

## **“PORTAFOLIO DE EVIDENCIAS”**

# **APLICA LA METODOLOGIA DE DESARROLLO RAPIDO DE APLICACIONES CON PROGRAMACION ORIENTADA A EVENTOS**

**Profesor: L.I. Mario Lagunas Brito**

**NOMBRE: Luis Eduardo Bahena Castillo**

**GRADO Y GRUPO: 3BPM**

**Nº DE LISTA: #2**

**FECHA: VIERNES 29 DE NOVIEMBRE DEL 2019**

**CORREO ELECTRONICO:**

**[luiseduardobahenacastillo007@gmail.com](mailto:luiseduardobahenacastillo007@gmail.com)**



## INDICE

|   |  |
|---|--|
| PORTADA.....                                    |  |
| INDICE.....                                     |  |
| APUNTE: SENTENCIAS DE CONTROL.....              |  |
| APUNTE: CONTROL PROGRESSBAR.....                |  |
| APUNTE: OPERADORES LOGICOS.....                 |  |
| EJERCICIOS DE OPERADORES LOGICOS.....           |  |
| EJERCICIOS CON SENTENCIA IF.....                |  |
| EJERCICIOS DE CICLO For (Visual Basic 6.0)..... |  |
| ANEXO DE APLICACIONES.....                      |  |
| Aplicación 9:Contraseña Temporizada.....        |  |
| Aplicación 10: Decisiones.....                  |  |
| Aplicación 11: Fechas con año bisiesto.....     |  |
| Aplicación 12: Linea de tiempo .....            |  |
| Aplicación 13: Fecha, hora y collage.....       |  |
| Aplicación 14: Estados.....                     |  |
| Examen  |  |
| Aplicación de Países.....                       |  |

## COMENTARIO PERSONAL.....



## APUNTE: SENTENCIAS DE CONTROL

Las **sentencias de control**, denominadas también *estructuras de control*, permiten tomar decisiones y realizar un proceso repetidas veces. Son los denominados bifurcaciones y bucles. Este tipo de estructuras son comunes en cuanto a concepto en la mayoría de los lenguajes de programación, aunque su sintaxis puede variar de un lenguaje de programación a otro. Se trata de unas estructuras muy importantes ya que son las encargadas de controlar el *flujo* de un programa según los requerimientos del mismo. **Visual Basic 6.0** dispone de las siguientes estructuras de control:

- |                        |   |                                 |
|------------------------|---|---------------------------------|
| • If ... Then ... Else | } | Sentencias de Decisión          |
| • Select Case          |   |                                 |
| • For ... Next         | } | Sentencias Repetitivas (Ciclos) |
| • Do ... Loop          |   |                                 |
| • While ... Wend       |   |                                 |
| • For Each ... Next    |   |                                 |

### 1.1 Sentencias de Decisión.

#### 1.1.1 Sentencia IF ... THEN ... ELSE ...

##### a) If Simple.

Esta estructura ejecuta una o más sentencias si se cumple con una condición, de lo contrario continúa la ejecución normal del programa. No tiene parte falsa. Su estructura es la siguiente:

```
If condicion Then
    sentencia(s)
End If
```

**b) If Doble.**

Esta estructura permite ejecutar condicionalmente una o más sentencias y puede escribirse de dos formas. La primera ocupa sólo una línea y tiene la forma siguiente:

```
If condicion Then sentencia1 [Else sentencia2]
```

La segunda es más general y se muestra a continuación:

```
If condicion Then
    sentencia(s)
[Else
    sentencia(s) ]
End If
```

Si **condiciones True (verdadera)**, se ejecutan las sentencias que están a continuación de **Then**, y si **condiciones False (falsa)**, se ejecutan las sentencias que están a continuación de **Else**, si esta cláusula ha sido especificada (pues es opcional).

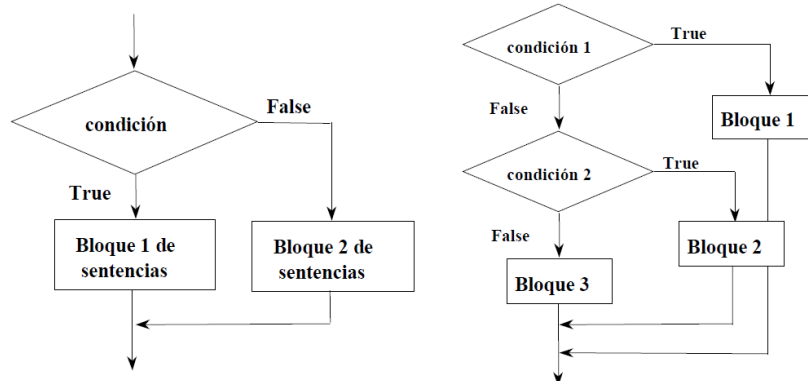
**c) If Anidado.**

Para indicar que se quiere ejecutar uno de varios bloques de sentencias dependientes cada uno de ellos de una condición, la estructura adecuada es la siguiente:

```
If condicion1 Then
    sentencias1
Else If condicion2 Then
    sentencias2
Else
    sentencia-n
End If
```

Si se cumple la **condicion1** se ejecutan las **sentencias1**, y si no se cumple, se examinan secuencialmente las condiciones siguientes hasta **Else**, ejecutándose las sentencias correspondientes al primer **Else If** cuya condición se cumpla. Si todas las condiciones son falsas, se ejecutan las sentencias-n correspondientes a **Else**, que es la opción por defecto.

La siguiente figura presenta esquemáticamente ambas formas de representar estas sentencias:



Por ejemplo:

```
Numero = 53                                'Se inicializa la variable.
If Numero < 10 Then
    Digitos = 1
```

## Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL

```
ElseIfNumero< 100 Then
    'En este caso la condición se cumple (True) luego se ejecuta lo siguiente.
    Digitos = 2
Else 'En el caso en que no se cumplan los dos anteriores se asigna 3
    Digitos = 3
End If
```

### 1.1.2 Sentencia SELECT.. CASE (If Múltiple)

Esta sentencia permite ejecutar una de entre varias acciones en función del valor de una expresión. Es una alternativa a *If...Then... ElseIf* cuando se compara la misma expresión con diferentes valores. Su forma general es la siguiente:

```
Select Case expresion
    Case etiq1
        [sentencias1]
    Case etiq2
        [sentencias2]
    Case Else
        sentenciasn
EndSelect
```

donde *expresiones* una expresión numérica o alfanumérica, y *etiq1*, *etiq2*, ... pueden adoptar las formas siguientes:

- *expresion*
- *expresion To expresion*
- **Is** operador-de-relación *expresion*
- combinación de las anteriores separadas por comas

Por ejemplo,

```
Numero = 8
Select Case Numero
    Case 1 To 5
        Resultado = "Se encuentra entre 1 y 5"
    Case 6, 7, 8
        Resultado = "Se encuentra entre 6 y 8"
    Case Is = 9 , Is = 10
        Resultado = "El valor es 9 o 10"
    Case Else
        Resultado = "El número no se encuentra entre 1 y 10"
EndSelect
```

Cuando se utiliza la forma *expresion To expresion*, el valor más pequeño debe aparecer en primer lugar.

Cuando se ejecuta una sentencia *Select Case*, **Visual Basic** evalúa la *expresion* y el control del programa se transfiere a la sentencia cuya etiqueta tenga el mismo valor que la expresión evaluada, ejecutando a continuación el correspondiente bloque de sentencias. Si no existe un valor igual a la *expresion* entonces se ejecutan las sentencias a continuación de *Case Else*.

## 1.2 Sentencias Repetitivas.

### 1.2.1 Sentencia FOR ... NEXT

La sentencia **For** da lugar a un ciclo o bucle, y permite ejecutar un conjunto de sentencias cierto número de veces. Su forma general es:

```
For variable = expresion1 To expresion2 [Step expresion3]
    [sentencias]
ExitFor
[sentencias]
Next [variable]
```

Cuando se ejecuta una sentencia **For**, primero se asigna el valor de la **expresion1** a la variable y se comprueba si su valor es mayor o menor que la **expresion2**.

