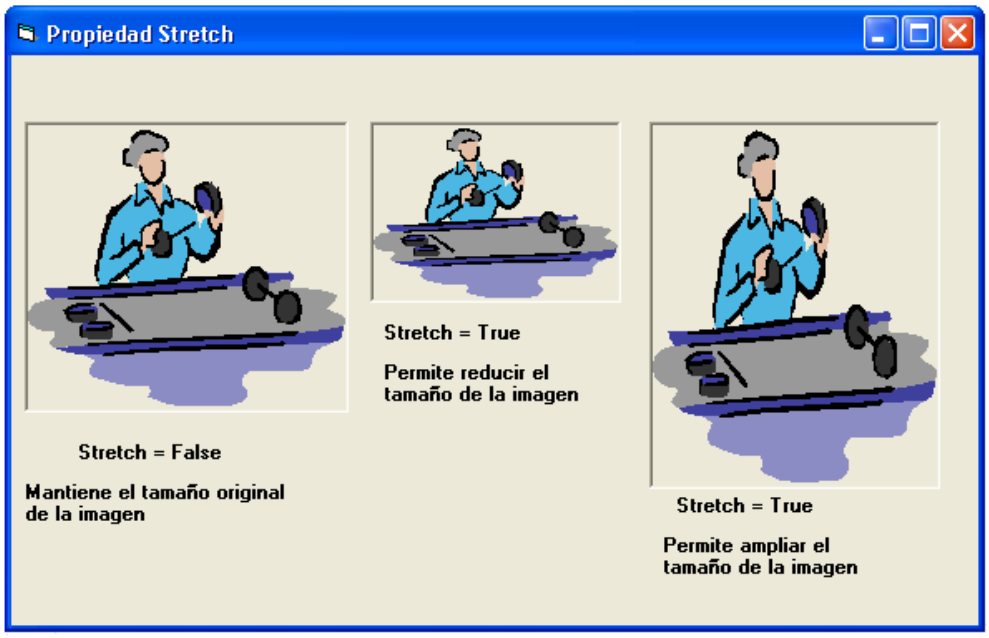
Las propiedades más importantes de un control Image son las siguientes:

Nombre: permite identificar el control Image. Se acostumbra comenzar los nombres de este tipo de control con las letras img.

Picture: se utiliza para seleccionar la imagen que se quiere cargar en el control Image.

Stretch: cuando esta propiedad vale False, el tamaño del control se ajusta al tamaño de la imagen. Si la propiedad Stretch es igual a True, la imagen se ajustará al tamaño del control. En la figura 5.12 se muestra el efecto que tiene esta propiedad.

Figura 5.12. Efecto de la propiedad Stretch del control Image



BorderStyle: si esta propiedad se establece en 1 se observa un borde en el control Image, tal como el que muestran las imágenes de la figura 5.12.

5.2.9 Control PictureBox

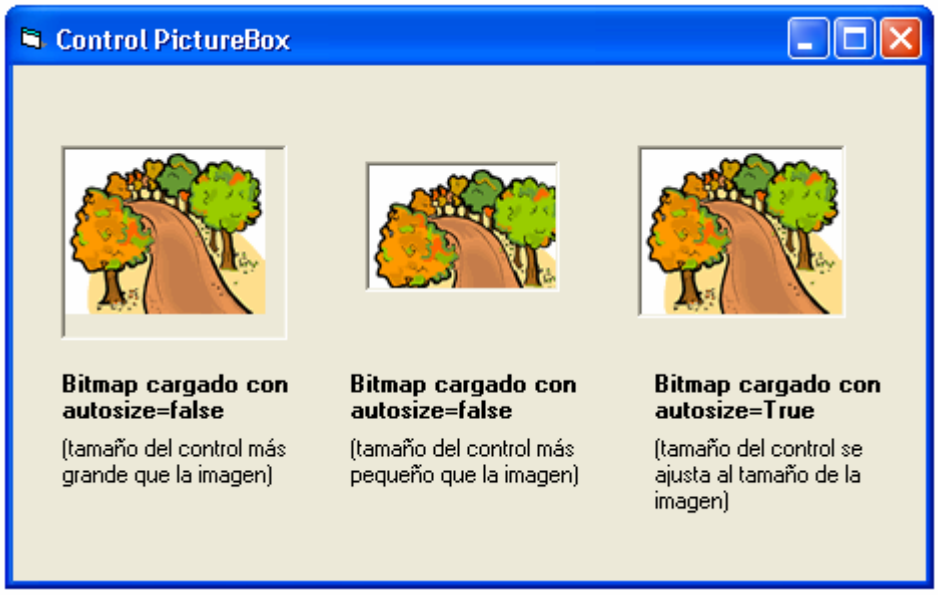
Un PictureBox se utiliza para mostrar imágenes, pero además puede ser un contenedor de otros controles (como el Frame). De esta forma puede utilizarse para agrupar controles mostrando una imagen de fondo.

Este control es más potente que el control Image, tiene más propiedades, eventos y métodos. Sin embargo el PictureBox consume más recursos, siendo más lento que un Image.

Algunas propiedades son: Nombre (se acostumbra colocarle el prefijo pic), Picture y BorderStyle, que trabajan igual que en los controles Image. No tiene la propiedad Stretch, pero tiene la propiedad **Autosize**, la cual determina la forma en que el control responderá al tamaño de la imagen cargada.

Si se carga una imagen y Autosize está en False (valor por defecto), el control no se ajusta al tamaño de la imagen. Si Autosize es igual a True, el control se ajustará. La imagen siempre se mostrará en su tamaño original, ubicándose en la esquina superior izquierda del PictureBox. En la figura 5.12 se muestran algunos ejemplos.

Figura 5.12. Efecto de la propiedad Autosize del control PictureBox



La propiedad Autosize funciona de manera distinta con las imágenes tipo metarchivos. Estas son las únicas imágenes que ajustan su tamaño al del control PictureBox. Si Autosize es True, el tamaño del control se ajusta al tamaño de la imagen, y si está en False la imagen se ajusta al tamaño del control. En la figura 5.13 puede observarse el efecto de la propiedad Autosize cuando se cargan metarchivos.

Figura 5.13. Efecto de la propiedad Autosize del control PictureBox con metarchivos.

5.3. Eventos

Un evento es algo que sucede durante la ejecución de una aplicación, específicamente, son las acciones del usuario sobre el programa. Ejemplos de eventos son hacer clic sobre un botón, pulsar una tecla o una combinación de teclas, hacer doble clic sobre el nombre de un archivo para abrirlo, escribir en un cuadro de texto, elegir una opción de un menú, colocar el ratón en un área determinada de la pantalla, etc.

En la programación por eventos, un programa debe responder a distintos eventos que el usuario genera. Por ejemplo, si se tiene un formulario con dos botones Aceptar y Cancelar, el programa debe tener un código que le diga que hacer cuando el usuario hace clic sobre alguno de ellos. Supóngase que se quiere solicitar nuevos datos si el usuario oprime el botón Aceptar, y si pulsa el botón Cancelar, el programa debe cerrarse; para ejecutar estas acciones es necesario escribir las instrucciones en Visual Basic apropiadas.

El código correspondiente a cada evento se coloca en los llamados ***procedimientos de evento***, que son rutinas o módulos que contienen las instrucciones necesarias para realizar las acciones programadas para un evento concreto. El programa sólo responderá a aquellos eventos que tengan un código asociado. De cierto modo, un programa escrito en Visual Basic, no es más que una colección de pequeñas rutinas.

Cada control que se coloca en un formulario soporta uno o más eventos. Por ejemplo, si se coloca un cuadro de texto en un formulario, tres posibles eventos que pueden ocurrir son hacer clic sobre él, introducir texto o hacer doble clic.

Si se codificara un procedimiento para el evento hacer doble clic, las instrucciones del código se ejecutarán tan pronto el usuario haga doble clic sobre el cuadro de texto.

En el cuadro 5.1 se describen algunos eventos generales con los que se puede trabajar en Visual Basic.

Tabla 5.1. Algunos eventos que soporta Visual Basic.

