CARPETA

SISTEMAS TECNOLÓGICOS

(Informática Aplicada)

3° AÑO

CICLO LECTIVO 2024

Lenguajes de Programación

Concepto de lenguaje de programación

Lenguaje artificial que se utiliza para expresar programas de ordenador.

Cada ordenador, según su diseño, 'entiende' un cierto conjunto de instrucciones elementales (lenguaje máquina). No obstante, para facilitar la tarea del programador, se dispone también de lenguajes de alto nivel más fáciles de manejar y que no dependen del diseño específico de cada ordenador. Los programas escritos en un lenguaje de alto nivel no podrán ser ejecutados por un ordenador mientras no sean traducidos al lenguaje propio de éste.

Para definir un lenguaje de programación es necesario especificar:

- Conjunto de símbolos y palabras clave utilizables.
- Reglas gramaticales para construir sentencias (instrucciones, ordenes) sintáctica y semánticamente correctas.
  a) Sintaxis: Conjunto de normas que determinan cómo escribir las sentencias del lenguaje.
  b) Semántica: Interpretación de las sentencias. Indica el significado de las mismas.

Paradigmas de programación

Un paradigma de programación es una colección de patrones conceptuales que moldean la forma de razonar sobre problemas, de formular soluciones y de estructurar programas. Los paradigmas de programación son:

- Programación imperativa
- Programación funcional
- Programación lógica
- Programación orientada a objetos

Programación imperativa:

En este paradigma, un programa es una secuencia finita de instrucciones, que se ejecutan una tras otra. Los datos utilizados se almacenan en memoria principal y se referencian utilizando variables.

```
leer
 (x)
leer
 (y)
resultado = x +
        y
escribir(resulta
       do)
```

Ejemplo de lenguajes que utilizan este paradigma: Pascal, Ada, Cobol, C, Modula-2 y Fortran.

Programación funcional:

Paradigma en el que todas las sentencias son funciones en el sentido matemático del término. Un programa es una función que se define por composición de funciones más simples.

La misión del ordenador será evaluar funciones.

```
predecesor(x)=x-1,
si x>0
sucesor(x)=x+1
suma(x,0) =x
suma (x, y) =sucesor (suma (x, predecesor(y)))

? - suma (3,2)
```

Ejemplo de lenguaje: LISP.

Programación lógica:

En este paradigma un programa consiste en declarar una serie de hechos (elementos conocidos, relación de objetos concretos) y reglas (relación general entre objetos que cumplen unas propiedades) y luego preguntar por un resultado.

Programación orientada a objetos (POO):

El paradigma orientado a objetos (OO) se refiere a un estilo de programación. Un lenguaje de programación orientado a objetos (LOO) puede ser tanto imperativo como funcional o lógico. Lo que caracteriza un LOO es la forma de manejar la información que está basada en tres conceptos:

- Clase. - Tipo de dato con unas determinadas propiedades y una determinada funcionalidad (ejemplo: clase 'persona').
- Objeto. - Entidad de una determinada clase con un determinado estado (valores del conjunto de sus propiedades) capaz de interactuar con otros objetos (ejemplos:
- 'Pedro', 'Sonia',
- Herencia. - Propiedad por la que es posible construir nuevas clases a partir de clases ya existentes (ejemplo: la clase 'persona' podría construirse a partir de la clase 'ser vivo').

Ejemplos de LOO: Smalltalk, C++, Java. Visual.Net

Desarrollo histórico de los lenguajes de programación

Lenguajes máquina (código máquina):

Es el lenguaje que comprende la máquina de forma directa. Internamente, el ordenador representa la información utilizando únicamente unos y ceros. Por tanto, un programa escrito en lenguaje máquina (o código máquina) estará formado por una secuencia finita de unos y ceros.

01011010 10101010 ...
Este lenguaje rara vez se emplea para programar ya que tiene muchos inconvenientes:

- Difícil de escribir y entender.
- Laboriosa modificación y corrección de errores.
- Depende del hardware (distintos ordenadores ⇒ distintos lenguajes máquina).
- Repertorio reducido de instrucciones.

Generaciones de los lenguajes:

1. Primera generación:

- Lenguajes máquina y lenguaje ensamblador.
- Dependen totalmente de la máquina.

2. Segunda generación (finales de los 50 y principios de los 60):

- Fortran: Científico y de ingeniería.
- Cobol: Aplicaciones de procesamiento de datos.
- Algol: Predecesor de lenguajes de 3ª generación.
- Basic: Originalmente para enseñar a programar.

3. Tercera generación (hacia los años 70 – crisis del software):

- Lenguajes de programación estructurada.
- Posibilidades procedimentales y de estructura de datos.

a) De propósito general:
- Pascal: Bloques estructurados, tipificación de datos.
- C: Originalmente para sistemas, gran flexibilidad.
- Ada: para aplicaciones de tiempo real.

b) Orientados a Objetos:
- Smalltalk.
- Eiffel.
- C++.
- Java.

c) Especializados (sintaxis diseñada para una aplicación particular):
- LISP: Demostración de teoremas.
- Prolog: inteligencia artificial.
- Apl: tratamiento de vectores y matrices.

4. Cuarta generación (finales de los años 80):

- Alto nivel de abstracción.
- No son necesarios detalles algorítmicos.
- Ejemplo: Sql (Structured Query Language) orientados a tratamiento de datos.

Traductores

Introducción

Un traductor es un programa que toma como entrada un programa escrito en un lenguaje fuente y lo transforma en un programa escrito en lenguaje máquina.
El proceso de conversión se denomina traducción, que puede realizarse de dos formas diferentes: por interpretación o por compilación.

Compiladores e intérpretes

Intérprete:
Es un programa que toma como entrada un programa escrito en lenguaje fuente y lo va traduciendo y ejecutando instrucción por instrucción (de una en una).
Compilador:
Es un programa que toma como entrada un programa fuente y genera un programa equivalente llamado programa objeto o código objeto.

Algoritmos

Es una formula para resolver un problema.
Es un conjunto de acciones o secuencia de operaciones que ejecutadas en un determinado orden resuelven el problema.
Es una secuencia ordenada de pasos - sin ambigüedades -, repetible, que es solución de un determinado problema.

Las características fundamentales que debe cumplir todo algoritmo son:

- Debe ser preciso e indicar el orden de realización de cada paso
- Debe estar definido (si se repite n veces los pasos se debe obtener siempre el mismo resultado)
- Debe ser finito (debe tener un número finito de pasos)
- Es independiente del lenguaje de programación que se utilice

1. La definición de un algoritmo debe describir tres partes Entrada, Proceso, Salida.
2. La programación es adaptar el algoritmo a la computadora en el lenguaje que corresponda.
3. El algoritmo es independiente según donde lo implemente.
4. El algoritmo trata de resolver problemas mediante programas.

Fases:

- Análisis preliminar o evaluación del problema: Estudiar el problema en general y ver que parte nos interesa.
- Definición o análisis del problema: El objetivo de ésta fase es comprender el problema para lo cual como resultado tenemos que obtener la especificación de las entradas y salidas del problema. Tiene que quedar claro que entra y que sale, las posibles condiciones o restricciones, ...
- Diseño del algoritmo: Diseñar la solución. Una vez comprendido el problema se trata de determinar qué pasos o acciones tenemos que realizar para resolverlo.

Como criterios a seguir a la hora de dar la solución algorítmica hay que tener en cuenta:

1. Si el problema es bastante complicado lo mejor es dividirlo en partes más pequeñas e intentar dividirlo en partes más pequeñas e intentar resolverlas por separado. Esta metodología de "divide y vencerás" también se conoce con el nombre de diseño descendente. Las ventajas de aplicar esto son:
   - Al dividir el problema en módulos o partes se comprende más fácilmente.
   - Al hacer modificaciones es más fácil sobre un módulo en particular que en todo el algoritmo.
   - En cuanto a los resultados, se probarán mucho mejor comprobando si cada módulo da el resultado correcto que si intentamos probar de un golpe todo el programa porque si se produce un error sabemos en que módulo ha sido.
2. Una segunda filosofía a la hora de diseñar algoritmos es el refinamiento por pasos, y es partir de una idea general e ir concretando cada vez más esa descripción hasta que tengamos algo tan concreto para resolver. Pasamos de lo más complejo a lo más simple.

La representación de los algoritmos:

Una vez que tenemos la solución hay que implementarla con alguna representación. Las representaciones más usadas son los diagramas y el seudocódigo.

Procedimientos

Son los pasos que definen el empleo especifico de cada elemento del sistema o el contexto procedimental en que reside el sistema. Los sistemas basados en computadoras Intentan poner orden a un desarrollo de sistemas e intentan poner al software en su contexto por lo tanto establece enlaces que unen al software con los elementos de un sistema basado en computadoras. El papel del ingeniero de sistemas es definir los elementos de un sistema especifico basado en computadora en el contexto de la jerarquía global de sistemas. Por lo tanto, examinaremos las tareas que constituyen los sistemas de computación.

En el proceso cada elemento se implementa especificando los componentes técnicos que dan funcionalidad necesaria a un elemento en el contexto software, un componente podría ser un programa de computadora, un componente de programa reutilizable, un modulo, una clase u objeto o incluso una sentencia de lenguaje de programación.

Sintaxis y Semántica:

- La sintaxis de un lenguaje define como se pueden poner juntos símbolos, palabras reservadas, e identificadores para hacer un programa válido.
- La semántica de un constructor de un lenguaje es el significado del constructor; ella define su papel en un programa.
- Un programa sintácticamente correcto no implica que sea lógicamente (semánticamente) correcto.
- Los programas son más fáciles de construir y codificar cuando están constituidos por componentes separados.
- Un componente software puede considerarse como cualquier elemento de software que transforma una entrada en una salida.

Declaración:

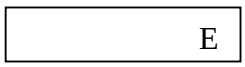
Hay dos clases de declaraciones:

- Implícitas. Se supone que ocurren en un lugar del texto como consecuencia de las semánticas de otra construcción, como por ejemplo una cláusula de contexto.
- Explícitas. Aparecen en el texto del programa, como la variable i del ejemplo anterior. Sintácticamente son declarative_item's. Toman dos formas:
    - Las denominadas declaraciones básicas como las declaraciones de tipo, de variables (objetos), excepciones o las especificaciones de procedimientos y paquetes. Sintácticamente basic_declaration_item.
    - Los cuerpos de los procedimientos y de los paquetes. Sintácticamente body.

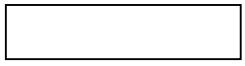
Diagramación:

Una vez comprendido el problema, se hace una representación gráfica de los pasos a seguir para resolverlo. Esta representación se llama diagramación. -
Los diagramas son un conjunto de símbolos que han convenido de distintas maneras distintos autores. En este curso adoptaremos los diagramas de NASSI-SCHNEIDERMAN, más conocidos como diagramas de CHAPIN; que permiten representar adecuadamente las estructuras de control de los lenguajes estructurados como Visual.

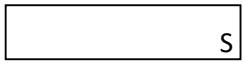
Veamos algunos símbolos:



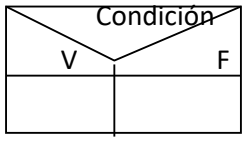
Lectura o entrada de datos



Ordenes u operaciones



Salida de datos y/o resultados



Pregunta o comparación, Decisión simple

Diagramas de nassi - schneiderman

El diagrama N-S o también conocido como diagrama de Chapín es una técnica de especificación de algoritmos que combina la descripción textual, propia del pseudocódigo, con la representación gráfica del diagrama de flujo.
El diagrama N-S cuenta con un conjunto limitado de símbolos para representar los pasos del algoritmo, por ello se apoya en expresiones del lenguaje natural; sin embargo, dado que el lenguaje natural es muy extenso y se presta para la ambigüedad, solo se utiliza un conjunto de palabras, a las que se denomina palabras reservadas.
Las palabras reservadas más utilizadas son:
Inicio Fin Leer Escribir Mientras Repita Hasta Para
Incrementar Decrementar Hacer Función
Entero Real Caracter Cadena
Lógico Retornar
Los símbolos utilizados en el diagrama de Chapin son corresponden a cada tipo de estructura. Dado que se tienen tres tipos de estructuras, se utilizan tres símbolos. Esto hace que los procesos del algoritmo sean más fáciles de representar y de interpretar.

Instrucciones

Definición:

Son aquellas que se ejecutan una después de otra. Se tienen tres tipos de instrucciones secuenciales: la declaración de variables, asignación, instrucción Leer e instrucción Escribir.
La mayoría de algoritmos actúan sobre un conjunto de datos suministrados por el usuario y se espera que a partir de dichos valores y desarrollando los procesos programados se genere información de salida o resultados.