En caso de ser menor se ejecutan las sentencias, y en caso de ser mayor el control del programa salta a las líneas a continuación de **Next**. Todo esto sucede en caso de ser la **expresion3** positiva.

En caso contrario (**expresion3** negativa) se ejecutarán las sentencias cuando la variable sea mayor que **expresion2**.

Una vez ejecutadas las sentencias, la variable se incrementa en el valor de la **expresion3**, o en 1 si **Step** no se especifica, volviéndose a efectuar la comparación entre la variable y la **expresion2**, y así sucesivamente.

La sentencia **ExitFor** es opcional y permite salir de un bucle **For ... Next** antes de que éste finalice. Por ejemplo,

```
MyString="Informática "
For Words = 3 To 1 Step -1           '3 veces decrementando de 1 en 1.
    For Chars = Words To Words+4     '5 veces.
        MyString = MyString&Chars   'Se añade el número Chars al
string.                               'El valor de MyString es: Informática 34567 23456 12345
    Next Chars                       'Se incrementa el contador
    MyString = MyString& " "         'Se añade un espacio.
Next Words
```

### 1.2.2 Sentencia DO ... LOOP

Un **Loop (bucle)** repite la ejecución de un conjunto de sentencias mientras una condición dada sea cierta, o hasta que una condición dada sea cierta. La condición puede ser verificada antes o después de ejecutarse el conjunto de sentencias. Sus posibles formas son las siguientes:

```
'Formato 1:
Do [{While/Until} condicion]
    [sentencias]
    [Exit Do]
[sentencias]
Loop

'Formato 2:
Do
    [sentencias]
    [Exit Do]
[sentencias]
Loop [{While/Until} condicion]
```

## Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL

La sentencia opcional **Exit Do** permite salir de una bucle **Do ...Loop** antes de que finalice éste. Por ejemplo,

```
Check = True                                'Se inicializan las variables.
Counts = 0
Do                                           'Empieza sin comprobar ninguna condición.
    Do While Counts < 20                    ' Bucle que acaba si Counts >= 20 o con Exit Do.
        Counts = Counts + 1                'Se incrementa Counts.
        If Counts = 10 Then                 'Si Counts es 10.
            Check = False                  'Se asigna a Check el valor False.
            Exit Do                        'Se acaba el segundo Do.
        EndIf
    Loop
LoopUntil Check = False                     'Salir del "loop" si Check es False.
```

En el ejemplo mostrado, se sale de los bucles siempre con **Counts = 10**. Es necesario fijarse que si se inicializa **Counts** con un número mayor o igual a 10 se entraría en un bucle infinito (el primer bucle acabaría con **Counts = 20** pero el segundo no finalizaría nunca, bloqueándose el programa y a veces el ordenador).

### 1.2.3 Sentencia WHILE ... WEND

Esta sentencia es otra forma de generar bucles que se recorren mientras se cumpla la condición inicial. Su estructura es la siguiente:

```
While condicion
    [sentencias]
Wend
```

Por ejemplo,

```
Counts = 0                                'Se inicializa la variable.
While Counts < 20                          'Se comprueba el valor de Counts.
    Counts = Counts + 1                    'Se incrementa el valor de Counts.
Wend                                       'Se acaba el bucle cuando Counts > 19.
```

En cualquier caso se recuerda que la mejor forma de mirar y aprender el funcionamiento de todas estas sentencias es mediante el uso del **Help de Visual Basic**. Ofrece una explicación de cada comando con ejemplos de utilización.

### 1.2.4 Sentencia FOR EACH ... NEXT

Esta construcción es similar al bucle **For**, con la diferencia de que la variable que controla la repetición del bucle no toma valores entre un mínimo y un máximo, sino a partir de los elementos de un array (o de una colección de objetos). La forma general es la siguiente:

```
ForEach variable In grupo
    [sentencias]
Next variable
```

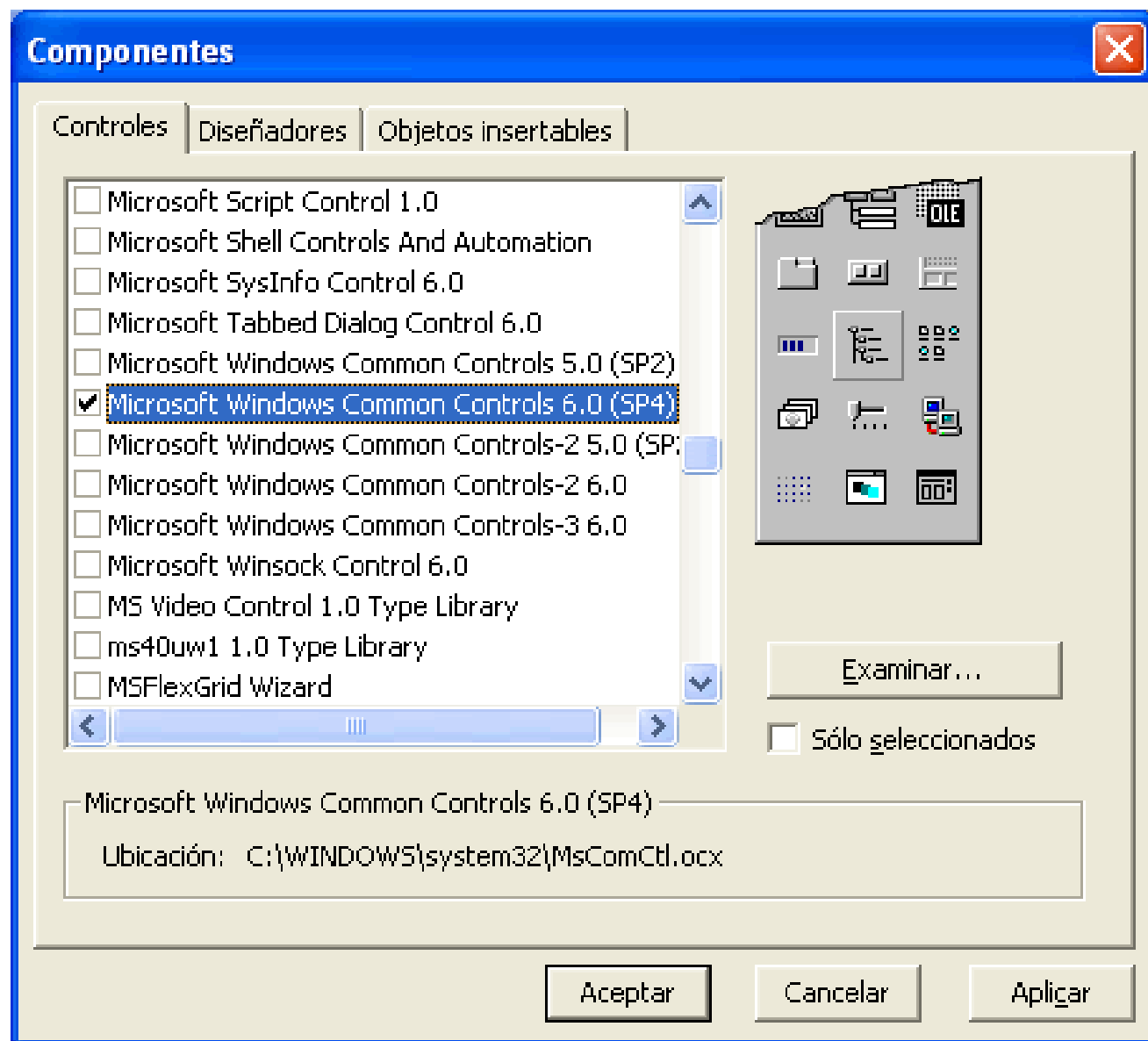
Con arrays **variable** tiene que ser de tipo **Variant**. Con colecciones **variable** puede ser **Variant** o una variable de tipo **Object**. Esta construcción es muy útil cuando no se sabe el número de elementos que tiene el array o la colección de objetos.