NOMBRE DEL EVENTO (VISUAL BASIC)	DESCRIPCIÓN
Load	Se produce al cargar un formulario, esto es, al visualizarse un formulario.
Unload	Se genera al cerrar un formulario mediante el botón cerrar.
KeyPress	Sucede cuando el usuario pulsa y suelta determinada tecla.
KeyDown	Se produce cuando el usuario pulsa determinada tecla
KeyUp	Ocurre cuando el usuario suelta una determinada tecla
Click	Se genera cuando el usuario pulsa y suelta uno de los botones del ratón sobre un formulario o control. Para el caso de un botón de comando, botón de opción y casilla de selección, el evento ocurre cuando se pulsa el botón izquierdo del ratón.
DblClick	Sucede cuando se hace clic dos veces seguidas con el botón izquierdo del ratón sobre un formulario o control
MouseDown	Se genera cuando el usuario pulsa cualquiera de los botones del ratón
MouseUp	Sucede al soltar un botón del ratón que previamente se había pulsado
MouseMove	Ocurre al mover el ratón sobre un control o un formulario.

¿Cómo codificar un procedimiento de evento?

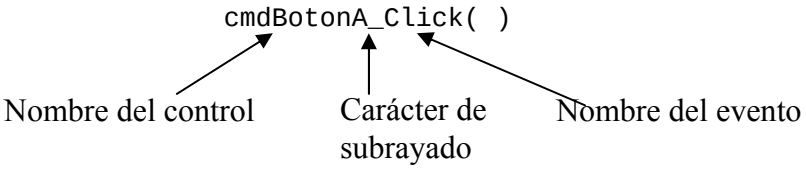
Los procedimientos de evento se comienzan con la instrucción `Private Sub` y finalizan con `End Sub`. Estas instrucciones representan la primera y la última línea, respectivamente, y entre ellas se encuentra lo que se denomina cuerpo del procedimiento, allí se colocan todas las instrucciones que el programa debe seguir cuando ocurra el evento en cuestión.

Sintaxis de un procedimiento de evento:

```
Private Sub  NombredelProcedimiento ( )

Instrucciones

End Sub
```

El nombre del procedimiento de evento está compuesto del nombre de un control, un carácter de subrayado y el nombre de un evento. Así por ejemplo, si se desea programar

el evento clic de un botón de comando llamado cmdBotonA, es necesario tener un procedimiento de nombre:

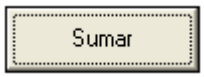


Y si se desea hacer un procedimiento para el evento doble clic, su nombre sería: `cmdBotonA_DblClick ( )`.

No es necesario saber de memoria los nombres de los eventos (dblclick, keydowm, mousemove, etc.), ya que cuando se está agregando controles a los formularios, es posible hacer doble clic sobre alguno de ellos y aparece la ventana de código, en la parte superior derecha se encuentra una lista despleglable con todos los eventos posibles para el control señalado.

Ejemplo de un procedimiento de evento

El siguiente código corresponde a un procedimiento para el evento clic de un botón de comando llamado cmdSumar.




```
Private Sub cmdSumar_Click( )
Dim a as Single, b as Single, suma as Single
a = 3
b = 8
suma = a + b
MsgBox ("La suma es " & suma)
End Sub
```

Las instrucciones de este procedimiento se ejecutarán inmediatamente después que el usuario haga clic sobre el botón Sumar.

5.4. Construcción de programas con una interfaz gráfica.

A partir de ahora, para crear un programa Visual Basic se seguirán estos pasos:

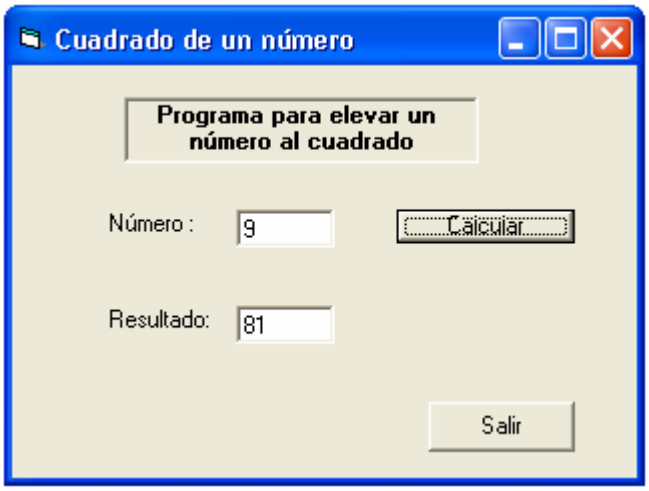
1. Se diseña y prepara la parte gráfica. Esto es, se colocan sobre el formulario los controles (botones, etiquetas, cuadros de texto, listas, etc.) que constituirán la aplicación.

2. Se define para cada control el código con el que se va a responder a los diferentes eventos. En aplicaciones sencillas, el evento que con mayor frecuencia se programa es el evento clic de los botones de comando.

A continuación se presentan algunos ejemplos resueltos, donde se trabaja con una interfaz gráfica y procedimientos de evento.

Ejemplo 1: programa que eleva un número entero al cuadrado.

a) Diseño de la interfaz gráfica



La interfaz de este programa consta de un formulario que tiene los siguientes controles:

- Tres etiquetas
- Dos cuadros de texto
- Dos botones de comando

Para comprender el programa, es conveniente indicar el nombre dado a los controles que se utilizan en el código. Recuérdese que el nombre de un control se coloca en la ventana propiedades de Visual Basic.

Control	***Nombre***
Cuadro de texto para escribir el número	TxtNúmero
Cuadro de texto para indicar el resultado	TxtResult
Botón calcular	CmdCalcular
Botón salir	CmdSalir

b) Código del programa

El código incluye dos procedimientos de evento, uno para el evento clic del botón Calcular y otro para el evento clic del botón Salir.

```
Option Explicit

Private Sub CmdCalcular_Click()
Dim num As Single, res As Single
num = txtNúmero.Text
res = num * num
TxtResult.Text = res
End Sub

Private Sub CmdSalir_Click()
Unload Me
End Sub
```

Acerca del código de este programa hay algunos detalles que vale la pena aclarar:

1. Es conveniente hacer la declaración de variables de la manera acostumbrada.

2. Obsérvese que el usuario introduce los datos en cuadros de texto. Por tal razón, el valor que tiene la propiedad Text del cuadro de texto donde el usuario escribe el número que desea elevar al cuadrado, debe asignarse a una variable:

```
num = txtNúmero.Text
```

`num`: es el nombre de la variable
`txtNúmero`: es el nombre del cuadro de texto donde el usuario escribe el número que desea elevar al cuadrado.
`Text`: hace referencia a la propiedad Text del cuadro de texto.

Es importante saber que para acceder a una propiedad de un control mediante código, se coloca el nombre del control seguido por un punto y luego se escribe la propiedad. En el caso anterior el control es un cuadro de texto de nombre txtNúmero y se está utilizando la propiedad Text.

3. Nótese que la salida del programa (resultado de elevar un número al cuadrado), se coloca en un cuadro de texto llamado txtResult. Para ello se le asigna a la propiedad Text de este cuadro la variable que almacena el resultado (res), mediante la instrucción:

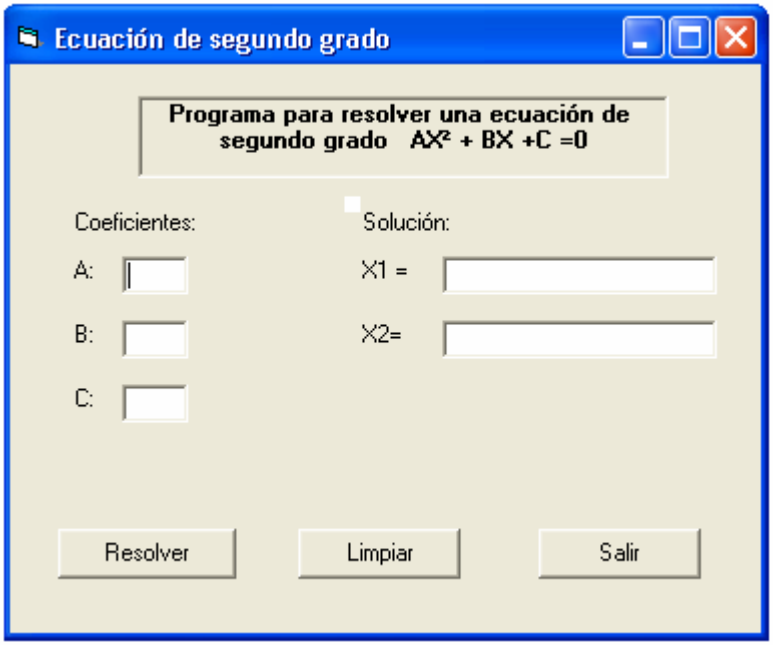
```
txtResult.Text = res
```

De esta manera en el cuadro de texto txtResult aparecerá el valor de la variable res.