Declaración de variables

Teniendo en cuenta la compatibilidad con la mayoría de los lenguajes, se recomienda que desde el diseño del programa se utilice una forma determinada para la declaración de las variables. Esta consiste en escribir el tipo de datos y la lista de identificadores que se tendrán de dicho tipo, separando cada identificador por medio de comas (,). Para mejorar la claridad de la declaración se puede colocar dos puntos (:) para separar el tipo de datos de la lista de identificadores.
Ejemplo:

- Entero: edad
- Real: estatura, peso, sueldo
- Cadena: nombre, dirección

Aunque algunos lenguajes de programación permiten declarar las variables en el momento en que se las necesita, es aconsejable, en favor de los buenos hábitos de programación, siempre declarar las variables antes de utilizarlas y el sitio más adecuado es el inicio del programa o de la función.

Asignación

Asignar un valor a una variable equivale a decir que se guarda dicho valor en la posición de memoria reservado para la variable en mención. Por lo tanto, para poder realizar una asignación es necesario primero haber declarado una variable, con lo cual se reserva un espacio de memoria suficiente para guardar un dato del tipo especificado.
Una expresión de asignación tiene la forma:
Variable = expresión
Donde la expresión puede estar formada por un valor, por un conjunto de valores y operadores o por una función.
Ejemplos:

- Edad = 10
- Estatura = 1.80
- Resultado = 2*3

Donde edad y resultado son variables de tipo entero y estatura de tipo real que se supone declaradas previamente.
Una asignación tiene tres partes, una variable, el signo igual y la expresión cuyo valor se asigna a la variable. La variable siempre va a la izquierda del igual, mientras que la expresión siempre estará a la derecha.
Ejemplos:

- Entero: X, Y
- X = 10
- Y = X * 2 + 8

En este ejemplo, la variable Y contendrá el valor 28.

Instrucción Leer

La instrucción LEER se utiliza para enviar información desde un dispositivo de entrada de datos hacia la memoria. En la memoria los datos son ubicados mediante el identificador (nombre de variable) utilizado como complemento de la instrucción LEER.
En diagrama N-S la instrucción de entrada se representa así:
Leer <lista de identificadores de variables>
Ejemplo:
Leer a, b
Donde "a" y "b" son las variables que recibirán los valores y que deben haberse declarado previamente.

Instrucción Escribir

Esta instrucción permite enviar datos desde la memoria hacia un dispositivo de salida como la pantalla o la impresora. La información que se envía puede ser constante o también el contenido de variables.

Escribir <lista de constantes y variables>
Ejemplo:
Escribir a, b
Cuando se escriben más de una variable es necesario separarlas con comas (,) y los mensajes se escriben entre comillas dobles " ". Si una variable es escrita entre comillas se mostrará el identificador y no el contenido.

Ejemplos:

1. Diseñar un algoritmo para calcular el área y el perímetro de un rectángulo
Definición del problema
Calcular área y perímetro de un rectángulo
Análisis del problema
Para desarrollar este problema es necesario conocer las fórmulas para obtener tanto el área como el perímetro de un rectángulo.
Sea b = base y h = altura, las fórmulas a utilizar son:
Área = b * h
Perímetro = 2 * (b + h)
Datos de entrada: b y h (base y altura)
Datos de salida: área y perímetro
Procesos: área = b * h
Perímetro = 2 * (b + h)

Diseño de la solución

Inicio
Entero: b, h, a, p
Leer b, h
a = b * h
p = 2 (b + h)
Escribir "área:", a
Escribir "perímetro:", p
Fin algoritmo

2. Un maestro desea saber que porcentaje de hombres y que porcentaje de mujeres hay en un grupo de estudiantes.
Definición del problema
Calcular porcentaje de hombres y mujeres en un grupo
Análisis del problema
Datos a tener en cuenta:
Número hombres
Número mujeres
Total, estudiantes
Porcentaje hombres
Porcentaje mujeres
Datos de entrada: número hombres, número mujeres
Datos salida: porcentaje hombres, porcentaje mujeres
Procesos: tot estudiantes = núm. hombres + núm. Mujeres
Porc.hombres = núm. Hombres / tot estudiantes*100
Porc.hombres = núm. mujeres / tot estudiantes*100

Diseño de la solución

Inicio

Entero: numm, numh, totest
Real: porch, porcm
Leer numm, numh
totest = numm + numh
porcm = numm / totest * 100
porch = numh / totest * 100
Escribir "porcentaje mujeres:", porcm
Escribir "porcentaje hombres:", porch
Fin algoritmo

Estructuras de decisión

Las estructuras de decisión o también llamadas de selección permiten que el algoritmo tome decisiones y ejecute u omita algunos procesos dependiendo del cumplimiento de una condición.
Se pueden manejar tres tipos de decisiones: simple, doble y múltiple.
Decisión simple y doble
Una decisión es simple, cuando solo se tiene determinado los pasos a seguir si el resultado de la condición es verdadero, mientras que, si es falso, la ejecución del algoritmo continúa después de la estructura condicional.
Una decisión cuando se tiene un curso de acción para el caso que el resultado de la comparación sea verdadero y otro para cuando sea falso.
En diagrama de Chapin el símbolo para representar una decisión es el siguiente:
Para ver el gráfico seleccione la opción "Descargar" del menú superior
Obsérvese que en la parte inferior se tienen los cuadros que indican dos posibilidades de acción, el conjunto de instrucciones 1 o el conjunto de instrucciones 2, solo uno de los dos. Si se tratase de una decisión simple, solo se tendrá instrucciones en las cajas que se ubican debajo de la cláusula Si, mientras que las que están bajo No estarán vacías.
Ejemplos:

1. Se desea un algoritmo para obtener el valor absoluto de un número

Definición del problema
Encontrar el valor absoluto de un número
Análisis del problema
Para encontrar el valor absoluto del número es necesario recordar que para los enteros positivos el valor es el mismo, mientras que para los enteros negativos es necesario cambiarlos de signo.
Datos de entrada: número
Datos de salida: valor absoluto
Proceso: número = número * (-1)
Diseño de la solución

Inicio	
Entero: num, absnum	
Leer num	
num < 0 Si	No
absnum = num * (-1)	absnum = num
Escribir "Valor absoluto: ", absnum	
Fin algoritmo	

2. Dados dos números ¿cuál es mayor? y ¿cuál es menor?
Definición del problema: Identificar el mayor y menor de dos números
Análisis del problema:
Datos de entrada: num1, num2
Datos salida: mayor, menor
Proceso: comparación
Diseño de la solución

<table>
<tr><td colspan="3">Inicio</td></tr>
<tr><td colspan="3">Entero: n1, n2</td></tr>
<tr><td colspan="3">Leer n1, n2</td></tr>
<tr><td colspan="3">n1 = n2
Si No</td></tr>
<tr><td rowspan="3">Escribir
"Números iguales"</td><td colspan="2">n1 > n2
Si No</td></tr>
<tr><td>Escribir
n1, "Mayor"</td><td>Escribir
n2, "Mayor"</td></tr>
<tr><td>Escribir
n2, "Menor"</td><td>Escribir
n1, "Menor"</td></tr>
<tr><td colspan="3">Fin algoritmo</td></tr>
</table>

Decisión múltiple

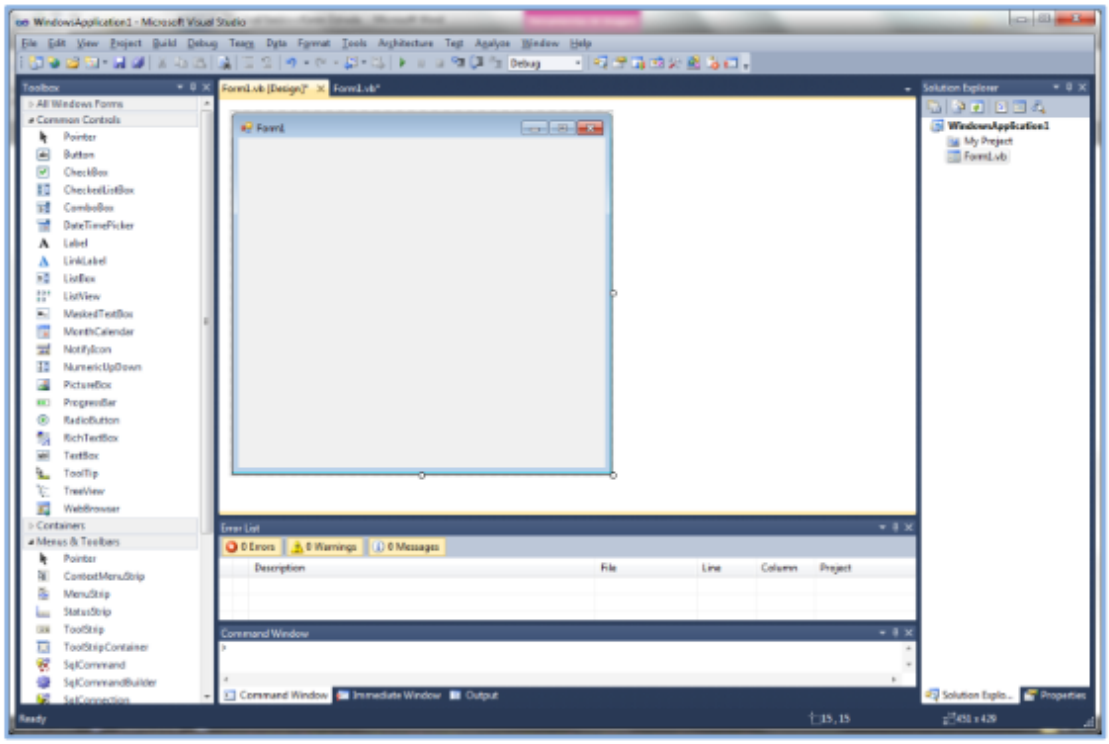
Muchas decisiones deben tomarse, no solo entre dos alternativas, sino de un conjunto mayor. Estos casos bien pueden solucionarse utilizando condicionales dobles anidados; sin embargo, en favor de la claridad del algoritmo y la facilidad para el programador, es mejor utilizar una estructura de decisión múltiple, la cual es fácil de llevar a un lenguaje de programación, ya que éstos incluyen alguna instrucción con este fin.
La decisión múltiple determina el valor de una variable y dependiendo de éste sigue un curso de acción. Es importante tener en cuenta que solo se verifica la condición de igualdad entre la variable y la constante.
En diagrama N-S la estructura de selección múltiple tiene la forma:

<table>
<tr><td colspan="5">variable
1 2 3 … otro</td></tr>
<tr><td>Acción
1</td><td>Acción
2</td><td>Acción
3</td><td>Acción
n</td><td>Acción
m</td></tr>
</table>

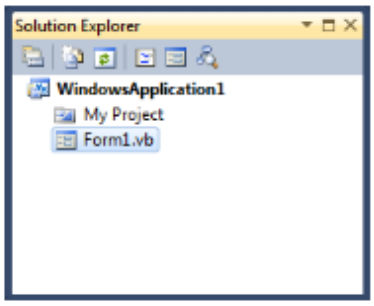
ENTORNO DE VISUAL BASIC:

Después de haber creado un Proyecto de Windows Forms Application, se mostrará la ventana de Visual Basic:

VENTANA EXPLORADOR DE PROYECTO

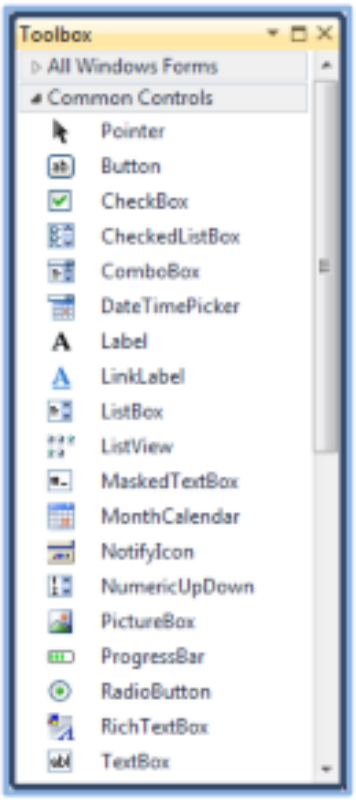


1.6.1. VENTANA EXPLORADOR DE PROYECTO



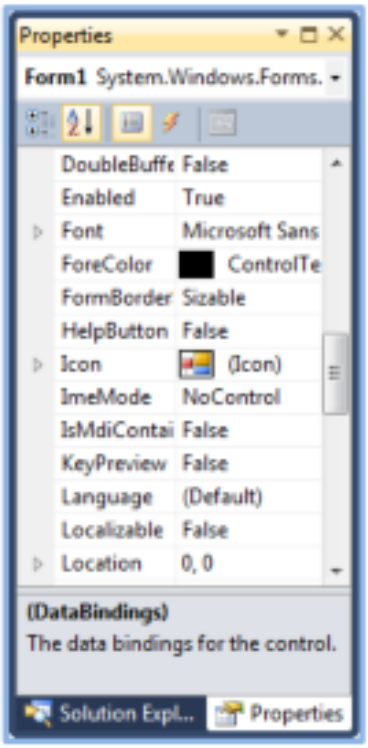
CTRL+R,
CTRL+ALT+L

1.6.2. CUADRO DE HERRAMIENTAS



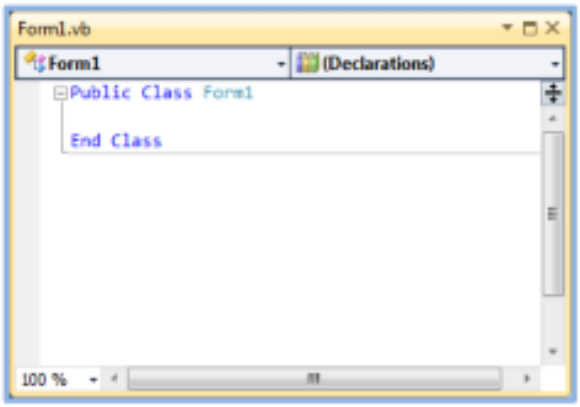
CTRL+ALT+X

1.6.3. VENTANA DE PROPIEDADES



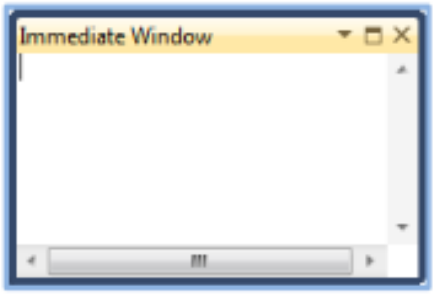
F4

1.6.4. VENTANA EDITOR DE CÓDIGO



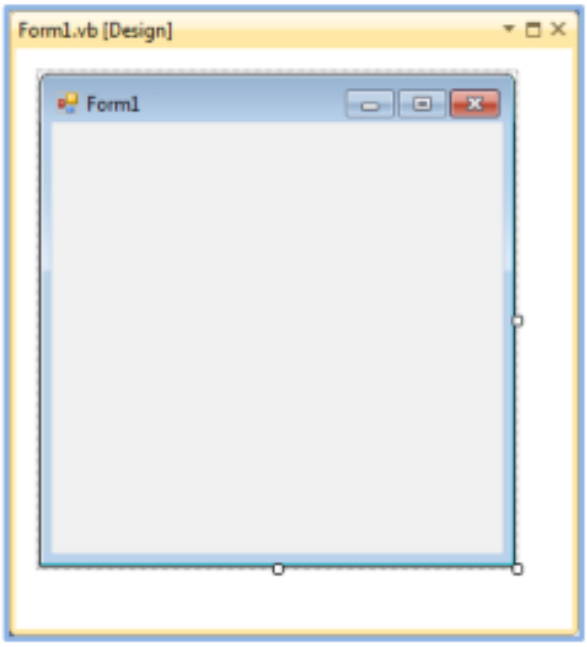
F7

1.6.5. VENTANA DE DEPURACIÓN



CTRL+G

1.6.6. VENTANA DEL FORMULARIO



SHIFT+F7

1.7. TERMINOLOGÍA

1.7.1. OBJETOS
Instancia de la clase, tiene propiedades atributos.

1.7.2. CLASE
Concepto, idea, las características y comportamientos comunes de los objetos.

1.7.3. PROPIEDADES
Características de los objetos, calificativo.

1.7.4. MÉTODOS
Se programa.

1.7.5. EVENTOS
Es una acción que se aplica a los objetos.

1.8. ALGUNOS OBJETOS Y CONTROLES

- Formularios (Form)
- Botones de comando (Button)
- Etiquetas (Label)
- Cuadros de textos (TextBox)
- CheckBox
- RadioButton
- ListBox

1.9. ALGUNAS PROPIEDADES

- Name (nombre)
- Caption (titulo)
- Text (texto)
- Font (fuente)
- Fore color (color de primer plano)
- Backcolor (color de fondo)
- Enabled (disponible)

1.10. ALGUNOS MÉTODOS

1.10.1. SET FOCUS (ENTREGAR EL ENFOQUE)

Este método se utiliza para hacer que un objeto reciba el enfoque. Este método es uno de los más usados para los controles de Visual Basic 6.0.

1.10.2. DRAG

Inicia, termina o cancela una operación de arrastre de cualquier control, excepto loscontroles Line, Menu, Shape, Timer o CommonDialog.

1.10.3. MOVE

Se utiliza para mover un control o formulario, especificando sus coordenadas (Top, Left)y su tamaño (Width, Height).

1.11. ALGUNOS EVENTOS

1.11.1. CLICK

Al hacer click. Ocurre cuando el usuario presiona y suelta un botón del mouse sobre un objeto.

1.11.2. DBLCLICK

Ocurre cuando el usuario presiona y suelta dos veces un botón del mouse sobre unobjeto.

1.11.3. DRAGDROP

Ocurre como resultado de arrastrar y soltar con el mouse un control sobre un determinado tipo de objeto.

1.11.4. KEYDOWN

Ocurre cuando el usuario mantiene presionada una tecla.

1.11.5. KEYUP

Ocurre cuando el usuario termina la operación de pulsar una tecla. Se podría decir, que este evento ocurre precisamente al terminar el evento KeyDown.

1.11.6. KEYPRESS

Ocurre como resultado de presionar y soltar una tecla.

1.11.7. MOUSEDOWN

Ocurre cuando el usuario presiona un botón del mouse, pero a diferencia del evento.

1.11.8. MOUSEUP

El evento MouseUp se produce cuando el usuario suelta el botón del mouse. Es un compañero útil a los eventos MouseDown y MouseMove.

1.11.9. MOUSEMOVE

Este evento ocurre mientras el usuario mueve o desplaza el puntero del mouse sobre un objeto.

1.11.10. CHANGE

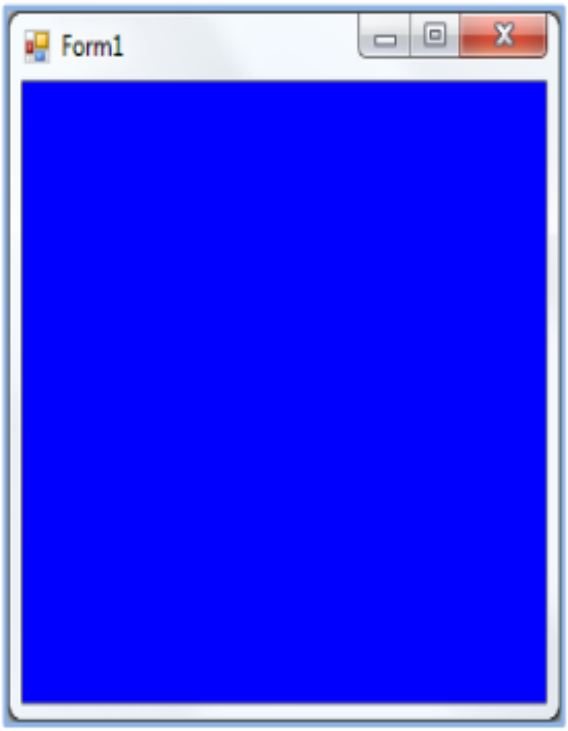
Al cambiar

1.11.11. LOAD

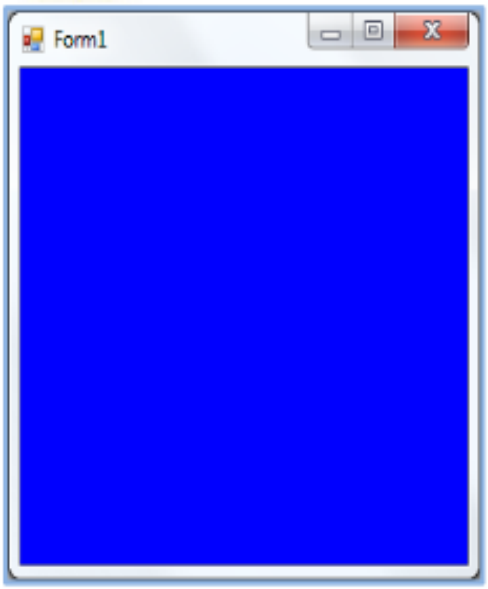
Al cargarse en memoria

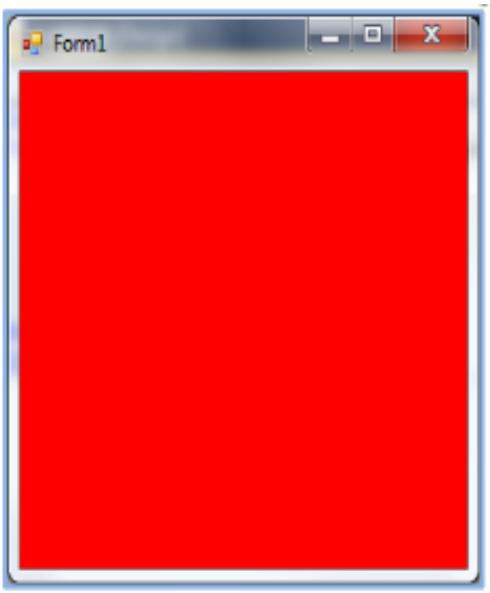
EJERCICIOS

- Cambiar de color al formulario con el evento doubleclick. Programar el cambio del color de fondo del formulario al dar doble click. Cambia al dar doble click, cambia de azul a rojo y de rojo a azul.

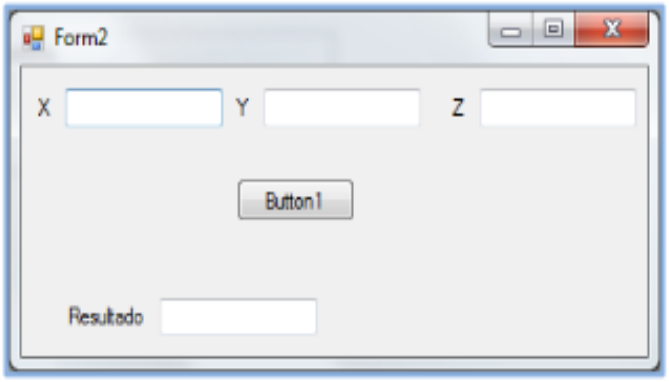



```
Public Class Form1
    Private Sub Form1_MouseDoubleClick(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles Me.MouseDoubleClick
        If Me.BackColor = Color.Blue Then
            Me.BackColor = Color.Red
        ElseIf Me.BackColor = Color.Red Then
            Me.BackColor = Color.Blue
        End If
    End Sub
End Class
```

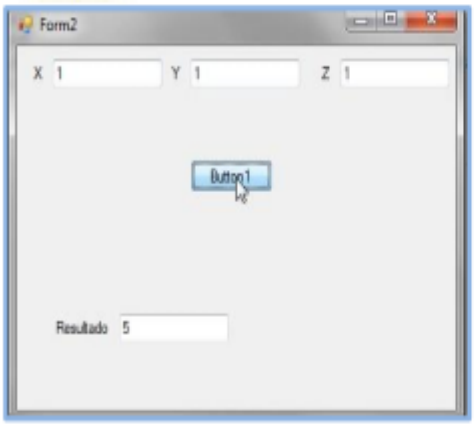




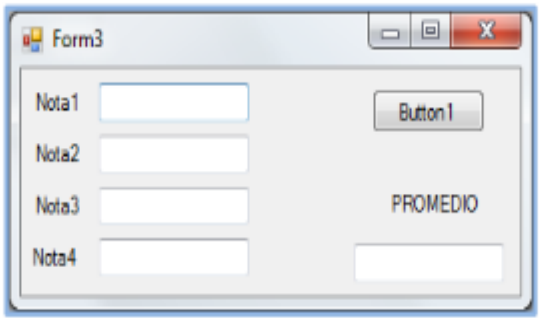
- Resolviendo una ecuación con tres variables




```
Public Class Form2

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim x, y, z, r As Double
        x = TextBox1.Text
        y = TextBox2.Text
        z = TextBox3.Text
        r = ((2 * Math.Pow(x, 4)) + 3 * Math.Pow(x, 2) * Math.Pow(y, 2)) / (y +
(Math.Sqrt(4 * x * y * Math.Pow(z, 2))) - 2 * z)
        TextBox4.Text = r

    End Sub
End Class
```



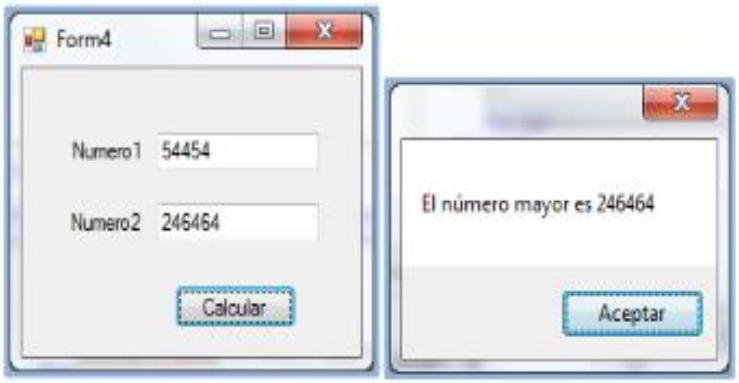
- Calcular promedios eliminando la menor nota




```
Public Class Form3

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim n1, n2, n3, n4, v, p As Double
        n1 = TextBox1.Text
        n2 = TextBox2.Text
        n3 = TextBox3.Text
        n4 = TextBox4.Text
        v = n1
        If n2 < v Then
            v = n2
        End If
        If n3 < v Then
            v = n3
        End If
        If n4 < v Then
            v = n4
        End If
        p = (n1 + n2 + n3 + n4 - v) / 3
        TextBox5.Text = p
    End Sub
End Class
```