## APUNTE: CONTROL PROGRESSBAR

### 1 - Descripción del control

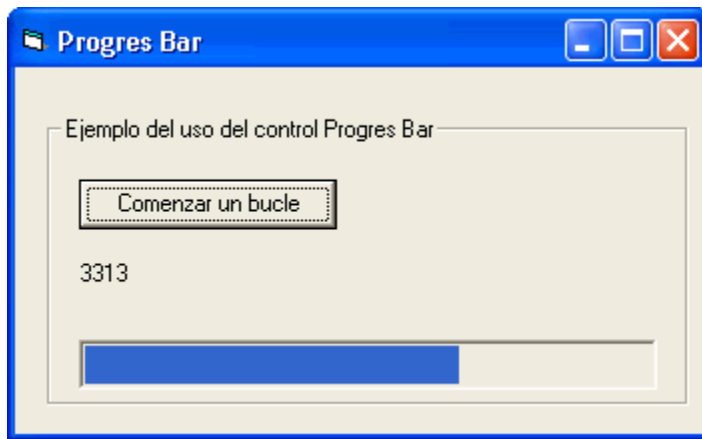
El **control ProgressBar** nos permite medir el progreso de una tarea o proceso en nuestro programa

Este control viene con el conjunto de controles que se incorporan en el ocx MsComctl32.ocx con el nombre de Microsoft Windows Common Control 6





Una vista previa del control ProgressBar:



Si bien es posible crear barras de progreso sin utilizar un control adicional, mediante el ProgressBar podemos implementar de una manera rápida y fácil, mediciones de progreso de una determinada tarea

---

## 2 - Propiedad Value , Max y Min

Este control posee 3 propiedades fundamentales para trabajar en el código a la hora de mostrar dicho progreso en la barra.

Estas tres propiedades principales son:

- **Value**: Propiedad que indica el valor actual de la barra, o mejor dicho del proceso que estamos midiendo
- **Max** : La Propiedad Max establece el valor máximo para la barra o proceso
- **Min** : Establece el **valor mínimo**.

**Nota:** Este control también posee otras propiedades, pero que no son determinantes y que están relacionadas mas a su aspecto gráfico.

## APUNTE: OPERADORES LOGICOS

Son símbolos o palabras que forman expresiones lógicas cuyo resultado siempre será un **valor lógico (True-False)**.

Depende del lenguaje de programación si se presenta con simbolos o palabras.

### Los operadores lógicos son:

- And – Y (Conjuncion) **&&**
- Or – O (Disyuncion) **|| pipeline**  
**||**
- Not – No (Negacion) **!**

### Prioridades o Jerarquia:

1. Not
2. And, Or

### Jerarquia de todos los operadores:

El uso de los operadores lógicos en los lenguajes de programación es para la toma de decisiones en el programa. Para ello se puede hacer mediante dos estructuras de programación:

- Decision (If, Select Case)
- Ciclos (While, Do, For)

Las condiciones en una sentencia if que solo llevan operadores relacionales se consideran **condiciones simples**.

Cuando en una condición en una sentencia If se utilizan Operadores relacionales y lógicos, se consideran **condiciones complejas**.

## TABLA DE VALORES DE VERDAD

### OPERADOR AND

Cuando se utiliza este operador, su resultado será verdadero solo cuando sus **dos operadores** son verdaderos, en cualquier otro caso, el resultado será falso.

| p | q | p and q |
|---|---|---------|
| V | V | V       |

|   |              |                     |
|---|--------------|---------------------|
| 1 | Parentesis   | ()                  |
| 2 | Aritmeticos  | ^,*,/, \,Mod, +, -  |
| 3 | Relacionales | >, >=, <, <=, =, <> |
| 4 | Logicos      | Not, And, Or        |
| V | F            | F                   |
| F | V            | F                   |
| F | V            | F                   |

### OPERADOR NOT

En una expresión lógica que utiliza el operador Not, consta de solo un operando, donde su resultado equivaldrá a invertir su valor lógico.

| p | Not p |
|---|-------|
| V | F     |
| F | V     |

### OPERADOR OR

En expresiones lógicas que utilizan el operador or, su resultado será verdadero cuando al menos uno de sus operandos o ambos son verdaderos.

| p | q | p or q |
|---|---|--------|
| V | V | V      |
| V | F | V      |
| F | V | V      |
| F | F | F      |

## EJERCICIOS DE OPERADORES LÓGICOS

**1. F and V or F=F**

F and V or F  
F or F  
F

**2. V or F and V=V**

V or F and V  
V and V  
V

**3. Not F or V= V**

Not F or V  
V or V  
V

**4. V or F and Not V=F**

V or F and Not V  
V or F and F  
V and F  
F

**5. F and F or Not F=V**

F and F or Not F  
F and F or V  
F or V  
V

**6. (F and V) or Not F=V**

(F and V) or Not F  
F or Not F

F or V  
V

**7. (For V and V) and V or Not F=V**

(F or V and V) and V or Not F

(V and V) and V or Not F

V and V or Not F

V and V or V

V or V

V

**8. Not (F or V and F) and V=V**

Not (F or V and F) and V

Not (V and F) and V

Not F and V

V and V

V

**9. Not (F or V) and Not F or V=V**

Not (F or V) and Not F or V

Not V and Not F or V

F and V or V

F or V

V

**10.V or F and V or Not V=V**

V or F and V or Not V

V or F and V or F

V and V or F

V or F

V

| p | q | Not p and q     | p or q and Not p     | Not (p and q or p) or q     |
|---|---|-----------------|----------------------|-----------------------------|
| V | V | Not V and V = F | V or V and Not V = F | Not (V and V or V) or V = V |
| V | F | Not V and F = F | V or F and Not V = F | Not (V and F or V) or F = F |
| F | V | Not F and V = V | F or V and Not F = V | Not (F and V or F) or V = V |
| F | F | Not F and F = F | F or F and Not F = F | Not (F and F or F) or F = V |

## EJERCICIOS CON SENTENCIA IF

Realiza la instrucción if necesaria para las siguientes situaciones, e indica el resultado que dará la condición que formules con los valores dados.

1. A los alumnos que tengan 0 (cero) faltas.

Sentencia: If faltas = 0 Then

| Faltas | Resultado |
|--------|-----------|
| 4      | F         |
| 0      | V         |
| 1      | F         |

2. A los alumnos que tengan 0 (cero) faltas y calificación aprobada

Sentencia: If faltas = 0 and calif > 5 Then

| Faltas | Calificación | Resultado |
|--------|--------------|-----------|
| 2      | 8            | F         |
| 3      | 5            | F         |
| 0      | 10           | V         |
| 0      | 5            | F         |
| 0      | 12           | V         |

3. A los alumnos que tengan 0 (cero) faltas y calificación aprobada pero que no sea mayor a 10

Sentencia: If faltas = 0 or faltas = 2 and calif > 5 Then

| Faltas | Calificación | Resultado |
|--------|--------------|-----------|
| 2      | 8            | F         |
| 3      | 5            | F         |
| 0      | 10           | V         |
| 0      | 5            | F         |
| 0      | 12           | F         |

4. A los alumnos que tengan 0, 1 o 2 faltas y calificación reprobada.

Sentencia: If faltas = 0 or faltas = 2 and calif <= 5 Then

| Faltas | Calificación | Resultado |
|--------|--------------|-----------|
| 2      | 8            | F         |
| 3      | 5            | F         |
| 0      | 10           | F         |

**Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL**

|   |    |   |
|---|----|---|
| 0 | 5  | V |
| 0 | 12 | F |

5. A los bebés varones de 3 a 6 meses de edad

Sentencia: If sexo = "H" or sexo = "M" and mes>=3 and mes<=6 Then

| Sexo | Edad | Resultado |
|------|------|-----------|
| H    | 5    | V         |
| H    | 6    | V         |
| F    | 4    | F         |

6. A los estudiantes que obtuvieron calificación final de NP o los que obtuvieron NA y mínimo el 80% de asistencias.

Sentencia: If calif = NP or (Calif = NA and Asistencia>= 80 ) Then

| CalFinal | PorcAsistencias | Resultado |
|----------|-----------------|-----------|
| NP       | 90              | V         |
| NP       | 70              | V         |
| NA       | 80              | V         |
| NA       | 75              | F         |

7. A los empleados que son de categoría "A", "B" o "C" y su sueldo mensual sea menor a \$10,000

Sentencia: If E = "a" or E = "b" or E = "c" and sueldo<10000 Then

| Categoria | Sueldo | Resultado |
|-----------|--------|-----------|
|-----------|--------|-----------|

**Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL**

|   |       |   |
|---|-------|---|
| C | 10000 | F |
| A | 9999  | V |
| D | 8500  | F |
| C | 11000 | F |
| B | 7800  | V |

8. A los empleados que son de categoría “A”, “B” o “C” y su sueldo sea menor a \$8,000; o que sean del área de sistemas y su sueldo mensual sea menor a \$10,000.