4. El procedimiento del evento clic del botón cmdSalir incluye la instrucción Unload Me, que no se había visto hasta ahora. Unload descarga un formulario de la memoria, y cuando se escribe a continuación la palabra Me, descarga el formulario activo. En este ejemplo Unload Me descarga el formulario y cierra el programa.

Ejemplo 2: programa que resuelve una ecuación de segundo grado.

a) Diseño de la interfaz gráfica



Al hacer clic sobre el botón limpiar, se borran los valores escritos en los cuadros de texto y se dejan éstos en blanco, para que el usuario pueda introducir nuevos datos y resolver otra ecuación de segundo grado.

Nombres de los controles utilizados en el código:

Control	Nombre
Cuadro de texto para el coeficiente A	txtCoefA
Cuadro de texto para el coeficiente B	txtCoefB
Cuadro de texto para el coeficiente C	txtCoefC
Cuadro de texto para X1	txtValorX1
Cuadro de texto para X2	txtValorX2
Botón resolver	cmdResolver
Botón limpiar	cmdLimpiar
Botón Salir	cmdSalir

b) Código del programa

```
Option Explicit
Private Sub cmdLimpiar_Click()
```

```
txtCoefA.Text = ""
txtCoefB.Text = ""
txtCoefC.Text = ""
txtValorX1.Text = ""
txtValorX2.Text = ""
End Sub

Private Sub cmdResolver_Click()
Dim a As Single, b As Single, c As Single, d As Single
Dim X1 As Single, X2 As Single, pr As Single, pi As Single

a = txtcoefA.Text
b = txtCoefB.Text
c = txtCoefC.Text
d = b ^ 2 - 4 * a * c
If d >= 0 Then
X1 = (-b + Sqr(d)) / (2 * a)
X2 = (-b - Sqr(d)) / (2 * a)
txtValorX1.Text = X1
txtValorX2.Text = X2
Else
pr = -b / (2 * a)
pi = Abs(d)
txtValorX1.Text = pr & " + " & pi & "i"
txtValorX2.Text = pr & " - " & pi & "i"
End If

End Sub

Private Sub Salir_Click()
Unload Me
End Sub
```

Ejemplo 3: programa para calcular el salario mensual de un trabajador, de acuerdo a los siguientes criterios:

-El salario base depende del tipo de empleado: obreros, personal administrativo (no profesional), técnicos y profesionales. El salario base es 200000, 300000, 450000 y 900000 Bs., respectivamente.

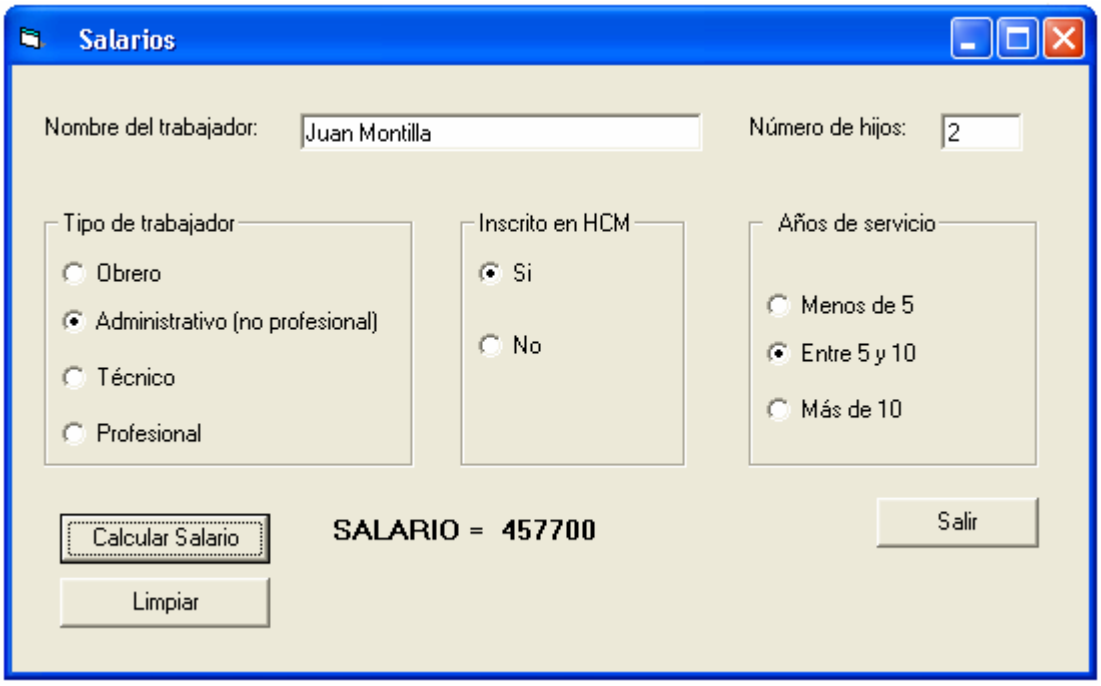
-Por cada hijo, el empleado obtiene un bono de 80000 Bs.

- Si el empleado está inscrito en el seguro HCM se le descuenta de su salario 52300 Bs.

-Dependiendo de los años de servicio que el empleado tiene en la empresa, se da la siguiente bonificación:

Años de servicio	**Bonificación**
Menos de 5	0
Entre 5 y 10	50000
Más de 10	95000

a) Diseño de la interfaz gráfica



Nombre de los controles utilizados en el código:

Control	**Nombre**
Cuadro de texto para el número de hijos	txtHijos
Botón de Opción obrero	optObr
Botón de Opción Administrativo	optAdm
Botón de Opción Técnico	optTec
Botón de Opción Profesional	optProf
Botón de Opción menos de 5 (Años de servicio)	optAño5
Botón de Opción entre 5 y 10 (Años de servicio)	optAño5_10
Botón de Opción más de 10 (Años de servicio)	optAño10
Botón de Opción Si (HCM)	optHCM1
Botón de Opción No (HCM)	optHCM2
Botón Calcular salario	cmdCalcularSal
Botón Limpiar	cmdLimpiar
Botón Salir	cmdSalir
Etiqueta para escribir el salario	lblSalario

b) Código del programa

```
Option Explicit

Private Sub cmdCalcularSal_Click()
Dim nh As Byte, nom As String, sb As Single
Dim bonoaño As Single, seguro As Single, sal As Single
```

```
nh = txthijos.Text

If optObr.Value = True Then
sb = 200000
Else
  If OptAdm.Value = True Then
  sb = 300000
  Else
    If OptTec.Value = True Then
    sb = 450000
    Else
    sb = 900000
    End If
  End If
End If

If optaño5.Value = True Then
bonoaño = 0
Else
  If optAño5_10.Value = True Then
  bonoaño = 50000
  Else
    If optAño10.Value = True Then
     bonoaño = 100000
    End If
  End If
End If

If optHCM1 = True Then
  seguro = 52300
Else
  seguro = 0
End If

sal = sb + 80000 * nh + bonoaño - seguro
LblSalario.Caption = "SALARIO = " & sal
End Sub

Private Sub cmdLimpiar_Click()
nombre.Text = " "
hijos.Text = " "
LblSalario.Caption = " "
End Sub

Private Sub cmdSalir_Click()
Unload Me
End Sub
```

Comentarios sobre el código:

1. Para determinar si el usuario seleccionó un botón de opción determinado se utiliza la propiedad Value. Si es True significa que el usuario seleccionó el botón, si es False no lo seleccionó. Por ejemplo: hay 4 tipos de empleado (obrero, administrativo, técnico y profesional), para saber si un usuario seleccionó la primera opción se utiliza la propiedad Value de la siguiente forma:

```
If optObr.Value = True Then
```

Con esta instrucción se verifica si el usuario seleccionó el botón de opción de nombre optObr, el cual corresponde a la opción Obrero. Se procede de forma similar para las demás opciones, también para el grupo de botones de opción correspondientes a la inscripción en el seguro HCM y a los años de servicio.