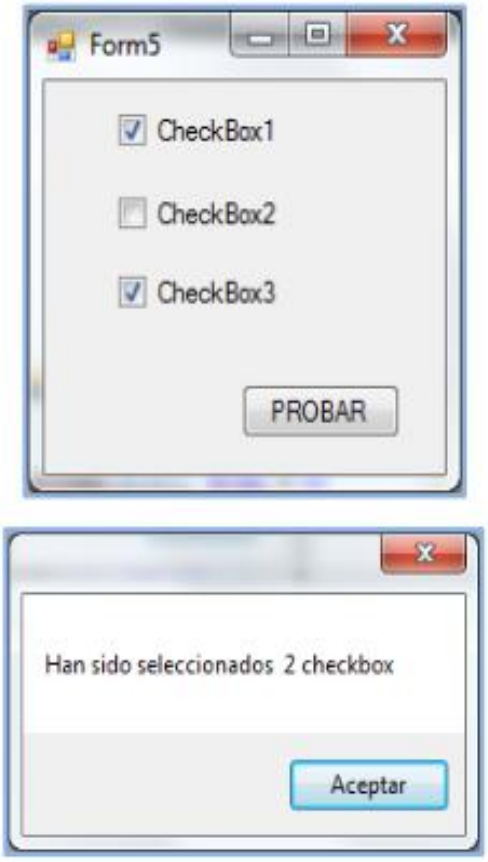
- Calculando el número mayor


```
Public Class Form4
    Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
        Dim n1, n2 As Double
        n1 = TextBox1.Text
        n2 = TextBox2.Text
        If n1 > n2 Then
            MessageBox.Show("El número mayor es " & n1)
        Else
            MessageBox.Show("El número mayor es " & n2)
        End If
    End Sub
End Class
```



- Contando la cantidad de checkbox seleccionados


```
Public Class Form5

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim C As Integer
        C = 0
        If CheckBox1.Checked Then
            C = C + 1

            If CheckBox2.Checked Then
                C = C + 1
            End If
            If CheckBox3.Checked Then
                C = C + 1
            End If
            MessageBox.Show("Han sido seleccionados " + Str(C) + " checkbox")
        End If
    End Sub
End Class
```


Form5
CheckBox1
CheckBox2
CheckBox3
PROBAR
Han sido seleccionados 2 checkbox
Aceptar

Estructura de decisión: El IF

El if evalúa si una expresión es verdadera o falsa y ejecuta un grupo de instrucciones según el resultado de la evaluación. ¿Cómo funcionan estas condiciones?:
"Si mañana llueve voy al cine, si no voy a jugar al golf".
Acá evaluamos una condición, si es verdadero que mañana llueve, voy a ir al cine, pero si no se cumple la condición, o sea, que mañana no llueve voy a ir a jugar al golf, divertidísimo.
Entonces siempre que tengamos que ejecutar cierta acción o no en base a una condición vamos a usar el If.

Sintaxis

En Visual Basic la sintaxis es la siguiente:

```
If condición Then
      Sentencias
      Sentencias
Else
      Sentencias
      Sentencias
End If
```

Entonces donde dice condición ponemos lo que queremos evaluar. Si se cumple irá al primer parte del IF, si no irá a la parte del Else ("si no" en inglés).
¿Qué podemos poner como condición? Podemos comparar una variable con otra, un valor con una variable, un valor con otro valor (aunque no sea muy útil) o podemos usar funciones que nos devuelven un valor True o False (los veremos más adelante).
¿Cómo los comparamos? Con los operadores:

>	Mayor
<	Menor
>=	Mayor o Igual
<=	Menor o Igual
=	Igual
<>	Diferente

Ejemplos:

```
Dim numero As Integer
numero = TextBox1.Text
If numero > 2 Then
      MessageBox.Show(“El numero es mayor a
      2”)
Else
      MessageBox.Show(“El numero es menor que
2”)
End If
```

```
Dim numero As Integer
Numero = TextBox1.Text
If numero <> 2 Then
      MessageBox.Show(“El numero no es 2”)
Else
      MessageBox.Show(“El numero es 2,
increíble”)
End If
```

Operadores lógicos

¿Qué pasa si yo tengo una condición parecida a esta?
“Si mañana llueve Y tengo plata voy a ir al cine, si no voy a ir a jugar al golf”.
Ahora para ir al cine además de que llueva necesito tener plata, entonces si no tengo plata o no llueve voy a terminar yendo a jugar al golf, una fiesta.
Esto en programación se lo conoce como el operador lógico Y, en Visual Studio se expresa con And.
Ahora cambiemos la condición:
“Si mañana llueve o tengo plata voy al cine, si no voy a ir a jugar al golf”.
Ahora voy a ir al cine si llego a tener plata o si llueve, en caso de que ni tenga plata ni llueva voy a ir a jugar al golf.
Esto en programación se conoce como el operador lógico O, en Visual Studio lo expresamos con la palabra Or.
Y volvemos a cambiar la condición:
“Si no llueve, voy al cine, si no voy a ir a jugar al golf”.
En este caso pongo que NO llueva como condición. En programación en el negador. En Visual Studio se expresa como Not (condición).

Ejemplos:

```
Dim numero As Integer
numero = TextBox1.Text
If numero >=3 And numero <=7 Then
      MessageBox.Show("El numero esta entre 3 y
7")
End If

If numero <3 Or numero >7 Then
      MessageBox.Show("El numero no esta entre 3 y 7")
End If

If Not numero = 5 Then
      MessageBox.Show("El numero no es 5")
Else
      MessageBox.Show("El numero es 5, felicitaciones")
End If
```

La condición anidada

Muchas veces vamos a necesitar evaluar un valor dentro de un If, ya sea en su rama verdadera o en su rama falsa. A esto lo llamamos "anidar condiciones". O sea, que adentro de un If puede haber otro If.
Vamos a explicarlo con un ejemplo:
Ingresamos dos números a través de textBox, queremos ver a través de un If anidado si los dos son mayores a 5 o solo uno de ellos.

```
Dim n1, n2 As Integer
n1 = TextBox1.Text
n2= TextBox2.Text
If n1>5 Then
      If n2>5 Then
            msgBox("Los dos números son mayores a 5")
      Else
            msgBox("Solo el primero es mayor a 5")
      End If
Else
      If n2>5 Then
            msgBox("Solo el segundo es mayor a 5")
      Else
            msgBox("Los dos son menores a 5")
      End If
End If
```

Estructura de selección múltiple: Select Case

Es una manera diferente al If anidado de evaluar condiciones múltiples. La misma consiste en evaluar los casos en los que una sola variable tenga determinado valor. La instrucción Select Case permite utilizar tantas condiciones (o casos) como sea necesario, y conviene escribir el código para situaciones en las que hay muchas opciones. Por ejemplo, suponga que el programa utilizó una variable String para almacenar una opción de color y se necesitaba obtener el valor de color. El código para la instrucción Select Case podría ser similar al siguiente:

```
Select Case Color
  Case "rojo"
      MsgBox("Seleccionó rojo")
  Case "azul"
      MsgBox("seleccionó azul")
  Case "verde"
      MsgBox("seleccionó verde")
End Select
```

La sintaxis a emplear con Visual Basic es la que indicamos a continuación. Como se podrá comprobar, mediante esta instrucción establecemos una serie de casos que se podrían cumplir para el valor de una expresión o variable, y en función del valor de la variable se ejecutarán una serie de instrucciones asociadas u otra.

```
Select Case [expresión]
        Case [valor expresión 1]
           Instrucción 1
           Instrucción 2
        Case [valor expresión 2]
           Instrucción 3
           Instrucción 4
        .
        .
        .
        Case [valor expresión n]
           Instrucción k
        Case Else
           Instrucción m
End Select
```

El Case Else es la opción por default, o sea que si no cumple con ninguno de los casos indicados el programa irá directamente al Case Else y ejecutará lo que allí programamos. Esto se usa mayormente para mostrar algún mensaje de error si hubo un mal ingreso de datos. La expresión a evaluar puede ser un valor numérico o una cadena de texto. Sólo se puede evaluar una expresión y no múltiples expresiones. La evaluación de expresiones puede ser:

- De coincidencia: por ejemplo, Case 12 indicaría que si la expresión evaluada vale 12 se ejecutarán las instrucciones anexas.
- De intervalo: usando la palabra clave To. Por ejemplo, Case 12 To 14 indicaría que si la expresión evaluada tiene un valor comprendido entre 12 y 14 (incluidos los extremos de los intervalos), se ejecutarán las instrucciones anexas.
- De comparación: usando la palabra clave Is. Por ejemplo, Case Is <= 14 indicaría que si la expresión evaluada tiene un valor menor o igual a 14 se ejecutarán las instrucciones anexas.

¿Qué pasa cuando dos casos son válidos? Solamente se ejecutará el primero en el que aparezca una coincidencia, o sea el que aparezca primero se queda con el caso. Las instrucciones Select Case se pueden anidar. Cada instrucción Select Case debe tener su correspondiente terminación End Select. El Select Case guarda gran similitud con la estructura If, por lo que varias cosas de las que se realizan con un Select Case pueden ser realizadas con un If anidado.

Ejercicios

1) Realizar un programa que al ingresar un día de la semana (lunes, martes, miércoles, jueves, viernes, sábado, domingo) indique qué número de día es.
2) Realizar un programa que al ingresar un número del 1 al 7 indique a qué día de la semana corresponde,
3) Realizar una calculadora de la siguiente manera. Se ingresan dos números y el tipo de operación a través de TextBox. Utilizando una estructura del tipo Select Case determinar qué tipo de operación quiere hacer el usuario (+, -, *, /), realizarla e informar el resultado de la manera que le parezca más adecuada (a través de un MessageBox o en un Label).
4) El usuario ingresa dos números correspondientes al día y mes de un año. Realizar un programa que determine qué día del año es (por ejemplo, el 1° de enero es el día número 1, el 16 de marzo es el día 75 del año y que el 31 de diciembre es el día 365).

Paso a paso del ejercicio 1:

Lo primero que debemos realizar es el diseño del formulario. Como el programa es sencillo solamente necesitamos un TextBox para ingresar los días de la semana, un label que le indique al usuario que debe ingresar en el TextBox y obviamente un botón que ejecute la acción pedida. Para programar sobre el botón hacemos doble click en el mismo.

Codificación.

Lo primero que debemos realizar es declarar la variable que vamos a evaluar en el Select Case, como vamos a ingresar los días de la semana será del tipo String.

```
Dim dia As String
```

Ahora debemos guardar lo que está en el TextBox en esa variable.

```
dia=TextBox1.Text
```

Lo que realizamos acá es guardar en la variable n lo que el usuario ingresó en el TextBox1 (TextBox1.Text es el texto que se encuentra en el Cuadro al momento de ejecutar la función).
No está de más agregarle verificaciones al ingreso de datos.
Lo siguiente es evaluar las condiciones a través de un Select Case.

```
Select Case dia
  Case "lunes"
      MsgBox("El"+dia+" es el día 1 de la semana")

  Case "martes"
      MsgBox("El"+dia+" es el día 2 de la semana")
  Case "miercoles"
      MsgBox("El"+dia+" es el día 3 de la semana")
  Case "jueves"
      MsgBox("El"+dia+" es el día 4 de la semana")
  Case "viernes"
      MsgBox("El"+dia+" es el día 5 de la semana")
  Case "sabado"
      MsgBox("El"+dia+" es el día 6 de la semana")
  Case "domingo"
      MsgBox("El"+dia+" es el día 7 de la semana")
  Case Else
      MsgBox("Ingrese el dia correctamente")
      Me.Close()
End Select
```

En caso de que no haya ingresado ningún día válido (haya tenido algún error al ingresarlo o simplemente golpeó la cabeza contra el teclado) Lo que se ejecutará es el Case Else donde le indica que ingrese el día correctamente y luego cierra el programa. Recordemos que el Case Else es la opción por Default, o sea que se ejecutará en caso de no cumplirse ninguno de los casos anteriores. Los ejercicios siguientes consisten de estructuras similares, revisar bien qué tipo de datos me conviene usar, como haría el diseño del formulario, cómo haría el Select Case según los casos que tengo que evaluar y A trabajar.

Instituto Juan XXIII
Sistemas Tecnológicos 3° (Informática)

Estructuras de iteración: Ciclos

Como vimos anteriormente en programación existen las estructuras de decisión que nos permiten realizar acciones según se cumpla o no una condición. Ahora veremos que también existen estructuras que nos permiten realizar varias veces una misma acción (iterar). Según tenga una cantidad de iteraciones (ciclos) definida o no debemos usar determinada estructura.