Sentencia: If E = “a” or E = “b” or E = “c” and sueldo < 8000 or area = “Sistema” and sueldo < 10000 Then

| Categoría | Sueldo | Área         | Resultado |
|-----------|--------|--------------|-----------|
| C         | 8000   | Sistemas     | V         |
| A         | 7999   | Contabilidad | V         |
| D         | 6000   | Compras      | F         |
| B         | 10000  | Sistemas     | F         |
| A         | 9000   | Manufactura  | F         |

9. A los bebés de 12 a 18 meses de edad.

Sentencia: If sexo = “H” or sexo = “M” and mes >= 12 and mes <= 18 Then

| Edad | Resultado |
|------|-----------|
| 19   | F         |
| 12   | V         |
| 15   | V         |

10. A todas las estudiantes mujeres de 18 a 20 años originarias del estado de Morelos

Sentencia: If sexo = “H” or sexo = “M” and edad >= 18 and edad <= 20 and estado = “Morelos” Then

| Sexo | Edad | Estado Nacimiento | Resultado |
|------|------|-------------------|-----------|
| F    | 17   | Morelos           | V         |
| F    | 18   | Morelos           | V         |
| F    | 20   | Morelos           | V         |
| H    | 19   | Morelos           | F         |

11. A todas las casas que se ubican en la colonia “Reforma”, “Tabachines” o “Delicias” de la ciudad de Cuernavaca y su precio no sea mayor a \$5’000,000

Sentencia: If casas = “Reforma” or casa = “Tabachines” or casa = “Delicias” and ciudad = “Cuernavaca” and not precio < 5000000 Then



**Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL**

| Colonia     | Ciudad     | Precio  | Resultado |
|-------------|------------|---------|-----------|
| Tabachines  | Cuernavaca | 5000000 | V         |
| Teopanzolco | Cuernavaca | 3500000 | V         |
| Delicias    | CDMX       | 4000000 | F         |

12. A todas las casas que su superficie sea máximo de 500 m<sup>2</sup> y su costo entre 2 y 3 millones de pesos.

Sentencia: If casas <= 500 and precio < 2000000 precio < 3000000 Then

| Superficie | Costo   | Resultado |
|------------|---------|-----------|
| 500        | 3000000 | V         |
| 600        | 2500000 | F         |

13. A todas las películas de género “terror” realizadas del año 2010 a la fecha.

Sentencia: If película = “Terror” and año >= 2010 and año <= 2019 Then

| Genero  | Año  | Resultado |
|---------|------|-----------|
| Terror  | 2019 | V         |
| Terror  | 2009 | F         |
| Comicas | 2010 | F         |

14. A todas las películas infantiles realizadas por Walt Disney Studios.

Sentencia: If película = “Infantiles” and dirigidas = “Walt Disney Studios” Then

| Genero     | Productor           | Resultado |
|------------|---------------------|-----------|
| Infantiles | Pixar               | F         |
| Comicas    | Walt Disney Studios | F         |
| Infantiles | Walt Disney Studios | V         |

15. A todos los libros de la editorial “McGraw-Hill” y “Trillas” editados en el año 2015 a la fecha.

Sentencia: If libros = “McGraw-Hill” and libros = “Trillas” and año = 2015 and año = 2019 Then

| Editorial   | Edicion | Resultado |
|-------------|---------|-----------|
| Trillas     | 2015    | V         |
| McGraw-Hill | 2019    | V         |
| Trillas     | 2014    | F         |

16. A todos los libros de temas de Computación cuyo costo sea menor a \$500

Sentencia: If libros= "Computacion" and costo <500 Then

| Tema        | Costo | Resultado |
|-------------|-------|-----------|
| Filosofia   | 499   | F         |
| Computación | 499   | V         |
| Cómputo     | 400   | F         |

## EJERCICIOS DE CICLO For (Visual Basic 6.0)

### Sección I.

- 1) Modificar la variable de control de 1 a 100 en incrementos de 1.  
***For i=1 To 100***
- 2) Modificar la variable de control de 100 a 1 en decrementos de 1.  
***For i=100 To 1 Step -1***
- 3) Modificar la variable de control de 7 a 77 en incrementos de 7.  
***For i=7 To 77 Step 7***
- 4) Modificar la variable de control de 20 a 2 en decrementos de 2.  
***For i=20 To 2 Step -2***
- 5) Modificar la variable de control con la siguiente secuencia de valores: 2, 5, 8, 11, 14, 17, 20.  
***For i=2 To 20 Step 3***
- 6) Modificar la variable de control con la siguiente secuencia de valores: 99, 88, 77, 66, 55, 44, 33, 22, 11, 0.  
***For i=99 To 0 Step -11***
- 7) Hacer la sentencia que acceda a los enteros impares del 999 al 1.  
***For i=999 To 1 Step -2***
- 8) El siguiente código debe acceder a los números pares del 2 al 10.  
***For i=2 To 10 Step 2***
- 9) El siguiente código debe acceder a los enteros del 100 al 150  
***For i=100 To 150***
- 10) Escribe instrucciones For que impriman las siguientes secuencias de valores:
  - a) 1, 2, 3, 4, 5, 6, 7  
***For i=1 To 7***
  - b) 3, 8, 13, 18, 23  
***For i=3 To 23 Step 5***
  - c) 20, 14, 8, 2, -4, -10  
***For i=20 To -10 Step -6***
  - d) 19, 27, 35, 43, 51

**For i=19To51 Step 8**

## Sección II.

- 1) Indica cuáles son los valores que toma la variable de control en cada vuelta del ciclo: For x = 2 To 13  
Step 2  
**2, 4, 6, 8, 10, 12**
- 2) Indica cuáles son los valores que toma la variable de control en cada vuelta del ciclo: For x = 5 To 22  
Step 7  
**5, 12, 19**
- 3) Indica cuáles son los valores que toma la variable de control en cada vuelta del ciclo: For x = 3 To 17  
Step 3  
**3, 6, 9, 12, 15**
- 4) Indica cuáles son los valores que toma la variable de control en cada vuelta del ciclo: For x = 1 To 5  
Step 7  
**1**
- 5) Indica cuáles son los valores que toma la variable de control en cada vuelta del ciclo: For x = 12 To 2  
Step -3  
**12, 9, 6, 3**
- 6) Indica cuáles son los valores que toma la variable de control en cada vuelta del ciclo: For x = 5 To -5  
Step -2  
**5, 3, 1, -1, -3, -5**
- 7) Indica cuáles son los valores que toma la variable de control en cada vuelta del ciclo :  
For x = -10 To -20 Step -2  
**-10, -12, -14, -16, -18, -20**
- 8) Indica cuáles son los valores que toma la variable de control en cada vuelta del ciclo :  
For x = 1 To -15 Step -8  
**1, -7, -15**
- 9) Indica cuáles son los valores que toma la variable de control en cada vuelta del ciclo :  
For x = -1 To -20 Step -10  
**-1, -11**
- 10) Indica cuáles son los valores que toma la variable de control en cada vuelta del ciclo :  
For x = -10 To -20 Step 2  
**-10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10, 12, 14, 16.....**

## Aplicación 9. ContraseñaTemporizada

## Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL

La aplicación recibirá una contraseña de máximo 10 caracteres y los cuales no se deben visualizar en la caja de texto, en su lugar deben aparecer \*.