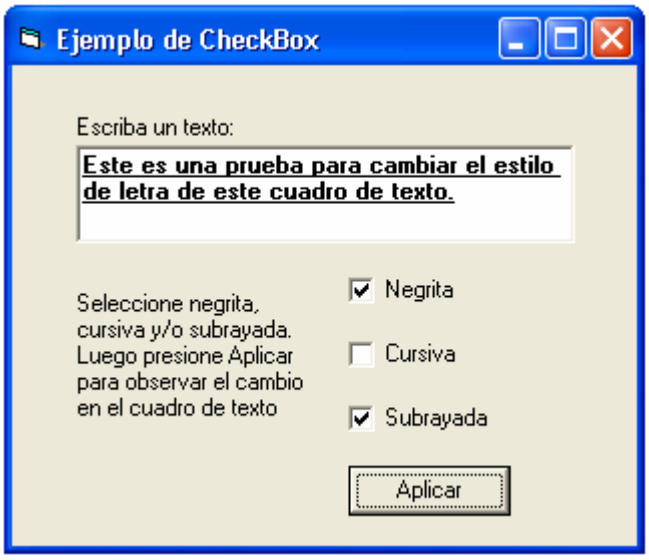
2. El resultado se escribe en una etiqueta de nombre lblResult, que en el momento de diseño del formulario se deja vacía. Luego, después de calcular el salario total (variable de nombre sal), el resultado se asigna a la propiedad Caption de dicha etiqueta mediante la instrucción:

```
LblSalario.Caption = "SALARIO = " & sal
```

Mostrar los resultados en una etiqueta es más conveniente, ya que el usuario no puede modificarlo.

Ejemplo 4: programa que permite al usuario escribir varias líneas de texto y seleccionar el estilo de la letra.

a) Diseño de la interfaz gráfica



En este formulario se incluye un cuadro de texto con la propiedad Multiline = True, con la finalidad que el usuario pueda escribir varias líneas de texto.

Se utilizan controles tipo Checkbox para seleccionar una o más opciones. El cambio de estilo de letra se observará después de hacer clic sobre el botón Aplicar.

Nombre de los controles utilizados en el código:

Control	**Nombre**
Cuadro de texto	txtTexto
CheckBox para seleccionar negrita	chkNeg
CheckBox para seleccionar cursiva	chkCur
CheckBox para seleccionar subrayada	chkSubr
Botón Aplicar	cmdAplicar

b) Código del programa

```
Private Sub cmdAplicar_Click()

If chkNeg.Value = 1 Then
  txtTexto.FontBold = True
Else
  txtTexto.FontBold = False
End If

If chkcur.Value = 1 Then
   txtTexto.FontItalic = True
Else
   txtTexto.FontItalic = False
End If

If chkSubr.Value = 1 Then
   txtTexto.FontUnderline = True
Else
   txtTexto.FontUnderline = False
End If
End Sub
```

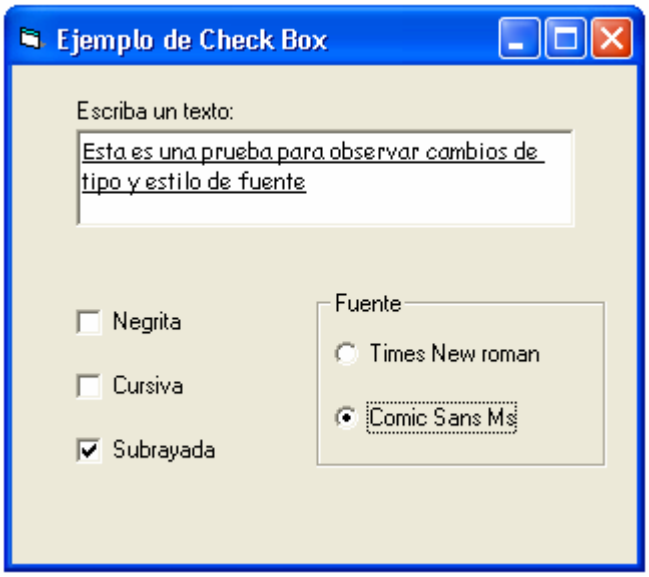
Obsérvese que en el código del programa se usan las siguientes propiedades de los cuadro de texto:

FontBold : si vale True la letra se muestra en negrita.
FontItalic: si se establece en True la letra es cursiva
Fontunderline: si es igual a True el texto aparece subrayado.

Estas propiedades también las tienen otros objetos de Visual Basic (formularios, etiquetas, botones de comando, botones de opción, etc.)

Ejemplo 5: este programa es parecido al ejemplo 4 pero además de seleccionar el estilo de letra, el usuario puede elegir entre dos tipos de fuente.

a) Diseño de la interfaz gráfica



Nótese que el formulario no tiene un botón de comando donde se pueda hacer clic para ver los cambios en el texto. Estos cambios se observarán inmediatamente después que el usuario haga clic sobre un checkbox (negrita, cursiva, subrayada) o sobre alguno de los botones de opción que indica la fuente.

Los nombres de los CheckBox y del cuadro de texto son los mismos utilizados en el ejemplo 4. Los botones de opción tienen como nombre optTim (Times New Roman) y optCom (Comic Sans Ms)

b) Código del programa

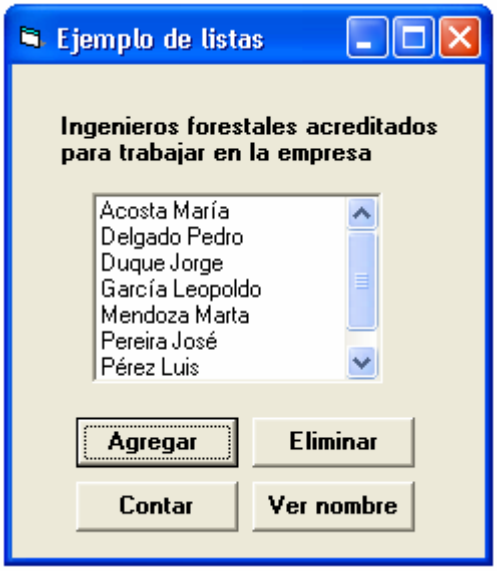
```
Private Sub chkCur_Click()
If chkcur.Value = 1 Then
texto.FontItalic = True
Else
texto.FontItalic = False
End If
End Sub

Private Sub chkNeg_Click()
If chkNeg.Value = 1 Then
texto.FontBold = True
Else
texto.FontBold = False
End If
End Sub

Private Sub chkSubr_Click()
If subr.Value = 1 Then
texto.FontUnderline = True
Else
texto.FontUnderline = False
End If
```

```
End Sub
Private Sub optTim_Click()
  texto.FontName = "times new roman"
End Sub

Private Sub optCom_Click()
texto.FontName = "comic sans ms"
End Sub
```

Comentarios sobre el código:
1. Se utiliza la propiedad FontName para indicar el nombre de la fuente.
2. Se programa el evento clic de los CheckBox y los botones de opción, para lograr que el cambio en la fuente del cuadro de texto se vea cuando el usuario haga clic sobre alguna de las opciones disponibles.

Ejemplo 6: este es un programa muy sencillo que ilustra el uso de un ListBox. Se tiene una lista de los ingenieros forestales que están acreditados para trabajar en una empresa determinada, el usuario puede agregar y eliminar ingenieros. El programa también puede indicar la cantidad de nombres que hay en la lista.

a) Diseño de la interfaz gráfica



El botón agregar (cmdAgregar) solicita el nombre del ingeniero que se desea incluir en la lista por medio de un InputBox y lo luego lo agrega.