Ciclo For

La estructura For...Next nos permite realizar una acción una cantidad determinada de veces, o sea, el programador determinará cuantas veces se va a iterar. Para esto vamos a necesitar una variable contadora (por lo general del tipo Integer y se llama en la mayoría de los lenguajes de programación i).

Sintaxis

```
Dim i As Integer  'declaro mi variable contadora
For i = valor_inicial To valor_final  'valor_inicial y valor_final son enteros
    [ sentencias ]
    [ código ]
    [ más código]
Next i  'llegamos al final del for acá
```

Ejemplo

En un formulario con un label vacío:

```
Dim i As Integer 'Lo primero que hacemos es declarar la contadora

Label1.Text=" " 'Nos aseguramos de que el label este vacio

For i = 0 To 5 'i va a ir de 0 a 5

    Label1.Text= Label1.Text&i.ToString()&VbCrlf 'al texto del label le
agregamos el valor de i en ese ciclo y un "enter" (VbCrlf)

Next i 'se aumenta i en 1
```

Copiar y pegar el código en un programa de Visual Basic y ver lo que se termina mostrando en el Label.
Recordemos que i aumenta en uno en cada ciclo arrancado de 0 y que va a seguir repitiendo ese ciclo hasta que i llegue a 5 en nuestro caso.
Dentro de un ciclo For pueden haber otros ciclos for como así cualquier estructura de decisión.
Importante: Si un For está dentro de otro For no puede usar la misma variable contadora que el primero, ya que provocará un mal funcionamiento del programa aunque compile igual.
En caso de que queramos salir del ciclo antes de terminarlo usaremos la instrucción `Exit For`.
Por ejemplo, en este caso:

```
Dim i As Integer 'Lo primero que hacemos es declarar la contadora

Label1.Text=" " 'Nos aseguramos de que el label este vacio

For i = 0 To 5 'i va a ir de 0 a 5

    Label1.Text= Label1.Text&i.ToString()&VbCrlf 'al texto del label le
agregamos el valor de i en ese ciclo y un "enter" (VbCrlf)

    If i=3 Then

      Exit For

    End If

Next i 'se aumenta i en 1
```

No va a mostrar nunca valores mayores a 3 ya que cuando i es 3 le digo que salga del ciclo.

Ejercicios

1) Se ingresa un número del 1 al 10 (verificar que esto sea así). Mostrar la tabla del número correspondiente. Por ejemplo, si el usuario ingresa el 2 que me muestre la tabla del 2.
2) El usuario ingresa un número del 1 al 100. Mostrar la cantidad de números enteros que hay entre ese número y 100.
3) Se le pide al usuario ingresar 5 números, determinar el promedio de los mismos, el número mayor, el número menor y la suma de los primeros tres números que ingresó.
4) Modificar el programa para que en lugar de 5 números enteros el usuario pueda decidir cuántos números quiere ingresar.
5) En un kiosco el dueño quiere determinar las siguientes cosas de los primeros 15 productos que vende por razones que no nos interesan:
    i. La sumatoria de los precios de esos 15 productos.
    ii. El precio mayor.
    iii. Si el total del precio de los productos es mayor que 500 que no me pida ingresar más datos y que cierre el programa.
    iv. Si el total final supera los 400 que me muestre un mensaje que diga "felicitaciones". Tampoco sabemos para qué necesita eso.

    El usuario ingresará los precios de los productos que venda de a uno hasta llegar a los 15 productos o que se corte el ciclo antes.

Ciclos condicionados: while y Do while

Hasta ahora vimos las estructuras de iteración que nos permitían repetir una misma acción una determinada cantidad de veces. Pero ¿Qué pasa cuando no sé cuántas veces voy a tener que iterar? En estos casos un ciclo For no nos sirve (O sí, pero es mucho más complejo) por lo que utilizamos los ciclos **while** y **do while**.

Ciclo While

Ejecuta una serie de instrucciones mientras una condición dada sea verdadera. Lo debemos usar cuando queramos repetir una serie de acciones una cantidad de veces indeterminada (en cada ejecución del programa varía según una condición, cuando esa condición no se cumple sale del ciclo).

Sintaxis

```
While condición
      [sentencias]
      [sentencias]
      [más sentencias]
End While
```

Si la condición es verdadera ejecutará el programa hasta encontrar la instrucción End While, Entonces vuelve al principio del While donde vuelve a evaluar la condición, de ser verdadera vuelve a ejecutar las sentencias hasta el final del ciclo y vuelve a evaluar la condición. Se repetirá esto hasta que la condición evaluada sea falsa, entonces seguirá con el resto del programa. Si la condición es falsa al entrar al ciclo por primera vez, no se ejecutará nunca lo que esté en el while. SI la variable que yo evalúo no cambia nunca dentro del ciclo, su resultado en la condición siempre será verdadero, por lo que se produce un ciclo infinito.

Ejemplo básico:

```
Dim counter As Integer = 0
While counter < 20
    counter += 1
    Label1.Text = Label1.Text & counter.ToString() &
Vbcrlf
End While
MsgBox("El ciclo se ejecutó " & CStr(counter) & " veces")
```

Ejemplo para calcular un promedio de los números que quiera ingresar el usuario:

```
Dim counter, numero, total As Integer
Dim promedio As Double

numero= InputBox("ingrese un Número, Ingrese 0 para finalizar")
' el primer ingreso siempre se hace afuera del ciclo

While numero <> 0
    counter += 1
    total += numero
    numero= InputBox("ingrese un Número, Ingrese 0 para finalizar")
' debo realizar un ingreso de datos en el ciclo para que se siga
ejecutando el programa
End While

promedio = total/counter
MsgBox("El promedio es" & CStr(promedio))
```

En este caso yo voy a ingresar datos hasta que no quiera más (finalizo el ingreso con un 0). Dentro del ciclo voy acumulando los números y contando cuántos ingresé. Al finalizar el ingreso calculo el promedio y lo muestro.

Ejercicios

1) Un negocio quiere indicar el total de ventas de un día. El usuario carga los importes de las mismas hasta que ingresa un 0 para indicar el final. Mostrar el total y si no se hicieron ventas en todo el día mostrarlo a través de un mensaje.
2) Se ingresa el apellido de una determinada cantidad de alumnos (para finalizar se ingresa FIN) y la nota final del año. Si esta es mayor o igual a 7 mostrar un mensaje que diga “El alumno de nombre (acá va el nombre del alumno) aprobó”, si es mayor o igual a 4 mostrar un mensaje que diga “El alumno de nombre (acá va el nombre del alumno) va a diciembre” y si es menor a 4 mostrar “El alumno de nombre (acá va el nombre del alumno) va a febrero”. Usar condiciones anidadas.
3) Se ingresa el apellido de una determinada cantidad de alumnos (para finalizar se ingresa FIN) y las notas de los tres trimestres. Calcule la nota final del año haciendo un promedio de las tres notas. Si esta es mayor o igual a 7 mostrar un mensaje que diga “El alumno de nombre (acá va el nombre del alumno) aprobó”, si es mayor o igual a 4 mostrar un mensaje que diga “El alumno de nombre (acá va el nombre del alumno) va a diciembre” y si es menor a 4 mostrar “El alumno de nombre (acá va el nombre del alumno) va a febrero”. Usar condiciones anidadas.
4) En un negocio de videojuegos hacen tres descuentos según el importe de la compra:
    - Si el importe es mayor a $500, el descuento es del 10%
    - Si el importe es mayor $1000, el descuento es del 15%
    - Si el importe es mayor a $2000, el descuento es del 20%

    Nos piden un programa que les permita ingresar las ventas que hacen en el día y que muestre en un mensaje el importe con descuento y que al finalizar el día (cuando se ingresa el importe 0 o un número negativo) muestre el total vendido.
5) Se ingresa una cantidad indeterminada de números. Determinar el promedio de los números pares.
6) Se ingresa una cantidad de números indeterminada (el ingreso finaliza con el valor -32). Mostrar el mayor.

Ciclo Do While

El ciclo Do While funciona de una manera similar al ciclo While con una sola diferencia: El ciclo while evalúa y después ejecuta, en cambio el do while ejecuta y después evalúa. ¿Qué cambia? En un While si la condición resulta falsa en la primera evaluación nunca va a entrar al ciclo, en cambio en un Do While como evalúa después de ejecutar sé que por lo menos una vez voy a realizar las instrucciones que estén en el ciclo.

¿Y para qué me sirve? Es muy útil cuando por ejemplo necesito ingresar datos como lo venimos haciendo porque usando un Do While le puedo pedir al usuario que vuelva a ingresar los datos hasta que lo haga correctamente.

Sintaxis

```
Do
        [sentencias]
        [sentencias]
        [más sentencias]
Loop Until condición
```

Las sentencias se ejecutan hasta que la condición del final sea **Verdadera.**

Ejemplo: Calculando un promedio, pero validando los datos que ingreso.

```
Dim counter, numero, total As Integer
Dim promedio As Double
Do
    numero= InputBox("ingrese un Número, Ingrese 0 para finalizar")
Loop Until numero>=0
' el primer ingreso siempre se hace afuera del ciclo
' me va a pedir que ingrese el numero hasta que ingrese un número mayor o
igual a 0.
While numero <> 0
    counter += 1
    total += numero
    Do
        numero= InputBox("ingrese un Número, Ingrese 0 para finalizar")
    Loop Until numero>=0
'debo realizar un ingreso de datos en el ciclo para que se siga ejecutando
el programa
End While

promedio = total/counter
MsgBox("El promedio es" & CStr(promedio))
```

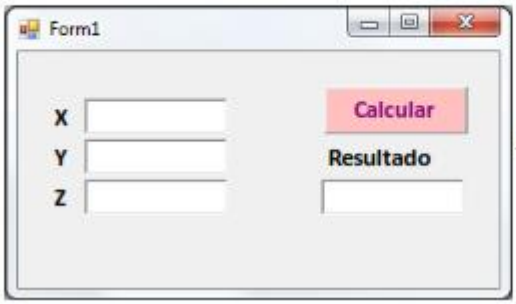
Ejercicios

1) Agregarles a los ejercicios realizados anteriormente la validación con un Do While de los datos ingresados.
2) Un negocio de venta de granos desea controlar las ventas realizadas. De cada una ingresa el importe total y un código que indica la forma de pago. El código puede ser:
    a. C : cooperativa , 30% de descuento
    b. E : contado, 10% de descuento
    c. T : con tarjeta, 12% de recargo

Se debe ingresar una F para finalizar el día de venta y arrojar los totales cendidos en tarjeta, al contado y cooperativa.
Validar los ingresos de todos los datos (que los importes no sean negativos o 0 y que las letras no sean diferentes C, E y T) a través de un ciclo Do While.

EJERCICIOS DE REFORZAMIENTO

1.



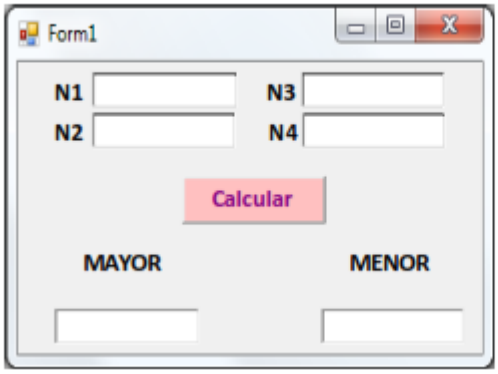
$$R = \frac{\sqrt{3x^4 + 2xy^2z} + 5x^2y}{3 + x^2y^2z^2}$$

RESOLUCIÓN:

EN MODO DE DISEÑO:

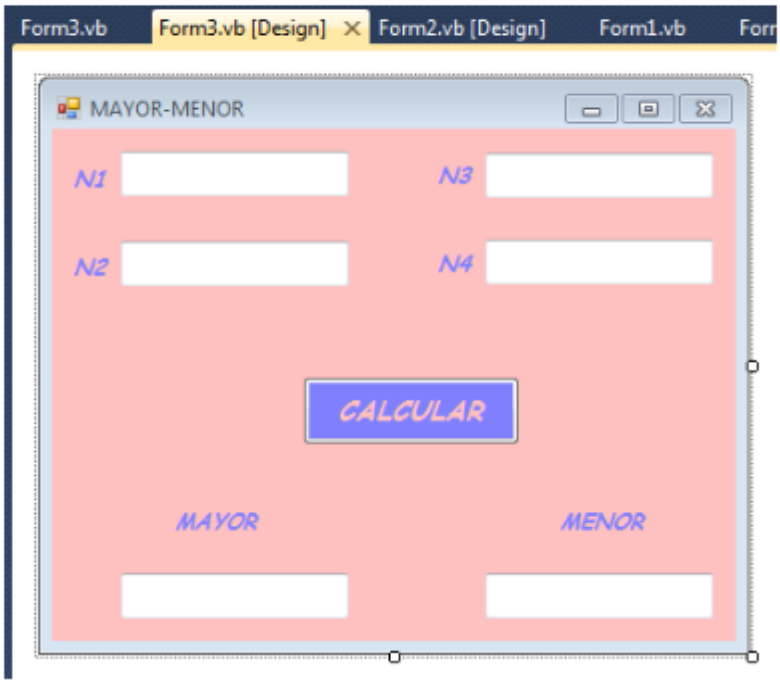
EL CÓDIGO:

```
Public Class Form1
    Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles Button1.Click
        Dim X, Y, Z, R As Double
        X = TextBox1.Text
        Y = TextBox2.Text
        Z = TextBox3.Text
        R = (Math.Sqrt(3 * Math.Pow(X, 4) + 2 * X * Math.Pow(Y, 2) * Z) + 5 * Math.Pow(X, 2)
          * Y) / (3 + Math.Pow(X, 2) * Math.Pow(Y, 2) * Math.Pow(Z, 2))
        TextBox4.Text = R
    End Sub
End Class
```



RESOLUCIÓN:

EN MODO DE DISEÑO:



EL CODIGO:

Form3.vb | Form3.vb [Design] | Form2.vb [Design] | Form1.vb | Form1.vb [Design]

Form3 | (Declarations)

```
Public Class Form3

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim MAY, MEN, N1, N2, N3, N4 As Double
        N1 = TextBox1.Text
        N2 = TextBox2.Text
        N3 = TextBox3.Text
        N4 = TextBox4.Text
        MAY = Math.Max(Math.Max(Math.Max(N1, N2), N3), N4)
        MEN = Math.Min(Math.Min(Math.Min(N1, N2), N3), N4)
        TextBox5.Text = MAY
        TextBox6.Text = MEN
    End Sub
End Class
```

100 %

PANTALLAS (FUNCIONAMIENTO):

JUEGO DE TRES EN RAYA

El Tres en Raya es un juego muy popular.