Dará los intentos que sean necesarios para capturar la contraseña correcta, la limitante será el tiempo:

dará 10 segundos para ingresar la contraseña.

Si ingresa una contraseña incorrecta, mandará un aviso en una etiqueta y mostrará una imagen correspondiente al error, ambos se visualizarán en un tiempo de 1 segundo.

Si ingresa la contraseña correcta, mandará un aviso en una caja de mensaje (MsgBox) y mostrará una

imagen alusiva al acierto. El programa se detiene hasta que el usuario selecciona salir de la aplicación.

Si transcurre el tiempo (10 segundos) y no logró ingresar la contraseña correcta, el programa da un aviso

en un MsgBox de que se terminaron las oportunidades y detiene la ejecución del programa.

Se debe utilizar un solo control de Imagen, en la cual se cargará la imagen correspondiente a través de

código y no de la ventana de propiedades.

Debe mostrar una barra de progreso y mostrar un cronómetro para que el usuario sepa el tiempo que le

queda para ingresar la contraseña.

### Temas:

**Controles:** App, Command Button, Label, TextBox, Image, Timer, ProgressBar

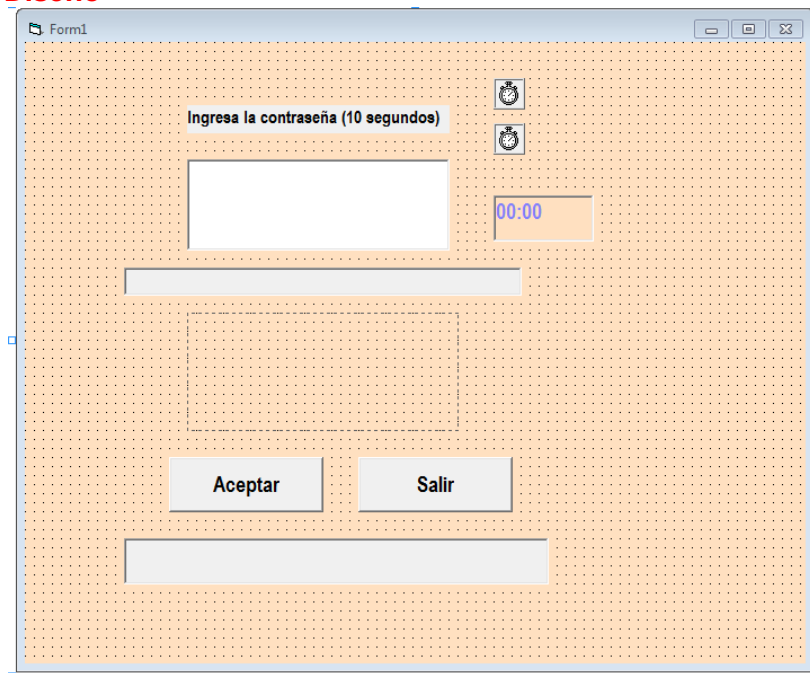
**Propiedades:** Alignment, Caption, Enabled, Interval, MaxLength, Name, PasswordChar, Path, Picture,

Text, Value

Eventos: Click(), Load(), GotFocus(), Timer()

Sentencias: If..Then..Else anidado, Variables Públicas, uso de variables indicadoras

### Diseño



### Codigo

```
Public Flag As Integer
Public Contraseña, Acierto, Error As String
Private Sub cmdAcptar_Click()
'Checa si contraseña es correcta
If txtCntrsña.Text = Contraseña Then
'detiene ambos timers
tmrPrgrso.Enabled = False
tmrPrgrso.Enabled = False
img.Picture = LoadPicture(Acierto) 'Carga imagen acierto
MsgBox "contraseñacorrecta welcome :)"
Else
img.Picture = LoadPicture(Error) 'cargaimgagen de error
'mande mensaje de error
lblMnsje.Caption = "Contraseña incorrecta prro"
End If
End Sub
Private Sub Form_Load()
'configura el timer del segundo
tmrPrgrso.Interval = 1000 'intervalo de un segundo
tmrPrgrso.Enabled = True
'configura el timer del periodo para capturar contraseña
tmrStop.Interval = 11000 'intervalo de diez segundo
tmrStop.Enabled = True
'Configura la caja de texto de la contraseña
txtCntrsña.PasswordChar = "*"
txtCntrsña.MaxLength = 10
'configura la barra de progreso
pbrPrgrso.Max = 10
pbrPrgrso.Min = 0
'configura la ruta de acceso de las imagenes
Acierto = App.Path & "\Correcto.jpg"
Error = App.Path & "\Error.jpg"
lblTmpos.Caption = "00:"
Flag = 0
Contraseña = "joshua"
End Sub
Private Sub tmrPrgrso_Timer()
'Este timer se ejecuta cada segundo para actualizar
'el cronometro y borro algun mensaje e imagen de rror
pbrPrgrso.Value = Flag 'La barra de progreso de avanza
'Configura el cronometro
If Flag < 10 Then
lblTmpos.Caption = "00:0" & Flag
Else
lblTmpos.Caption = "00:" & Flag
End If
lblMnsje.Caption = "" 'Borra si es que hubo mensaje de error
img.Picture = LoadPicture("") 'Borra la imagen
Flag = Flag + 1
End Sub
```

## Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL

```
Private Sub tmrStop_Timer()  
'Este timer se ejecuta si han transcurrido los 10 segundos  
tmrPrgrso.Enabled = False 'Detiene el timer  
MsgBox "Tiempo Terminado"  
Unload Me 'Se descarga el form
```

### Aplicación 10. Decisiones

Recibir tres números enteros, y por medio de botones de comando hacer los siguientes procesos:

1. De los primeros dos números, indicar si son iguales o cual es el mayor.
2. Indicar si los tres números son iguales
3. Indicar si los tres números son diferentes.
4. Si los números son diferentes, ordenarlos de mayor a menor.
5. Indicar si los tres números son iguales, diferentes o si hay dos números iguales, indicar cuáles son.

### Condigo

```
Private Sub Command1_Click()  
    If Text1.Text >= Text2.Text Then  
        If Text1.Text = Text2.Text Then  
Label3.Caption = "Son numeros iguales"  
        Else  
            Label3.Caption = Text1.Text & " es mayor que " & Text2.Text  
        End If  
    Else  
        Label3.Caption = Text2.Text & " es mayor que " & Text1.Text  
    End If  
End Sub  
  
Private Sub Command2_Click()  
'revisa que sean números diferentes, y si es así, los ordena de mayor a menor  
If Text1.Text <> Text2.Text And Text1.Text <> Text3.Text And Text2.Text <> Text3.Text Then  
    'El mayor es el 1  
    If Text1.Text > Text2.Text And Text1.Text > Text3.Text Then  
        If Text2.Text > Text3.Text Then  
'1,2,3  
Label3.Caption = Text1.Text & ", " & Text2.Text & ", " & Text3.Text  
        Else  
'1,3,2  
Label3.Caption = Text1.Text & ", " & Text3.Text & ", " & Text2.Text  
        EndIf  
    'el mayor es el 2  
    ElseIf Text2.Text > Text1.Text And Text2.Text > Text3.Text Then  
        If Text1.Text > Text3.Text Then  
'2,1,3  
Label3.Caption = Text2.Text & ", " & Text1.Text & ", " & Text3.Text  
        Else  
'2,3,1  
Label3.Caption = Text2.Text & ", " & Text3.Text & ", " & Text1.Text  
        EndIf  
    'el mayor es el 3  
    ElseIf Text3.Text > Text1.Text And Text3.Text > Text2.Text Then  
        If Text1.Text > Text2.Text Then  
            '3,1,2  
            Label3.Caption = Text3.Text & ", " & Text1.Text & ", " &  
Text2.Text  
        Else  
            '3,2,1  
            Label3.Caption = Text3.Text & ", " & Text2.Text & ", " &  
Text1.Text
```

## Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL

```
        End If
    End If
Else
    Label3.Caption = "Hay números iguales"
EndIf
End Sub
```

### Private Sub Command3\_Click()

**'evalua si hay dos numeros repetidos, indica cuales son**

```
If Text1.Text = Text2.Text Then
    Label3.Caption = "El primer y segundo número son iguales: " & Text1.Text
ElseIf Text1.Text = Text3.Text Then
    Label3.Caption = "El primer y tercer número son iguales: " & Text3.Text
ElseIf Text2.Text = Text3.Text Then
    Label3.Caption = "El segundo y tercer número son iguales: " & Text2.Text
Else
    Label3.Caption = "No hay numeros repetidos"
End If
End Sub
```

### Private Sub Command4\_Click()

**'Evalua si los tres numeros son iguales o no**

```
If Text1.Text = Text2.Text And Text1.Text = Text3.Text Then
    Label3.Caption = "Los tres numeros son iguales"
Else
    Label3.Caption = "Los tres numeros NO son iguales"
End If
End Sub
```

### Private Sub Command5\_Click()

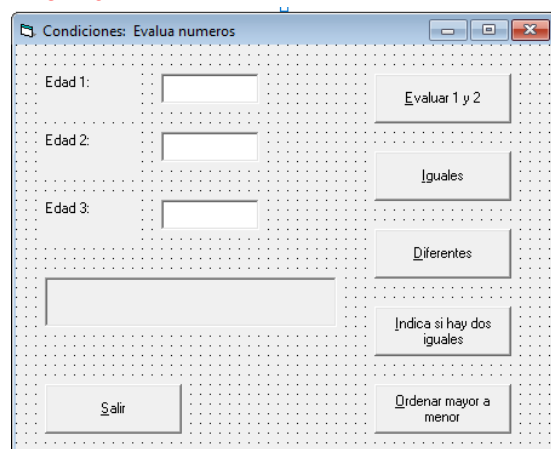
**'Evalua si los tres numeros son diferentes o no**

```
If Text1.Text <> Text2.Text And Text1.Text <> Text3.Text And Text2.Text <> Text3.Text Then
    Label3.Caption = "Los tres numeros SI son numeros diferentes"
Else
    Label3.Caption = "Los tres numeros NO son numeros diferentes"
End If
End Sub
```

### Private Sub Command6\_Click()

```
End
End Sub
```

## DISEÑO



#### FECHAS CON AÑO BISIETO

La aplicación debe permitir introducir una fecha (mes y día) y un número de días a sumar a dicha fecha. Debe devolver como salida la fecha calculada.

#### Características.

- La fecha se introducirá a través de una lista en un *ListBox* para seleccionar el mes, y dependiendo del mes seleccionado, en un *ComboBox* debe permitir seleccionar el día. Tomar en cuenta que hay meses de 31, 30 y 28 días.
- En otro *ComboBox* debe permitir seleccionar el número de días a sumar.
- Debe mostrar la fecha calculada en un *TextBox* protegido en el formato dd/mm/aa.
- El orden de captura de datos es: mes, día y días a sumar
- Al iniciar el programa, los *ComboBox* de los días del mes y los días a sumar deben estar inhabilitados. Cuando se elija el mes, debe habilitarse el *ComboBox* de días del mes. Después de elegir el día del mes, se debe habilitar el *ComboBox* de los días a sumar.



## Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL

- Cuando se elija los días a sumar, debe dar la fecha resultada en el *TextBox* protegido.

Para sumar los días en una fecha, ver el siguiente ejemplo:

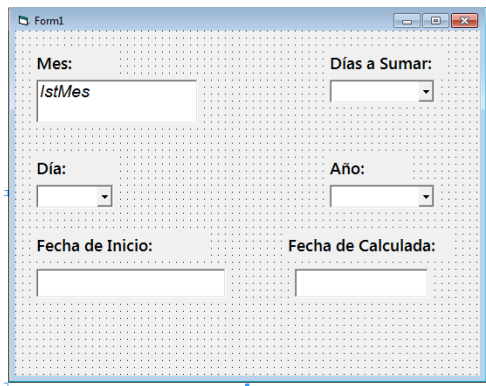
```
Fecha = DateAdd("d", 1, fecha) 'esto suma un dia a tu fecha
```

La función *DateAdd* funciona así:

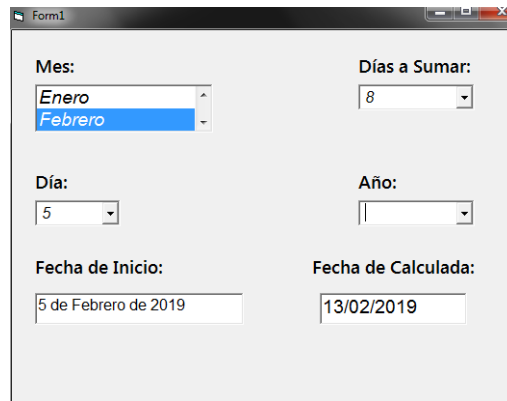
El primer valor es lo que quieres sumar, días, meses... esta es la tabla completa.

```
"yyyy" Year
"q" Quarter
"m" Month
"y" Day of year
"d" Day
"w" Weekday
"ww" Week
"h" Hour
"n" Minute
"s" Second
```

### DISEÑO



### Salida



### CODIGO

Public FechaInicio, FechaFin As Date

#### Private Sub cmbDia\_Click()

Dim Año As Integer

Año = Year(Date)

txtFechaInicioRnta.Text = cmbDia.Text & " de " & IstMes.Text & " de " & Año

FechaInicio = cmbDia.Text & "/" & IstMes.ListIndex + 1 & "/" & Año

cmbDiasRnta.Enabled = True

End Sub

#### Private Sub cmbDiasRnta\_Click()

FechaFin = DateAdd("d", Val(cmbDiasRnta.Text), FechaInicio)