El botón eliminar (cmdEliminar) saca de la lista al ingeniero que esté seleccionado. Si no hay alguno seleccionado, muestra un MsgBox indicándole al usuario que debe elegir un nombre de la lista.

El botón contar (cmdContar) muestra un MsgBox indicando la cantidad de ingenieros que hay en la lista.

El botón Ver nombre (cmdVer) muestra en un MsgBox, el nombre del ingeniero seleccionado.

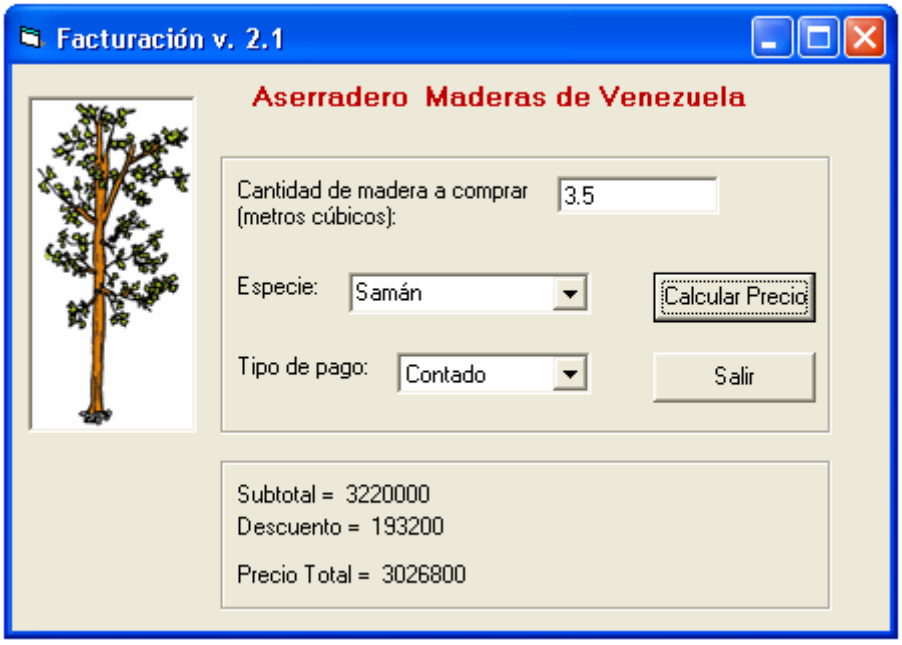
El nombre de la lista es lstIngFor. La propiedad Sorted de esta lista fue definida en true, con la finalidad que la lista esté ordenada alfabéticamente.

b) Código del programa

```
Option Explicit
Private Sub cmdagregar_Click()
Dim nom As String, resp As Integer
nom = InputBox("Escriba el nombre del Ingeniero Forestal que desea agregar", _
"agregar")
If nom <> "" then
LstIngFor.AddItem nom
resp = MsgBox(nom & " ha sido agregado a la lista", 0 + vbInformation, _
"Agregar")
else
resp = MsgBox ("Debe escribir un nombre", vbExclamation, "Agregar")
End If
End Sub

Private Sub cmdContar_Click()
Dim n As Integer, resp As Integer
n = LstIngFor.ListCount
resp = MsgBox("Hay "  &  n  &  " ingenieros forestales en la lista", _
0 + vbInformation, "Contar")
End Sub

Private Sub cmdEliminar_Click()
Dim resp As Integer, ingsel As Integer
Dim nom As String
ingsel = LstIngFor.ListIndex
If ingsel = -1 Then
 resp=MsgBox("Debe seleccionar en la lista el ingeniero que desea eliminar", _
 vbExclamation, "Eliminar")
Else
 nom = LstIngFor.List(ingsel)
 LstIngFor.RemoveItem (ingsel)
 resp = MsgBox(nom & " ha sido eliminado de la lista", 0 + vbInformation, _
 "Eliminar")
End If
End Sub

Private Sub CmdVer_Click()
Dim nom As String, resp As Integer
ingsel = LstIngFor.ListIndex
If ingsel = -1 Then
 resp = MsgBox("Debe seleccionar un ingeniero de la lista", vbExclamation, _
 "Ver")
Else
nom = LstIngFor.List(ingsel)
resp = MsgBox("El Ing. For. seleccionado es " & nom, vbInformation, _
"Selección")
End If
End Sub
```

Ejemplo 7: programa para calcular el precio total a pagar por la compra de madera húmeda en un aserradero, considerando la cantidad de metros cúbicos adquirida. El aserradero vende principalmente cedro, pardillo y samán, cuyos precios por metro cúbico son: Bs. 800000, 600000 y 920000. Eventualmente se puede vender otra especie, en ese caso el programa debe solicitar el precio por metro cúbico. Si el pago es de contado se aplican los siguientes descuentos: cedro (8%), pardillo (10%), samán (6%), otra especie (2%).

a) Diseño de la interfaz gráfica



En esta interfaz se usan dos ComboBox, uno que permite elegir entre Cedro, Pardillo y Samán, con la propiedad Style=1 para que pueda escribirse otra especie diferente. El segundo ComboBox presenta dos opciones de pago: crédito y contado, con la propiedad Style=3 para que el usuario no pueda escribir otra forma de pago distinta.

Obsérvese también que se utiliza un control Image para colocar una imagen en el formulario. También se pudo haber utilizado un PictureBox.

Nombre de los controles utilizados en el código:

Control	**Nombre**
Cuadro de texto para la cantidad de madera	txtcantidad
ComboBox para seleccionar especie	cboEspecie
ComboBox para seleccionar tipo de pago	cboTipopago
Frame donde colocan los resultados	frmResult
Etiqueta para el subtotal	lblSubtot
Etiqueta para el descuento	lblDescuento
Etiqueta para el total a pagar	lblPagototal
Botón Calcular Precio	cmdCalcular
Botón Salir	cmdSalir

b) Código del programa

```
Option Explicit
Private Sub cmdCalcular_Click()
Dim esp As Integer, tipop As Integer, cant As Single, d As Single,
preciototal As Single, precio As Single, resp as integer
Dim subtotal As Single
Const preciocedro = 800000, preciopard = 600000, preciosaman = 920000

If txtcantidad.Text = "" Or cbotipopago.Text = "" Or cboespecie.Text = "" Then
   resp = MsgBox("Faltan datos por introducir", vbCritical, "error")
Else
   cant = txtCantidad.Text
   esp = cboEspecie.ListIndex
   tipop = cboTipopago.Listindex
   d = 0
   Select Case esp
   Case 0
    subtotal = preciocedro * cant
    If tipop = 0 Then
      d = 0.08 * subtotal
    End If
   Case 1
       subtotal = preciopard * cant
       If tipop = 0 Then
         d = 0.1 * subtotal
        End If

   Case 2
       subtotal = preciosaman * cant
       If tipop = 0 Then
          d = 0.06 * subtotal
       End If

   Case Else
          precio = InputBox("Escriba el precio por metro cúbico")
          subtotal = precio * cant
          If tipop = 0 Then
            d = 0.02 * subtotal
          End If
   End Select

preciototal = subtotal – d
FrmResult.Visible = True
lblSubtot.Caption = "Subtotal = " & subtotal
lblDescuento.Caption = "Descuento = " & d
lblPagototal.Caption = "Precio Total = " & preciototal
End If

End Sub

Private Sub cmdSalir_Click()
Unload Me
End Sub
```

Comentarios sobre el código:
1. Los precios de la madera se colocaron como constantes para facilitar la actualización del programa, esto es, cuando cambien los precios sólo será necesario modificar las constantes.

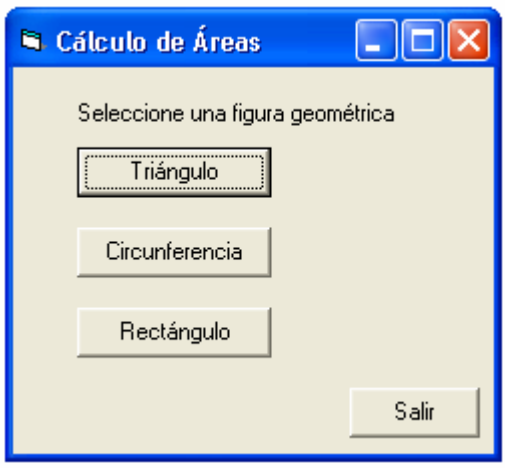
2. Se verifica que no falten datos antes de hacer los cálculos. Si el usuario no introdujo algún dato de entrada se muestra un mensaje de error.