Objetivo: El objetivo de este juego es lograr formar una línea recta con "X" o "0" en forma horizontal, vertical o diagonal.

Reglas:

- Se puede jugar solamente entre dos personas "X" y "0".
- Se seleccionará aquel que empiece primero "x" o "0".
- Después de hacer clic en las celdas en los diferentes turnos de cada jugador. Ganará aquel que haya cumplido con el objetivo del juego, iniciando así otro juego.

Juego en Visual Basic:

Modo Diseño:

Modo de programación o ejecución

```
Public Class Form1
    Dim nj As Integer
    Dim c As Integer
    Sub REINICIAR()
        Dim OBJ As Control
        For Each OBJ In Me.Controls
            If TypeOf OBJ Is PictureBox Then
                Dim IMG As PictureBox
                OBJ.Tag = ""
                IMG = OBJ
                IMG.Image = Nothing
            End If
            'MessageBox.Show(OBJ.Name)
            nj = 0
            c = 0
            RadioButton1.Enabled = True
            RadioButton1.Checked = True
            RadioButton2.Checked = False
            RadioButton2.Enabled = True
        Next
```

```
    Private Sub C11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles C11.Click, C12.Click, C13.Click, C21.Click,
C22.Click, C23.Click, C31.Click, C32.Click, C33.Click
        If nj <= 1 Then
            RadioButton1.Enabled = False
            RadioButton2.Enabled = False
        End If
        If sender.tag = "" Then
            If nj Mod 2 = 0 Then
                sender.image = Image.FromFile("I:\x.png")
                sender.tag = "x"
            Else
                sender.image = Image.FromFile("I:\o.png")
                sender.tag = "o"
            End If
            nj += 1
            c += 1
        Else
            MessageBox.Show("Celda Ocupada por: " + sender.tag)
        End If
        If C11.Tag = sender.tag And C12.Tag = sender.tag And C13.Tag = sender.tag Then
            MsgBox("BIEN HECHO, GANO " + sender.tag, MsgBoxStyle.Information, "ACIERTO")
            REINICIAR()
        ElseIf C11.Tag = sender.tag And C21.Tag = sender.tag And C31.Tag = sender.tag Then
            MsgBox("BIEN HECHO, GANO " + sender.tag, MsgBoxStyle.Information, "ACIERTO")
            REINICIAR()
        ElseIf C13.Tag = sender.tag And C23.Tag = sender.tag And C33.Tag = sender.tag Then
            MsgBox("BIEN HECHO, GANO " + sender.tag, MsgBoxStyle.Information, "ACIERTO")
            REINICIAR()
        ElseIf C31.Tag = sender.tag And C32.Tag = sender.tag And C33.Tag = sender.tag Then
            MsgBox("BIEN HECHO, GANO " + sender.tag, MsgBoxStyle.Information, "ACIERTO")
            REINICIAR()
        ElseIf C11.Tag = sender.tag And C22.Tag = sender.tag And C33.Tag = sender.tag Then
            MsgBox("BIEN HECHO, GANO " + sender.tag, MsgBoxStyle.Information, "ACIERTO")
            REINICIAR()
        ElseIf C13.Tag = sender.tag And C22.Tag = sender.tag And C31.Tag = sender.tag Then
            MsgBox("BIEN HECHO, GANO " + sender.tag, MsgBoxStyle.Information, "ACIERTO")
            REINICIAR()
        ElseIf C21.Tag = sender.tag And C22.Tag = sender.tag And C23.Tag = sender.tag Then
            MsgBox("BIEN HECHO, GANO " + sender.tag, MsgBoxStyle.Information, "ACIERTO")
            REINICIAR()
        ElseIf C12.Tag = sender.tag And C22.Tag = sender.tag And C32.Tag = sender.tag Then
            MsgBox("BIEN HECHO, GANO " + sender.tag, MsgBoxStyle.Information, "ACIERTO")
            REINICIAR()
        ElseIf c = 9 Then
            MsgBox("EL JUEGO TERMINO, NO HAY GANADOR", MsgBoxStyle.Critical, "DESACIERTO")
            REINICIAR()
        End If
    End Sub
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        nj = 1
        c = 0
    End Sub
    Private Sub RadioButton1_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
RadioButton1.CheckedChanged
        nj = 0
    End Sub
    Private Sub RadioButton2_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
RadioButton2.CheckedChanged
        nj = 1
    End Sub
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        REINICIAR()
    End Sub
End Class
```

¿Cómo funciona?

Como determinado esta check el aspa o cruz, se empieza a jugar:

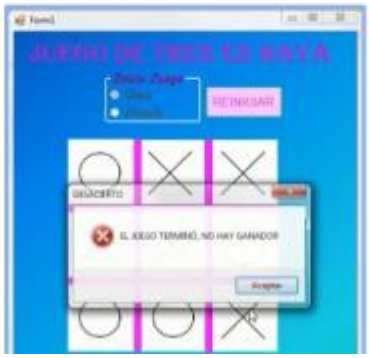
Caso cuando gana "X":

Se puede elegir que empiece el círculo:

Cuando se arrepienten del juego se puede reiniciar:

Caso cuando gana la "o":

Caso que no hay ganador:



JUEGO DE BUSCAMINAS

1. El Buscaminas es un juego aparentemente sencillo de memoria y lógica, además de ser uno de los juegos más populares.

Objetivo: Encontrar los recuadros vacíos y evitar las minas.

El tablero: Es una matriz de 6 x 6 (seis filas y seis columnas) y aleatoriamente están 6 minas en diferentes casilleros.

Cómo jugar

Las reglas del Buscaminas son simples:

1. Si se descubre una mina termina el juego. Se hace clic en el botón **Reiniciar** para empezar otro.
2. Si se desea reiniciar antes se hace clic en el botón Reiniciar.
3. Si se descubre una carta de un as corazones sigue el juego.

En Visual Basic

Modo de diseño

Código:

```
PublicClassBuscaminas
DimV(36) AsInteger
Dim I, N AsInteger
SubREINICIAR()
Dim OBJ AsControl

ForEach OBJ InMe.Controls
IfTypeOf OBJ IsPictureBoxThen
Dim IMG AsPictureBox
OBJ.Tag = ""
            IMG = OBJ
IMG.Image = Nothing
IMG.Enabled = True
EndIf
        I = 0
        N = 0
Next
EndSub

Subaleatorio()
Randomize()
Dim VAR, C AsInteger
      VAR = 1
While VAR > 0
         VAR = 0
         I = Math.Truncate(Rnd() * 36 + 1) 'genera un numero aleatorio en un rango de 1-36
For C = 0 To N
If V(C) = I Then
VAR += 1
EndIf
Next
EndWhile
      V(N) = I
      N += 1
EndSub

PrivateSub C11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles C11.Click, C12.Click, C13.Click, C14.Click, C15.Click, C16.Click, C21.Click,
C22.Click, C23.Click, C24.Click, C25.Click, C26.Click, C31.Click, C32.Click, C33.Click,
C34.Click, C35.Click, C36.Click, C41.Click, C42.Click, C43.Click, C44.Click, C45.Click,
C46.Click, C51.Click, C52.Click, C53.Click, C54.Click, C55.Click, C56.Click, C61.Click,
C62.Click, C63.Click, C64.Click, C65.Click, C66.Click
aleatorio()
sender.tag = CStr(i)
If I = 1 Or I = 2 Or I = 3 Or I = 4 Or I = 5 Or I = 6 Then
sender.image = Image.FromFile("f:\cartas\0.png")
MsgBox("MINA, PERDISTE", MsgBoxStyle.Critical, "DESACIERTO")
DimobjAsControl
ForEachobjInMe.Controls
IfTypeOfobjIsPictureBoxAndobj.Tag = ""Then
```

Cuando se arrepienten del juego se puede reiniciar:

Caso cuando gana la "o":

Caso que no hay ganador:

INCLUIR JUEGOS

JUEGO DE BUSCAMINAS

1. El Buscaminas es un juego aparentemente sencillo de memoria y lógica, además de ser uno de los juegos más populares.

Objetivo: Encontrar los recuadros vacíos y evitar las minas.

El tablero: Es una matriz de 6 x 6 (seis filas y seis columnas) y aleatoriamente están 6 minas en diferentes casilleros.

Cómo jugar

Las reglas del Buscaminas son simples:

1. Si se descubre una mina termina el juego. Se hace clic en el botón **Reiniciar** para empezar otro.
2. Si se desea reiniciar antes se hace clic en el botón Reiniciar.
3. Si se descubre una carta de un as corazones sigue el juego.

En Visual Basic

Modo de diseño

```
PublicClassBuscaminas
DimV(36) AsInteger
Dim I, N AsInteger
SubREINICIAR()
Dim OBJ AsControl

ForEach OBJ InMe.Controls
IfTypeOf OBJ IsPictureBoxThen
Dim IMG AsPictureBox
OBJ.Tag = ""
            IMG = OBJ
IMG.Image = Nothing
IMG.Enabled = True
EndIf
         I = 0
         N = 0
Next
EndSub

Subaleatorio()
Randomize()
Dim VAR, C AsInteger
       VAR = 1
While VAR > 0
          VAR = 0
          I = Math.Truncate(Rnd() * 36 + 1) 'genera un numero aleatorio en un rango de 1-36
For C = 0 To N
If V(C) = I Then
VAR += 1
EndIf
Next
EndWhile
       V(N) = I
       N += 1
EndSub

PrivateSub C11_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles C11.Click, C12.Click, C13.Click, C14.Click, C15.Click, C16.Click, C21.Click,
C22.Click, C23.Click, C24.Click, C25.Click, C26.Click, C31.Click, C32.Click, C33.Click,
C34.Click, C35.Click, C36.Click, C41.Click, C42.Click, C43.Click, C44.Click, C45.Click,
C46.Click, C51.Click, C52.Click, C53.Click, C54.Click, C55.Click, C56.Click, C61.Click,
C62.Click, C63.Click, C64.Click, C65.Click, C66.Click
aleatorio()
sender.tag = CStr(i)
If I = 1 Or I = 2 Or I = 3 Or I = 4 Or I = 5 Or I = 6 Then
sender.image = Image.FromFile("f:\cartas\0.png")
MsgBox("MINA, PERDISTE", MsgBoxStyle.Critical, "DESACIERTO")
DimobjAsControl
ForEachobjInMe.Controls
IfTypeOfobjIsPictureBoxAndobj.Tag = ""Then
```

```
aleatorio()
If I = 1 Or I = 2 Or I = 3 Or I = 4 Or I = 5 Or I = 6 Then
DimimagenAsPictureBox
imagen = obj
imagen.Image = Image.FromFile("f:\cartas\0.png")
EndIf
obj.Tag = CStr(I)
EndIf
Next

Else
sender.image = Image.FromFile("f:\cartas\1.png")
If N = 30 Then
MsgBox("BIEN HECHO, GANASTE!!!", MsgBoxStyle.Information, "ACIERTO")
REINICIAR()
EndIf
EndIf
EndSub

PrivateSub Button1_Click(ByVal sender AsSystem.Object, ByVal e AsSystem.EventArgs)
Handles Button1.Click
REINICIAR()
EndSub
EndClass
```

JUEGO DE MEMORIA

1. El Juego de Memoria, denominado también pescador es un juego, que como el nombre indica, se necesita tener buena memoria para poder ganarlo.

Objetivo: Encontrar la pareja de una de las imágenes dentro de un grupo de imágenes.