txtFechaFinRnta.Text = FechaFin

End Sub

#### Private Sub Form\_Load()

25 LUIS EDUARDO BAHENA CASTILLO

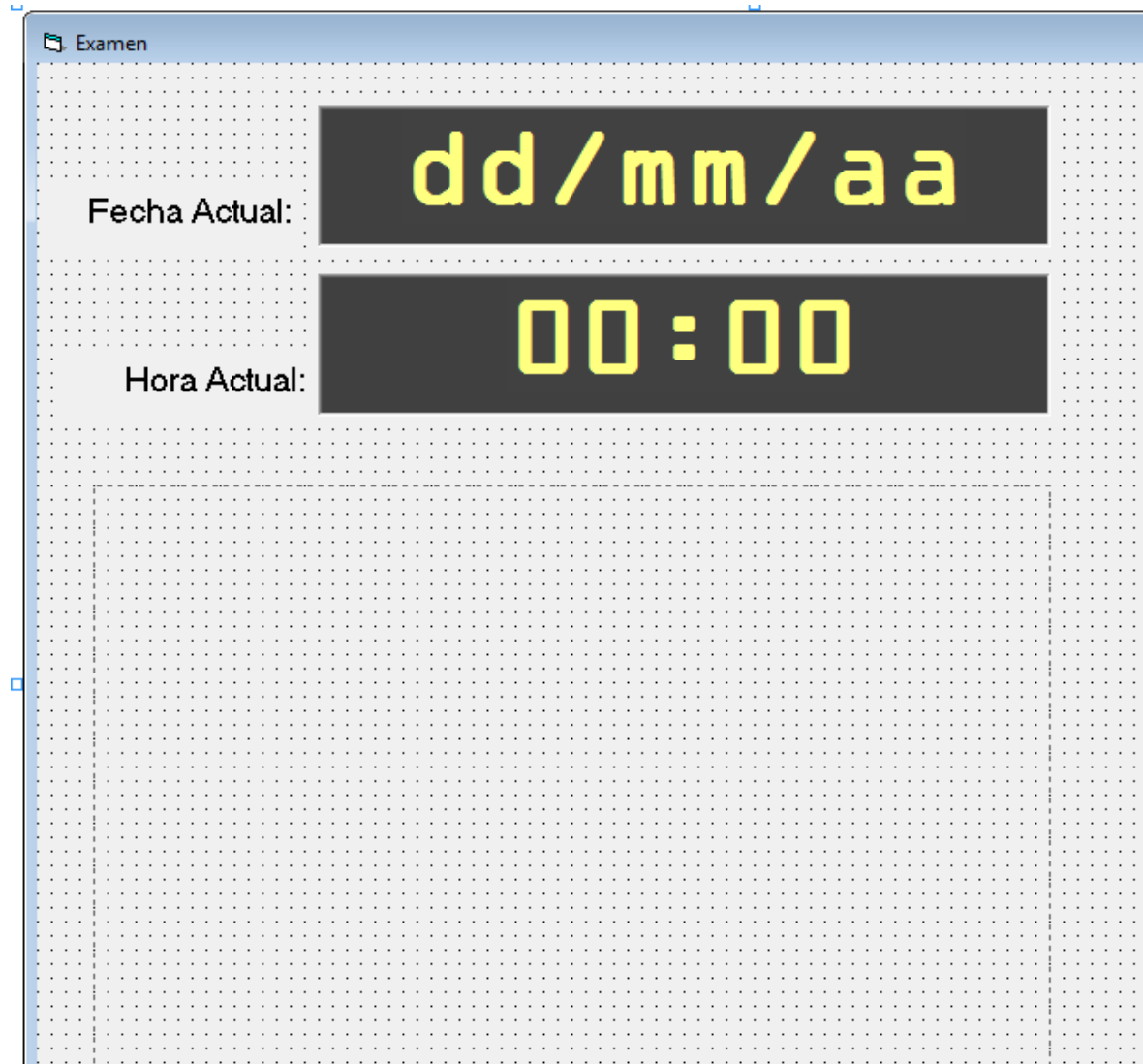
## Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL

```
With IstMes
    .AddItem "Enero"
    .AddItem "Febrero"
    .AddItem "Marzo"
    .AddItem "Abril"
    .AddItem "Mayo"
End With
For i = 1 To 30
cmbDiasRnta.AddItem
Next i
cmbDia.Enabled = False
cmbDiasRnta.Enabled = False
End Sub
```

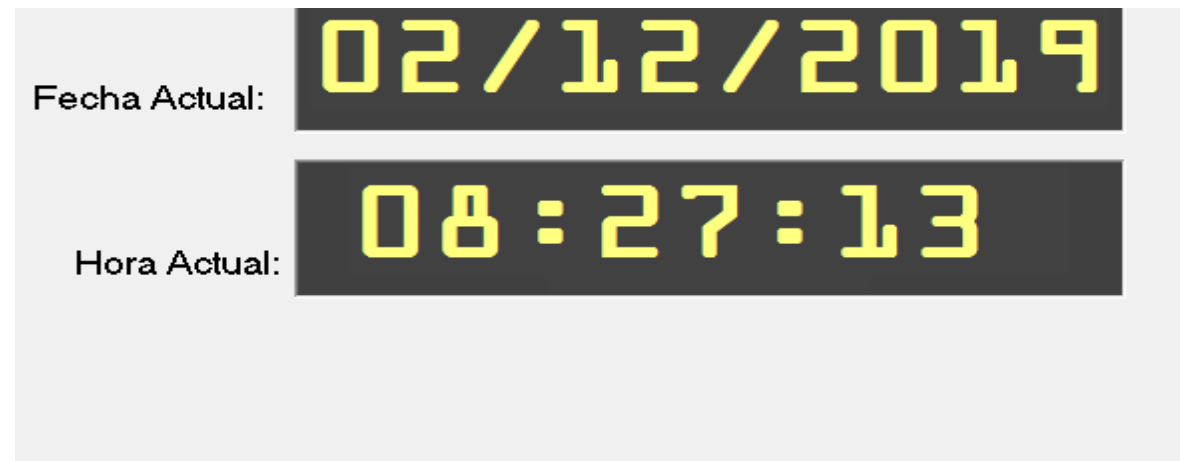
```
Private Sub IstMes_Click()
    Dim LimiteAs Integer
cmbDia.Clear
Select Case IstMes.ListIndex
Case 0, 2, 4    'enero, marzo, mayo
    Limite = 31
    Case 1    'febrero
        Limite = 28
    Case 3    'abril
        Limite = 30
End Select
    For i = 1 To Limite
cmbDia.AddItem
    Next i
cmbDia.Enabled = True
End Sub
```

**Fecha Hora Conllage**

**Diseño**



Salida



## estados

La aplicación mostrará un listado (con *ListBox*) con solo 6 estados de la República, al seleccionar el nombre de uno de ellos, se mostrará la siguiente información:

1. Capital del Estado
2. Extensión territorial (Km<sup>2</sup>)
3. Número de habitantes
4. Mapa del Estado.

## Características.

- La información mostrada no puede ser editada por el usuario, debe estar protegida y los datos numéricos alineados a la derecha con un formato de separador de miles (“###,###”).
- El mapa debe tener un tip de texto para indicar que haciendo doble clic puede agrandar la imagen a pantalla completa, de igual manera debe haber un *CommandButton* que también permita agrandar la imagen.
- Para mostrar el mapa, se debe utilizar un solo control de *Image* y a través de código cargar la imagen del mapa correspondiente.
- El *ListBox* también debe tener un tip de texto para indicar que al seleccionar un estado se mostrará su información.
- Para maximizar la imagen en pantalla completa, se debe cargar otro formulario el cual contendrá la imagen maximizada, el formulario no debe mostrar barra de título (por lo tanto tampoco botones maximizar, restaurar ni cerrar) y debe tener un tip de texto para indicar que se regresa a tamaño normal con otro doble clic, o bien presionando la tecla ESC.
- Cuando el usuario mueva el puntero del mouse sobre el mapa, debe cambiar el puntero de la flecha normal al puntero que indica que puede modificarse el tamaño de la imagen.
- Debe haber un *CommandButton* para salir de la aplicación. Pero en esta ocasión, al presionar el botón, debe mandar un cuadro de mensaje para preguntar al usuario si realmente desea salir de la aplicación y