3. Los resultados se escriben en tres etiquetas que están dentro de un frame de nombre Frmresul. Cuando el programa se ejecuta este marco no se ve, pues su propiedad Visible fue definida en False en tiempo de diseño. Para poder ver los resultados es necesario cambiar esta propiedad a True, lo que se hace con la instrucción:

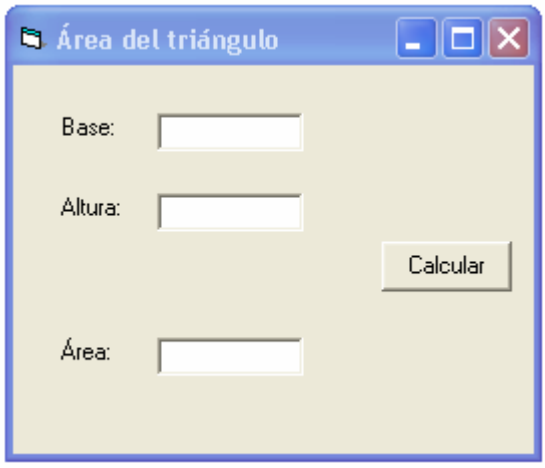
```
FrmResult.Visible = True
```

Ejemplo 8: programa que permite calcular el área de diferentes figuras geométricas: triángulo, circunferencia y rectángulo.

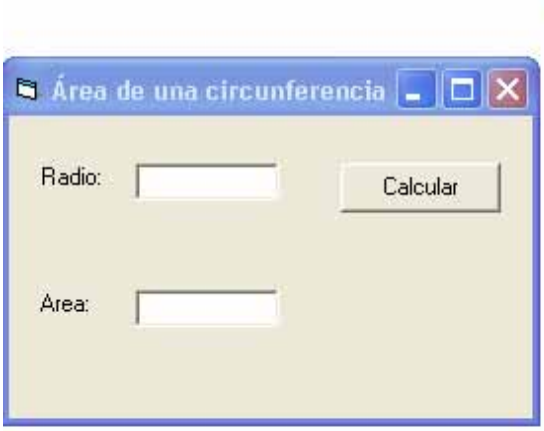
a) Diseño de la interfaz gráfica
Este programa utiliza cuatro formularios. El primer formulario es el formulario principal, llamado FormAreas, y en él se selecciona la figura geométrica deseada.
Hay un formulario adicional para cada tipo de figura, si el usuario selecciona triángulo se abre el formulario FormAreaT, si elige circunferencia aparece el formulario FormAreaC y si selecciona rectángulo, se observa el formulario FormAreaR.



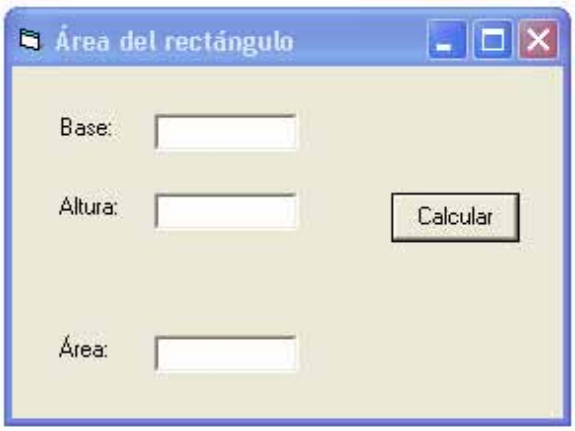
Formulario Principal (FormAreas)



Formulario FormAreaT



Formulario FormAreaC



Formulario FormAreaR

Nombre de los controles usados en el código:
Los botones de comando del formulario principal tienen por nombre cmdTriang, cmdCircunf, cmdRectan y cmdSalir. Los cuadros de texto donde se escriben los datos de entrada y salida se llaman: txtbase, txtaltura, txtradio y txtarea.

El botón calcular del formulario FormAreaT tiene por nombre cmdCalcularAT, el botón calcular del formulario FormAreaC se llama cmdCalcularAC, y el botón calcular del formulario FormAreaR tiene como nombre cmdCalcularAR.

b) Código del programa

```
‘ CÓDIGO DEL FORMULARIO FormAreas
Option Explicit
Private Sub cmdTriang_Click()
FormAreaT.Show
End Sub

Private Sub cmdCircunf_Click()
FormAreaC.Show
End Sub


Private Sub cmdRectan_Click()
FormAreaR.Show
End Sub

Private Sub cmdSalir_Click()
Unload FormAreaT
Unload FormAreaC
Unload FormAreaR
Unload Me
End Sub

‘CODIGO DEL FORMULARIO FormAreaT
Option Explicit
Private Sub cmdCalcularAT_Click()
Dim b As Single, h As Single, ar As Single
```

```
b = txtbase.Text
h = txtaltura.Text
ar = b * h / 2
txtarea.Text = ar

End Sub

‘CODIGO DEL FORMULARIO FormAreaC
Option Explicit
Private Sub cmdCalcularAC_Click()
Dim r As Single, ar As Single
r = txtradio.Text
ar = 3.1416 * r ^ 2
txtarea.Text = ar
End Sub

‘CODIGO DEL FORMULARIO FormAreaR
Option Explicit
Private Sub cmdCalcularAR_Click()
Dim b As Single, h As Single, ar As Single
b = txtbase.Text
h = txtaltura.Text
ar = b * h
txtarea.Text = ar
End Sub
```

Comentarios acerca del código:

1. El ejemplo anterior es un programa con múltiples formularios. En estos casos siempre debe haber un formulario principal, que es el que aparece al arrancar la aplicación. Por defecto, Visual Basic considera como formulario principal el primero que se haya creado. Sin embargo, es posible indicar cuál es el formulario principal en el menú Proyecto / Propiedades de proyecto, en la pestaña General y en la sección Objeto inicial.

2. En tiempo de diseño se utiliza el menú Proyecto/Agregar Formulario, para incorporar nuevos formularios.

3. Para que en tiempo de ejecución se pueda observar un formulario distinto al inicial, se utiliza el método Show. Por tal razón, en los procedimientos del evento clic de los botones de comando Triángulo, Circunferencia y Rectángulo, se escribieron las instrucciones FormAreaT.Show, FormAreaC.Show y FormAreaR.Show respectivamente, de manera que puedan observarse los formularios correspondientes a cada figura geométrica.

4. Obsérvese que en el procedimiento cmdSalir_Click se descargan todos los formularios de la aplicación con la instrucción Unload.