El tablero: Es una matriz de 4 x 4 (cuatro filas y cuatro columnas) y aleatoriamente están dispuestas 8 imágenes con sus respectivas parejas.

Al hablar de parejas se hace mención a la misma imagen.

Cómo jugar

Las reglas del Juego de Memoria son las siguientes:

a) Si se descubre la pareja de la imagen, previamente seleccionada; ambas se ocultan. Si se desea reiniciar antes se hace clic en el botón Reiniciar.
b) Si no se descubre la imagen, ambas se voltean.

En Visual Basic

Modo de diseño

```
Public Class Memoria
    Dim ig1 As New PictureBox
    Dim ig2 As New PictureBox
    Dim imag(16) As PictureBox
    Dim vector1(16) As Integer
    Dim j, cClick As Integer

    Private Sub Memoria_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Randomize()
        cClick = 0
        j = -1
    End Sub

    Function buscaRepet(ByVal ParamArray vector2() As Integer) As Integer 'ASIGNA UN
NUMERO ALEATORIO (1-8) A CADA PICTUREBOX
        Dim cont, estado, numero As Integer                                  'QUE SÓLO
SE REPITA DOS VECES
        estado = 0
        j = j + 1
        While estado < 1
            numero = Int(Rnd() * 8) + 1
            cont = 0
            For k As Integer = 0 To j
                If numero = vector2(k) Then
                    cont = cont + 1
                End If
            Next
            If cont <= 1 Then
                vector2(j) = numero
                estado = 1
            End If
        End While
        Return numero
    End Function

    Sub AsinaImagen(ByVal imagen As PictureBox)
        If imagen.Tag Is Nothing Then
            imagen.Tag = buscaRepet(vector1)
            imagen.Image = Image.FromFile(Application.StartupPath + "\IMAGS\" +
CStr(imagen.Tag) + ".jpg") ' APPLICATION.STARTUPPATH PARA TENER LAS IMÁGENES EN EL
MISMO PROYECTO
            imag(j) = imagen
        Else
            If imagen.BackColor <> Color.Black Then
                imagen.Image = Image.FromFile(Application.StartupPath + "\IMAGS\" +
CStr(imagen.Tag) + ".jpg")
            End If
        End If
    End Sub

    Private Sub P34_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles P11.Click, P12.Click, P13.Click, P14.Click, P21.Click, P22.Click, P23.Click,
```

```
P24.Click, P31.Click, P32.Click, P33.Click, P34.Click, P41.Click, P42.Click,
P43.Click, P44.Click
        Dim imgValida As New PictureBox
        imgValida = sender
        If imgValida.Image Is Nothing And imgValida.BackColor = Color.Pink Then
            cClick = cClick + 1
        End If
        If (cClick = 1) Then
            ig1 = sender
            AsinaImagen(ig1)
        ElseIf (cClick = 2) Then
            ig2 = sender
            AsinaImagen(ig2)
        Else
            If ig1.Tag = ig2.Tag Then
                ig1.BackColor = Color.Black
                ig2.BackColor = Color.Black
                ig1.Image = Nothing
                ig2.Image = Nothing
                ig1 = sender
                AsinaImagen(ig1)
            ElseIf ig2.Image Is Nothing = False Then
                ig1.Image = Nothing
                ig2.Image = Nothing
                ig1 = sender
                AsinaImagen(ig1)
            End If
            cClick = 1
        End If
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        For Z As Integer = 0 To j
            imag(Z).BackColor = Color.Pink
            imag(Z).Image = Nothing
            imag(Z).Tag = Nothing
            vector1(Z) = 0
        Next
        j = -1
        cClick = 0
    End Sub

End Class
```

OBJETO TIMER

OBJETO NO VISIBLE SE ACTIVA EN FUNCIÓN DEL TIEMPO

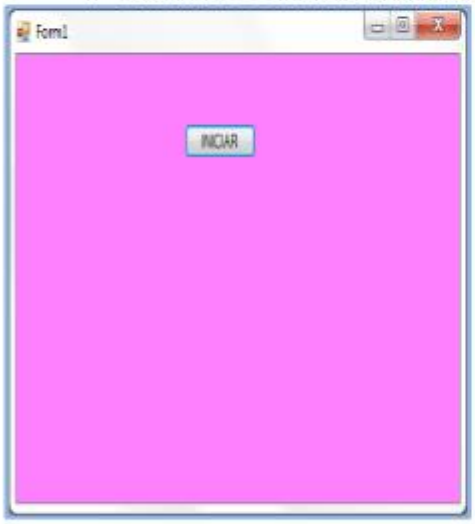
METODOS:

TIMER1.START()

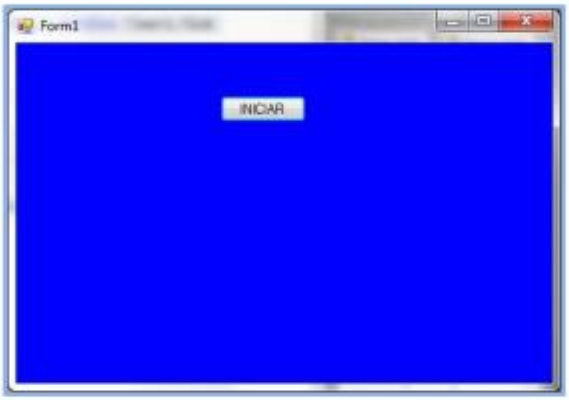
TIMER1.STOP() DETENER EL CRONOMETRO

INTERVAL=INTERVALO DE TIEMPO ENTRE CADA EVENTO

1. CAMBIAR DE COLOR AL FORMULARIO.

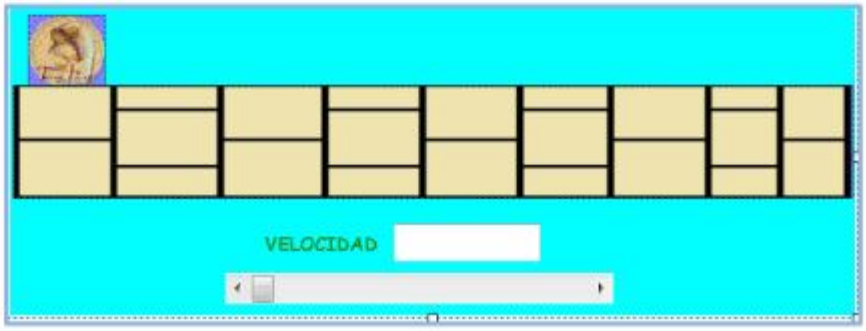



```
Public Class Form1
    Private Sub Timer1_Tick(sender As
System.Object, e As System.EventArgs)
Handles Timer1.Tick

        If Me.BackColor = Color.Blue
Then
            Me.BackColor = Color.Red
        Else
            Me.BackColor = Color.Blue
        End If
    End Sub

    Private Sub Button1_Click(sender As
System.Object, e As System.EventArgs)
Handles Button1.Click
        Me.BackColor = Color.Blue
        If Button1.Text = "INICIAR"
Then
            Timer1.Start()
            Button1.Text = "DETENER"
        Else
            Timer1.Stop()
            Button1.Text = "INICIAR"
        End If
    End Sub
End Class
```

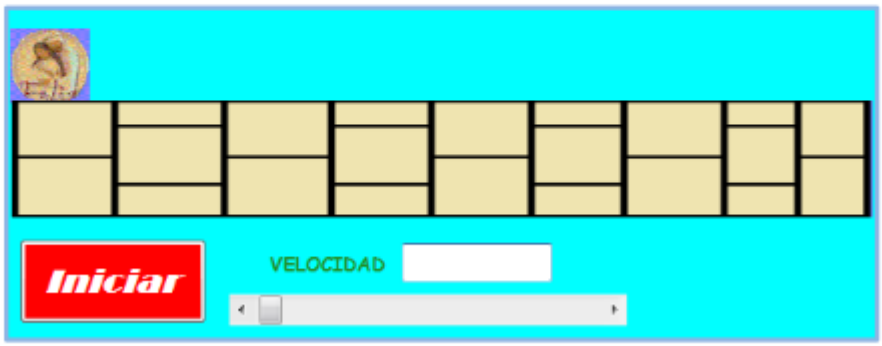


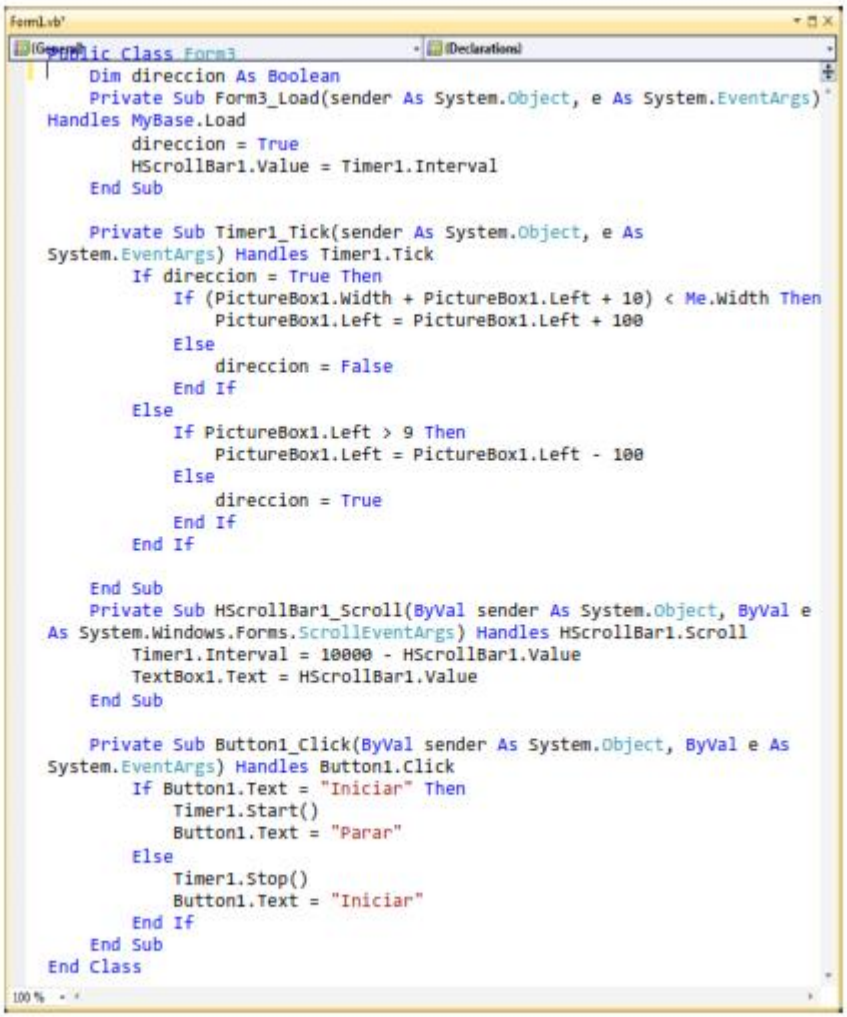
LEFT,HEIGHT,WIDTH,TOP

HSCROLLBAR

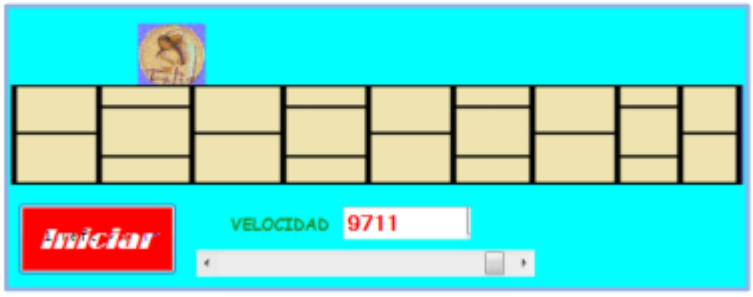


Programando el mismo ejemplo con radioButton:





```
Form1.vb*
(General)                                   (Declarations)
Public Class Form3
    Dim direccion As Boolean
    Private Sub Form3_Load(sender As System.Object, e As System.EventArgs)
Handles MyBase.Load
        direccion = True
        HScrollBar1.Value = Timer1.Interval
    End Sub

    Private Sub Timer1_Tick(sender As System.Object, e As
System.EventArgs) Handles Timer1.Tick
        If direccion = True Then
            If (PictureBox1.Width + PictureBox1.Left + 10) < Me.Width Then
                PictureBox1.Left = PictureBox1.Left + 100
            Else
                direccion = False
            End If
        Else
            If PictureBox1.Left > 9 Then
                PictureBox1.Left = PictureBox1.Left - 100
            Else
                direccion = True
            End If
        End If

    End Sub
    Private Sub HScrollBar1_Scroll(ByVal sender As System.Object, ByVal e
As System.Windows.Forms.ScrollEventArgs) Handles HScrollBar1.Scroll
        Timer1.Interval = 10000 - HScrollBar1.Value
        TextBox1.Text = HScrollBar1.Value
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        If Button1.Text = "Iniciar" Then
            Timer1.Start()
            Button1.Text = "Parar"
        Else
            Timer1.Stop()
            Button1.Text = "Iniciar"
        End If
    End Sub
End Class
100 %
```