|   |                      | Superficie         |      | Población                        |
|---|----------------------|--------------------|------|----------------------------------|
|    | Aguascalientes       | 5616               | 0,3  | 1 312 544                        |
|    | Baja California      | 71 450             | 3,6  | 3 315 766                        |
|    | Baja California Sur  | 73 909             | 3,8  | 712 029                          |
|    | Campeche             | 57 507             | 2,9  | 899 931                          |
|    | Chiapas              | 73 311             | 3,7  | 5 217 908                        |
|    | Chihuahua            | 247 455            | 12,6 | 3 556 574                        |
|    | Ciudad de México     | 1495               | 0,1  | 8 918 653                        |
|    | Coahuila de Zaragoza | 151 562            | 7,7  | 2 954 915                        |
|    | Colima               | 5627               | 0,3  | 711 235                          |
|    | Durango              | 123 317            | 6,3  | 1 754 754                        |
|    | Estado de México     | 22 351             | 1,1  | 16 187 608                       |
|    | Guanajuato           | 30 608             | 1,6  | 5 853 677                        |
|    | Guerrero             | 63 596             | 3,2  | 3 533 251                        |
|    | Hidalgo              | 20 813             | 1,1  | 2 858 359                        |
|    | Jalisco              | 78 588             | 4,0  | 7 844 830                        |
|   | Michoacán            | 58 599             | 3,0  | 4 584 471                        |
|  | Morelos              | 4879               | 0,2  | 1 903 811                        |
|  | Nayarit              | 27 857             | 1,4  | 1 181 050                        |
|   | Nombre               | (km <sup>2</sup> ) | %    | Estimación (2015) <sup>123</sup> |
|  | Nuevo León           | 64 156             | 3,3  | 5 119 504                        |
|  | Oaxaca               | 93 757             | 4,8  | 3 967 889                        |
|  | Puebla               | 34 306             | 1,7  | 6 168 883                        |
|  | Querétaro            | 11 699             | 0,6  | 2 038 372                        |
|  | Quintana Roo         | 44 705             | 2,3  | 1 501 562                        |
|  | San Luis Potosí      | 61 137             | 3,1  | 2 717 820                        |
|  | Sinaloa              | 58 200             | 2,9  | 2 966 321                        |
|  | Sonora               | 179 355            | 9,2  | 2 850 330                        |
|  | Tabasco              | 24 731             | 1,3  | 2 395 272                        |
|  | Tamaulipas           | 80 249             | 4,1  | 3 441 698                        |
|  | Tlaxcala             | 4016               | 0,2  | 1 272 847                        |
|  | Veracruz             | 71 826             | 3,7  | 8 112 505                        |
|  | Yucatán              | 39 524             | 2,0  | 2 097 175                        |
|  | Zacatecas            | 75 284             | 3,8  | 1 579 209                        |

## Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL

debe tener un ícono de pregunta (Question) y dos botones para responder “Sí” o “No”. Si a la pregunta responde que “Sí”, salir de la aplicación, en caso contrario, la aplicación debe seguir en ejecución.

### ***Temas:***

**Controles:** App, Command Button, Frame, Label, TextBox, Image, ListBox, Me, PictureBox

**Propiedades:** Alignment, BorderStyle, Caption, KeyPreview, ListIndex, Locked, **MousePointer**, Name, Path, Picture, Stretch, Text, **ToolTipText**, WindowState

**Métodos:** .AddItem, Show

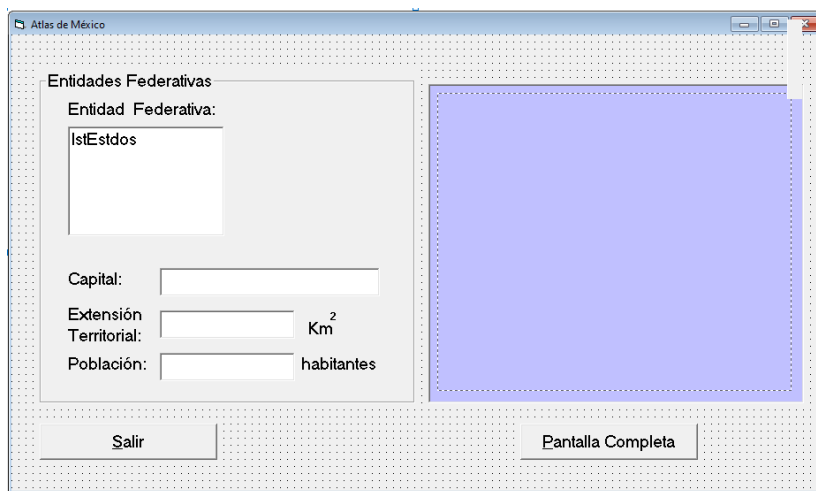
**Eventos:** Click(), **DblClick()**, Load(), **KeyPress()**

### **Sentencias:**

With..End With, *If..Then (Decisión Simple)*, *Select..Case*, *manejo de archivos de imagen*, **Format (formato de los datos)**, **Variables Públicas Globales (de Módulo)**, **LoadPicture()**, **Cuadros de Mensaje MsgBox utilizados como función**, **Unload**, **End**.

**Uso de archivos de módulo (\*.bas)**, **uso de más de un formulario en el proyecto (no modal)**, **constantes de VB (vbYes)**

## **Diseño**



## **SALIDA**





## Aplica la Metodologia de Desarrollo de Aplicaciones con POE– TERCER PARCIAL

CODIGO

```
Private Sub cmdSlir_Click()  
End  
End Sub
```

```
Private Sub Command1_Click()  
frmptllacmplta.Show 1  
End Sub
```

```
Private Sub Form_Load()  
'Llena la lista de Estados  
WithlstEstdos  
    .AddItem "Ciudad de Mexico"  
    .AddItem "Jalisco"  
    .AddItem "Guerrero"  
    .AddItem "Michoacan de Ocampo"  
    .AddItem "Morelos"  
    .AddItem "Nuevo Leon"  
End With
```

```
'Protege los TextBox  
txtCptal.Locked = True  
txtSprfcie.Locked = True  
txtPoblacion.Locked = True
```

```
'Alinea textos numericos a la derecha  
txtSprfcie.Alignment = 1  
txtPoblacion.Alignment = 1
```

```
'Agrega los tips de Texto  
lstEstdos.ToolTipText = "Selecciona un Estado, haz click"  
imgMpa.ToolTipText = "Doble click para ver en pantalla completa"  
imgMpa.MousePointer = 5  
imgMpa.Stretch = True  
End Sub
```

```
Private Sub imgMpa_DbClick()  
frmptllacmplta.Show 1  
End Sub
```

```
Private Sub lstEstdos_Click()  
    Dim ArchivoAs String  
  
    Select Case lstEstdos.ListIndex  
    Case 0 'Ciudad de Mexico  
        txtCptal.Text = "Ciudad de Mexico"  
        txtSprfcie.Text = Format("1495", "###,###")  
        txtPoblacion.Text = Format("8918653", "###,###")  
        Archivo = "Mapa de CDMX.jpg"  
    Case 1 'Jalisco  
        txtCptal.Text = "Guadalajara"  
        txtSprfcie.Text = Format("78588", "###,###")  
        txtPoblacion.Text = Format("7844830", "###,###")  
        Archivo = "Mapa de Jalisco.jpg"  
    Case 2 'Guerrero  
        txtCptal.Text = "Chilpancingo de los Bravo"
```

## Aplica la Metodología de Desarrollo de Aplicaciones con POE– TERCER PARCIAL

```
txtSprfcie.Text = Format("64281", "###,###")
txtPoblacion.Text = Format("3533251", "###,###")
    Archivo = "Mapa de Guerrero.jpg"
    Case 3 'Michoacan de Ocampo
txtCptal.Text = "Morelia"
txtSprfcie.Text = Format("58598", "###,###")
txtPoblacion.Text = Format("4584471", "###,###")
    Archivo = "Mapa de Michoacan.jpg"
    Case 4 'Morelos
txtCptal.Text = "Cuernavaca"
txtSprfcie.Text = Format("4950", "###,###")
txtPoblacion.Text = Format("177227", "###,###")
    Archivo = "Mapa de Morelos.jpg"
    Case 5 'Nuevo Leon
txtCptal.Text = "Monterrey"
txtSprfcie.Text = Format("64924", "###,###")
txtPoblacion.Text = Format("5189970", "###,###")
    Archivo = "Mapa de Nuevo Leon.jpg"
End Select
Mapa = App.Path & "\" & Archivo
imgMpa.Picture = LoadPicture(Mapa)
End Sub
```

## COMENTARIO PERSONAL

Pues me gusto este parcial por que vimos el concepto de la aplicación de Visual Basic, vimos los programas a realizar en la aplicación, todo prácticamente me pareció sencillo, lo único que se complica es la realización de códigos, donde hay veces que me desplegaban en pantalla error, puedo sugerir que me explique mas sobre los códigos para que no vuelvan a generar esos errores, pues este parcial lo sentí MUY DIFICIL, espero poder adquirir mas conocimientos en el siguiente semestre y recopilar conocimientos de este semestre