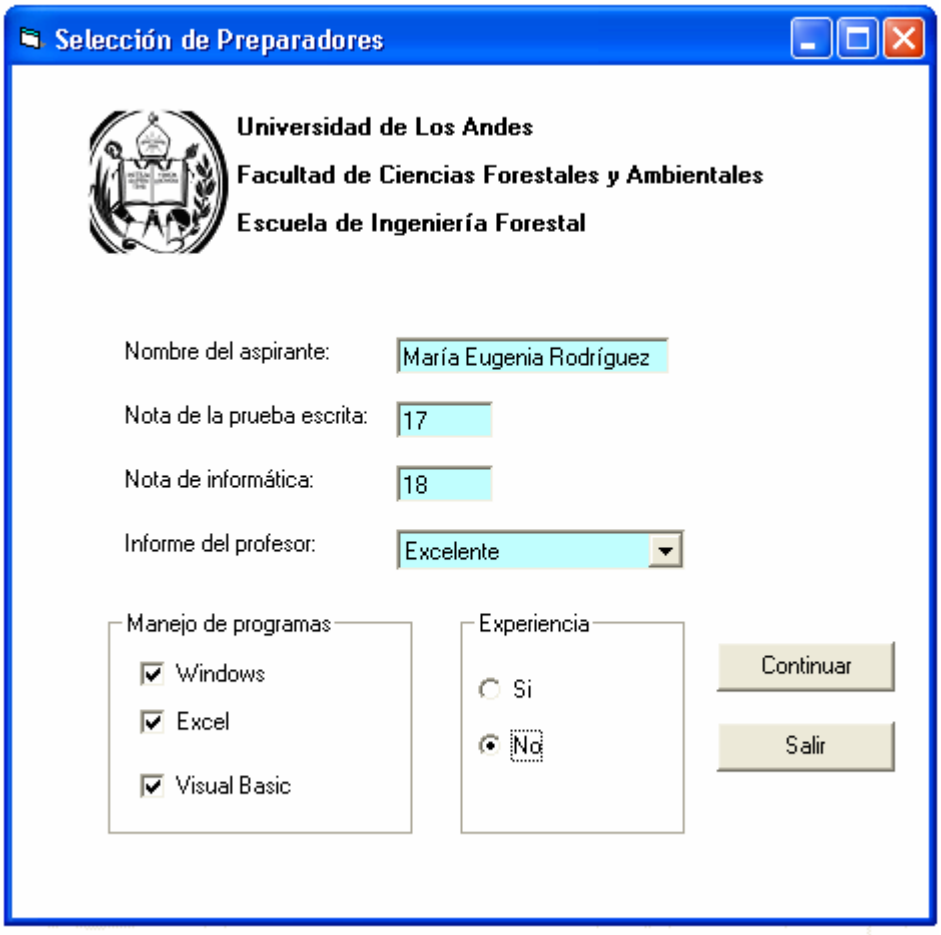
Ejemplo 8: programa que muestra como salida la calificación definitiva de un aspirante a preparador de Informática, la cual se calcula promediando la nota obtenida en la asignatura, la calificación del examen escrito y la nota de la prueba de

credenciales. Las dos primeras calificaciones son datos de entrada y la tercera debe determinarla el programa de acuerdo a los siguientes criterios:
- Manejo avanzado y/o cursos: lenguaje de programación Visual Basic (5 puntos), Sistema Operativo Windows (5 puntos), hoja de cálculo Excel (5 puntos).
- Experiencia como preparador (2 puntos)
- Informe elaborado por el profesor encargado acerca del aspirante, basado en una entrevista previa. La puntuación es: Excelente (3 puntos), Bueno (2 puntos), Regular (1 punto) y Malo (0 puntos)

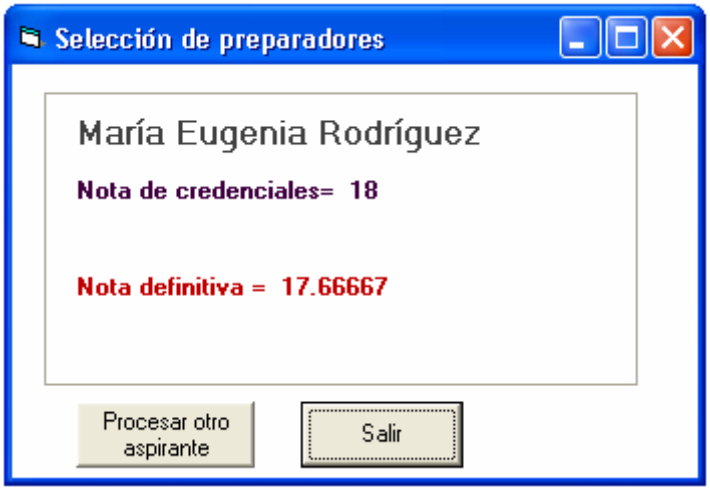
La puntuación de cada ítem debe sumarse y el total corresponde a la calificación de la prueba de credenciales.

a) Diseño de la interfaz gráfica

La interfaz gráfica está compuesta de dos formularios: FormEntrada donde se introducen los datos y FormResultados para mostrar la calificación definitiva.



Formulario FormEntrada



Formulario FormResultados

Nombre de los controles usados en el código:

Formulario FormEntrada

Control	Nombre
Cuadro de texto para el nombre	txtNombre
Cuadro de texto para la nota de la prueba escrita	txtNotape
Cuadro de texto para la nota de informática	txtNotainf
ComboBox para el informe del profesor	cboInforme
CheckBox para Windows	chkWIND
CheckBox para Excel	chkEXC
CheckBox para Visual Basic	chkVB
Botón de opción para Experiencia Si	optExpsi
Botón de opción para Experiencia No	optExpno
Botón Continuar	cmdContinuar
Botón Salir	cmdSalir

Formulario FormResultados

Control	Nombre
Etiqueta para el nombre	lblNombre_asp
Etiqueta para la nota de credenciales	lblNotacre
Etiqueta para la nota definitiva	lblNotadef
Botón para procesar otro aspirante	cmdOtroAsp
Botón Salir	cmdSalir

b) Código del programa

```
‘ Código del formulario FormEntrada
Option Explicit
Private Sub cmdContinuar_Click()
Dim npe As Single, ni As Single, nc As Single, nd As Single
```

```
Dim inf_prof As Byte

npe = txtNotape.Text
ni = txtNotainf.Text
inf_prof = cboInforme.ListIndex

If inf_prof = 0 Then
nc = 3
Else
  If inf_prof = 1 Then
     nc = 2
   Else
     If inf_prof = 2 Then
       nc = 1
      End If
   End If
End If

If chkWIND.Value = 1 Then
nc = nc + 5
End If

If chkEXC.Value = 1 Then
nc = nc + 5
End If

If chkVb.Value = 1 Then
  nc = nc + 5
End If

If optEXP_si.Value Then
   nc = nc + 2
End If

nd = (npe + ni + nc) / 3
FormResultados.Show
FormResultados.lblNombre_asp.Caption = txtNombre.Text
FormResultados.lblNotacre.Caption = "Nota de credenciales= " & nc
FormResultados.lblNotadef.Caption = "Nota definitiva = " & nd
End Sub

‘Código del formulario FormResultados
Private Sub cmdSalir_Click()
Unload FormResultados
Unload Me
End Sub

Private Sub cmdOtroAsp_Click()
Unload Me
FormEntrada.txtNombre.Text = ""
FormEntrada.txtNotainf.Text = ""
FormEntrada.txtNotape.Text = ""
FormEntrada.cboInforme.ListIndex = -1
FormEntrada.chkEXC.Value = 0
FormEntrada.chkVb.Value = 0
FormEntrada.chkWIND.Value = 0
FormEntrada.optExp_no.Value = True
End Sub
```

```
Private Sub cmdSalir_Click()
Unload Me
Unload FormEntrada
End Sub
```

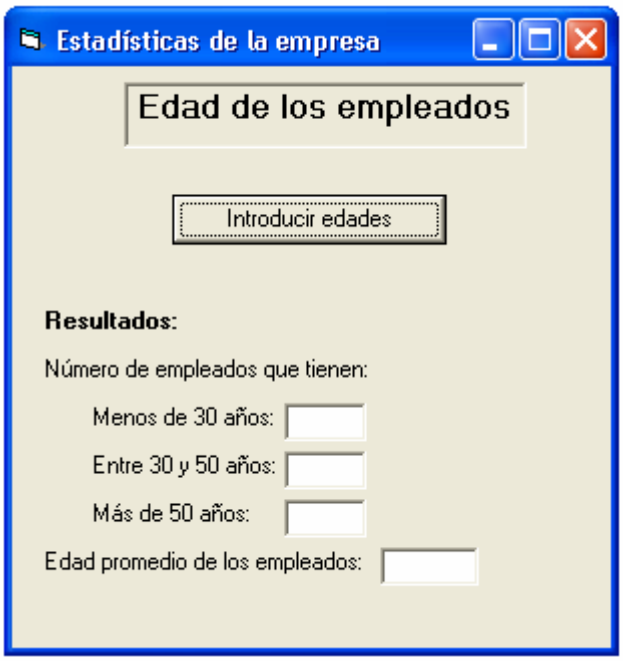
Nótese que en el código del formulario FormEntrada, se manda a escribir los resultados en etiquetas que están en el formulario FormResultados. Para hacer referencia a un objeto de otro formulario es necesario colocar el nombre de dicho formulario, un punto y luego el nombre del objeto. Ejemplo:

```
FormResultados.lblNombre_asp.Caption = txtNombre.Text
```

Esta instrucción se encuentra en el procedimiento de evento cmdContinuar_click del formulario FormEntrada y hace referencia a la etiqueta lblNombre_Asp del formulario FormResultados

Ejemplo 10: Programa que recibe como datos de entrada las edades de los 50 empleados de una empresa y calcula: a) el número de empleados que tiene menos de 30 años, b) el número de empleados que tiene entre 30 y 50 años, c) el número de empleados que tiene más de 50 años, y d) la edad promedio de los empleados. Este es igual al ejemplo 6 de estructuras de repetición, pero utiliza una interfaz gráfica.

a) Diseño de la interfaz gráfica



El botón introducir edades, mostrará un InputBox para solicitar la edad de cada empleado (aparecerán 50 InputBox). Finalizada la entrada de los datos se hacen los cálculos y se muestran los resultados en los cuadros de texto destinados para ello.

Nombre de los controles usados en el código:

Control	**Nombre**
Botón para introducir las edades	cmdInt_edad
Cuadro de texto para el número de empleados que tienen menos de 30 años	txtmenos_30
Cuadro de texto para el número de empleados que tienen entre 30 y 50 años	txtEntre30_50
Cuadro de texto para el número de empleados que tienen más de 50 años	txtMas_50
Cuadro de texto para la edad promedio	txtPromedio

b) Código del programa

```
Option Explicit
Private Sub cmdInt_edad_Click()
Dim i As Integer, edad As Byte, e1 As Byte, e2 As Byte, e3 As Byte
Dim prom As Single, sumaedad As Integer

sumaedad = 0
For i= 1 to 50 do
   edad = InputBox("Edad " & i)
   sumaedad = sumaedad + edad
   If edad < 30 Then
     e1 = e1 + 1
   Else
     If edad <= 50 Then
       e2 = e2 + 1
     Else
       e3 = e3 + 1
     End If
   End If
 Next i
prom = sumaedad / 5
txtPromedio.Text = prom
txtMenos_30.Text = e1
txtEntre30_50.Text = e2
txtMas_50.Text = e3

End Sub
```

5.5. Ejercicios propuestos

Realizar un programa en Visual Basic para resolver cada uno de los siguientes problemas. La solución debe incluir el diseño de la interfaz gráfica y el código respectivo.

1. En una empresa se desea un programa que realice un reajuste en los precios de sus productos. Los productos se agrupan en cuatro categorías y el aumento en sus precios se realiza en base a la siguiente tabla:

Categoría	Aumento de precio
A	15%
B	10%
C	8%
D	7%

Las entradas del programa deben ser: nombre del producto, categoría y precio actual. El programa debe mostrar el nuevo precio con el aumento.

2. Se desea un programa para calcular el salario semanal de un trabajador si se tienen como datos el número de horas trabajadas y el salario por hora. Cuando el salario bruto (número de horas trabajadas x salario por hora) es superior a 100000 Bs. se le descuenta el 1% de impuestos y si el empleado trabajó más de 40 horas, se le da una bonificación extra del 5%. El programa debe mostrar: salario bruto, impuesto, bonificación y salario neto. Usar dos formularios.

3. En un aserradero se necesita un programa que calcule el monto a pagar por una compra de madera. Si la cantidad de madera comprada es superior o igual a 10 metros cúbicos, el esquema de descuento a aplicar es el siguiente:

Especie	Descuento
Saqui-saqui	10 %
Cedro	8 %
Apamate	7 %
Samán	6 %
Pardillo	6 %
Otro	5 %

Si la cantidad de madera comprada es inferior a 10 metros cúbicos el descuento es del 2%, independientemente de la especie.

4. Realizar un programa que imite el funcionamiento de una calculadora simple. Esto es, debe aceptar dos números y hacer una de las siguientes operaciones: suma, resta, multiplicación, división.

5. Un negocio mayorista vende disquetes, discos compactos y ratones (mouse). Se necesita un programa que acepte como datos de entrada el precio unitario de estos artículos y la cantidad de cada uno de ellos pedida por un cliente; para calcular el precio total que éste debe pagar. Si el pago es de contado se aplican los siguientes descuentos de acuerdo al tipo de cliente:

Tipo de Cliente	Descuento
A	15%
B	10%
C	5%

Si el pago es a crédito se utiliza el siguiente esquema de descuentos:

Tipo de Cliente	Descuento
A	10%
B	5%
C	2%

6. Construir un programa que permita hacer cualquiera de las siguientes conversiones (el usuario debe elegir la conversión): Metros a pulgadas, metros a pies, metros a millas, pies a pulgadas.
El programa debe tener como entrada la cantidad que se desea convertir.

Nota:
1 metro equivale a 39.3701 pulgadas.
1 metro equivale a 0.0006214 millas
1 pie equivale a 12 pulgadas.

7. Realizar un programa para calcular la superficie y el volumen de los siguientes cuerpos sólidos: esfera, cilindro, cubo y cono. El usuario elige el cuerpo sólido con el que desea trabajar. Las fórmulas se dan en la siguiente tabla:

	Esfera	Cilindro	Cubo	Cono
Superficie	$4.\pi.r^2$	$2.\pi.r.h + 2.\pi.r^2$	$6.L^2$	$\pi.r\sqrt{r^2+h^2} + \pi.r$
Volumen	$4.\pi.r^3 / 3$	$\pi.r^2.h$	L^3	$\pi r^2 h/3$

8. En una compañía de seguros se quiere un programa que indique si a un conductor le corresponde o no un descuento sobre su póliza. Sólo se concederá un descuento en los siguientes casos:
-El conductor tiene más de 25 años, es mujer y hace un año o más que no tiene accidentes.
-El conductor tiene más de 25 años, es hombre y hace más de dos años que no tiene accidentes.
- El conductor tiene 25 años o menos y más de cinco años que no tiene un accidente.

9. Se necesita un programa que acepte como datos de entrada el diámetro y la altura de n árboles y calcule: a) diámetro promedio, b) altura promedio, c) no. de árboles con un diámetro inferior a 20 cm., d) no. de árboles con diámetro entre 20 y 30 cm., y e) no. de árboles con un diámetro superior a 30 cm.

10. Se tienen los promedios de los alumnos de un salón; hacer un programa que calcule: a) número y porcentaje de estudiantes aplazados, b) número y porcentaje de estudiantes reprobados, y c) promedio general del curso.

11. Realizar un programa que incluya una interfaz gráfica para los ejercicios propuestos en los capítulos anteriores.

BIBLIOGRAFÍA

1. CAIRÓ, O. *Metodología de la programación. Tomo 1.* México D.F.: Alfaomega, 1995. 476 p.

2. CORBÍ, A. F. LLOPIS., F. LLORENS, MONLLOR M., MORA F., ORTUÑO F., PÉREZ E., SATORRE R. *Fundamentos de programación. Volumen I. Metodología.* Alicante: Universidad de Alicante, 1998. 189 p.

3. GARCÍA DE JALÓN., J., J. RODRÍGUEZ, A. BRAZÁLEZ. *Aprenda Visual Basic 6.0 como si estuviera en primero.* San Sebastián: Universidad de Navarra, 1999. 99 p.

4. GARCÍA DE JALÓN., J., J. RODRÍGUEZ, A. BRAZÁLEZ. *Practique Visual Basic 6.0 como si estuviera en primero*. Madrid: Universidad de Navarra, 2003. 53 p.

5. JOYANES, L. *Fundamentos de programación*. México D.F: McGraw Hill, 1990. 702 p.

6. JOYANES L., L. RODRÍGUEZ, M. FERNÁNDEZ. *Fundamentos de programación. Libro de problemas*. Madrid: McGraw Hill, 1996.392 p.

7. LÓPEZ, L. *Programación Estructurada.* México D.F.: Alfaomega, 1994. 282 p.

8. MATA-TOLEDO, R., P. CUSHMAN. *Introducción a la programación.* México D.F: McGraw Hill, 2001. 331 p.

9. PERALTA, L. *Introducción a la programación estructurada: Algoritmos*. Bogotá: Politécnico Grancolombiano, 1999. 144 p.

10. PERRY, G. *Aprendiendo Visual Basic 6*. México: Prentice Hall, 1999. 457 p.