MOVIENDO UN OBJETO CON KEYPRESS

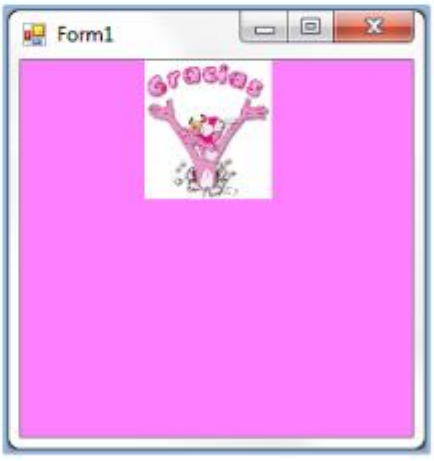
MOVIENDO OBJETOS CON KEY DOW
KEY PREVIEW=TRUE
Mover un objeto con las letras:
A=izquieda
S=derecha
W=arriba
Z=abajo

Se puede reemplazar el código:

```
'If Char.ToUpper(e.KeyChar) = "A" Then
'PictureBox1.Left -= 10
'ElseIf Char.ToUpper(e.KeyChar) = "S" Then
'PictureBox1.Left += 10
'ElseIf Char.ToUpper(e.KeyChar) = "W" Then
'PictureBox1.Top -= 10
'ElseIf Char.ToUpper(e.KeyChar) = "Z" Then
'PictureBox1.Top += 10
'End If
```

Por un iif anidado:

```
Public Class Form1

    Private Sub Form1_KeyPress(sender As Object, e As
System.Windows.Forms.KeyPressEventArgs) Handles Me.KeyPress
        PictureBox1.Left -= IIf(Char.ToUpper(e.KeyChar) = "A", 1, 0)
        PictureBox1.Left += IIf(Char.ToUpper(e.KeyChar) = "D", 1, 0)
        PictureBox1.Top -= IIf(Char.ToUpper(e.KeyChar) = "W", 1, 0)
        PictureBox1.Top += IIf(Char.ToUpper(e.KeyChar) = "Z", 1, 0)
    End Sub
End Class
```

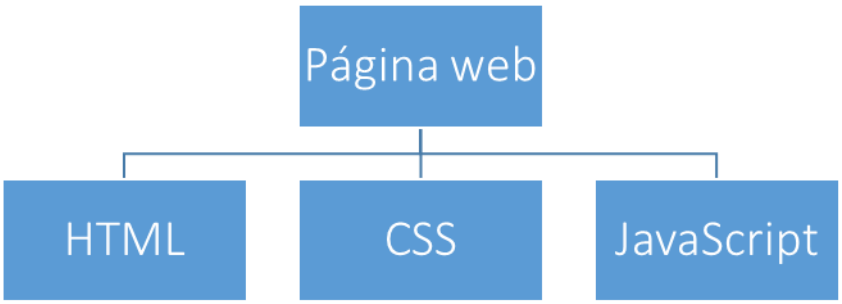


Páginas Web

¿Qué es la web? Son un montón de máquinas conectadas entre sí, las cuales acceden a páginas web. En un principio solamente la usaban científicos para compartir conocimientos. Hoy en día existen más de mil millones de páginas de todo tipo, desde hacer compras, aprender, comunicarse o compartir fotos de tu gatito. A las páginas web accedemos a través de navegadores (Firefox, Chrome, Opera, etcétera) en gran cantidad de dispositivos.
¿Cómo programamos una página?
Lo realizamos a través de tres lenguajes: HTML para marcar el contenido de nuestro sitio, CSS para darle estilo y JavaScript para poder hacerlo interactivo.

HTML

HyperText Markup Language (lenguaje de marcado de hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. El HTML se escribe en forma de etiquetas, rodeadas por corchetes angulares (<,>). Estas etiquetas nos permiten insertar textos con énfasis, imágenes, hipervínculos, tablas y demás.



JavaScript

Es un lenguaje interpretado orientado a objetos. Se usa en en páginas web para hacer que la página sea interactiva (que no solo sean objetos puestos), si no que “funcione”. Nos permite agregar funcionalidades como abrir nuevas ventanas o enviar mensajes al usuario.

Nuestra primera página

Para programar en HTML no necesitamos más que el bloc de notas. Ahí vamos a escribir lo siguiente

```
<HTML>
   <head>
      <title>Mi primera página</title>
   </head>
   <body>
      <h1>Hola a todos, hice mi primera página</h1>
   </body>
</html>
```

El próximo paso es guardarla como página web. Vamos a archivo, guardar como y en el nombre de archivo ponemos miPagina.html. Cuando se abra el archivo se verá en una nueva ventana del navegador.

Etiquetas

Como vimos antes HTML se maneja por etiquetas que nos dejan insertar varios elementos, todos van entre corchetes angulares. Muchos requieren una etiqueta de inicio y una de fin (<h1></h1>), otras inician y terminan en la misma etiqueta (<img />). La etiqueta que siempre debe estar es la primera que vemos en nuestra página de ejemplo y que indica que inicia y termina la página: <html> </html>. Dentro de esto es necesario aclarar el encabezado. <head> </head> donde con la etiqueta <title>titulo</title> ponemos el encabezado de la página que se ve en la pestaña. El cuerpo de la página donde va el contenido de la misma va entre las etiquetas <body></body>, es acá donde vamos a poner todos los elementos que veremos. ¿Cuáles son? Vamos a ver

Encabezados

Probemos lo siguiente en una página:

```
<html>
   <head>
       <title>Gritando</title>
   </head>
   <body>
       <h1>GOOOOOOOL</h1><br>
       <h2>GOOOOOOL</h2><br>
       <h3>GOOOOOL</h3><br>
       <h4>Offside</h4><br>
   </body>
</html>
```

¿Qué cambia? Con estas etiquetas podemos insertar títulos de diferentes jerarquías. Si quiero insertar texto luego de cada encabezado lo puedo hacer entre las etiquetas .

DIV

Nos permite dividir el contenido de una página web, dándole las características que queramos como por ejemplo el color de fondo y demás sin cambiar el resto de la página.

Listas

**Listas no ordenadas: **

Las listas no ordenadas van dentro de la etiqueta <ul> HTML y de su cierre . Cada punto que queramos añadir a la lista, lo haremos dentro de la etiqueta <li> y su cierre.

**Listas ordenadas: **

Las listas ordenadas van enmarcadas dentro de las etiquetas <ol> . Cada punto de la lista se escribe con la misma etiqueta que en las no numeradas: <li>. Pero al ser listas ordenadas los símbolos serán números y éstos se irán generando automáticamente por orden, conforme escribamos nuevos puntos.

Imágenes

El tag básico para colocar una imagen es "img ". Este tag, a diferencia de la gran mayoría de los tags de html, no necesita un cierre. Va acompañado de diferentes atributos que te vamos a explicar a continuación.
El atributo "src" es imprescindible para poder colocar una imagen. Este atributo es el que indica dónde se encuentra alojada la imagen que queremos mostrar. Se escribe así: <img src="x">, siendo "x" la dirección o la url dónde se encuentra situada la foto.
La foto podemos alinearla en la página como queramos mediante "align ", utilizando los atributos "left" para alinearla a la izquierda, "right" para alinearla a la derecha, "top" para alinearla arriba, "bottom" para alinearla abajo y "middle" para alinearla al centro.
También podemos poner una descripción de la imagen dentro de la misma para que la gente puede leerla al mantener el ratón encima de ella. Esta descripción podemos escribirla mediante el atributo "alt" y lo escribiríamos de la siguiente manera: alt= "x", siendo "x" la descripción que le gente leerá al pasar el ratón por encima.

Marquee

La etiqueta marquee HTML nos sirve para dar un efecto diferente al texto de nuestra página Web. Gracias a ella podemos conseguir texto en movimiento. Para aplicar este efecto a los textos, éstos deben estar dentro de la etiqueta marquee, entre su inicio "<marquee>" y su cierre "</marquee>".

El movimiento, la dirección de desplazamiento, la velocidad del mismo, todo es configurable gracias a los siguientes atributos:
align: Este atributo nos indicará si el texto dentro de la etiqueta se alineará en la zona alta del marquee ("top"), en el medio ("middle") o en la parte baja ("bottom").
bgcolor: Con este atributo definiremos el color de fondo que le queremos dar a la marquesina donde está el texto en movimiento.
height y width: El primero marca la altura que tendrá la marquesina y el segundo la anchura de la misma.
scrollamount: Este atributo define la cantidad de píxeles que queremos que se desplace el texto en cada movimiento. Por ejemplo: <marquee scrollamount="3″> </marquee> significará que el texto que vaya dentro de esa etiqueta se moverá a razón de tres píxeles por movimiento.
scrolldelay: Éste nos define la velocidad del texto que está dentro de la marquesina. A menor numeración, mayor velocidad. Es decir, un texto irá más rápido si el scrolldelay es 5, que si el scrolldelay es 20.
direction: Sirve para definir la dirección del movimiento: "left" para la izquierda, "right" para la derecha, "top" para arriba y "down" para abajo.
behavior: Gracias a este atributo podemos dar nuevos efectos a la marquesina. Si no especificamos este atributo, el texto se moverá de forma circular en el sentido que le hayamos marcado. Con behavior=" scroll" conseguiremos el mismo efecto: el texto se moverá circularmente. Con behavior=" slide" haremos que el texto se detenga al llegar al final de la marquesina. Y con el behavior=" alternate" el texto irá y volverá de un lado a otro de la marquesina.
Un ejemplo:

```
<marquee bgcolor="#FFFFFF" width="50%" scrolldelay="100”>
     Hola me estoy moviendo.
</marquee>
```

Hipervínculo

Es dentro de esta etiqueta y de su cierre () dónde encontraremos el enlace. Todo lo que esté dentro de ella, ya sea texto o una imagen, estará considerado como un enlace y el navegador lo interpretará así.

Atributo href
Para que un enlace esté activo debemos indicar dentro de él el destino del mismo. Para establecer este destino le colocamos a la etiqueta <a> este atributo. La etiqueta completa nos quedaría así:

```
<a href= “www.google.com.ar”>Click acá! </a>
```

Tablas

Una tabla en html viene marcada por las etiquetas <table> </table>. Entre esas dos etiquetas definiremos la tabla, las celdas que queremos, las columnas y las características de cada uno de estos parámetros. Pero vamos a empezar explicándote la etiqueta <table>.
Como ya ocurre con la etiqueta body, a una tabla también lo podemos definir el fondo de la misma. Esto lo podemos conseguir con el parámetro “bgcolor”, que nos pondrá un color de fondo, o “background” para poner una imagen de fondo. Recuerda que, si la imagen es más pequeña que la tabla, ésta se repetirá tanto a lo ancho como a lo largo.
Otro aspecto que podemos definir de la tabla es el borde. Esto lo haremos con el parámetro “border”. Como en todos los parámetros que ya hemos visto escribiremos: border= “x” siendo la x un número. Ese número indicará el grosor del borde. Si no ponemos borde o lo escribimos “0”, la tabla no mostrará borde ninguno. Por supuesto, también podemos darle color al borde, escribiendo la etiqueta “bordercolor” e indicando el color que queramos para nuestro borde.
El parámetro “width” indicará la anchura de la tabla. Esta anchura la podemos poner en píxeles (width= “300”) o con porcentaje (width= “100%”).
Las filas: <tr>
Como hemos visto en el encabezado las filas se escriben gracias a las etiquetas <tr> con su correspondiente cierre </tr>. El contenido de las columnas que están dentro de la fila lo podemos alinear tanto horizontal como verticalmente.
Para alinearlo verticalmente utilizaremos el atributo “valign” para poder alinearlo arriba de la celda (“top”), en el centro (“middle”) o debajo (“bottom”).

Para alinearlo horizontalmente utilizaremos el atributo “align”. Con este atributo podremos alinear el contenido de las celdas en el centro (“center”), a la izquierda (“left”), a la derecha (“right”) o justificado (“justify”).
Por supuesto a las filas también les podemos definir el color de fondo (“bgcolor”) y el color del borde (“bordercolor”).
Las celdas <td>
Las celdas que van dentro de cada fila las tenemos que escribirlas con la etiqueta <td> y su correspondiente cierre </td>.

JavaScript

Aunque existen diferentes formas de incluir JavaScript en un documento html, la más común es mediante la etiqueta <script> y su correspondiente cierre: </script>. Además, deberemos indicar dentro de esa etiqueta el lenguaje (language=” JavaScript”) y el tipo (type=”text/JavaScript”). Dentro de esta etiqueta irá el programa JavaScript. En un mismo documento html podemos incluir varias etiquetas <script>. El único requisito es que todas ellas estén convenientemente cerradas.

Algunas funciones que vamos a utilizar en nuestros scripts:

alert(“mensaje”)	Sirve para mostrar un mensaje en la ventana del navegador de la pagina
prompt(“mensaje”,”texto predeterminado”)	Muestra un mensaje con un cuadro de texto para ingresar datos. Similar a un InputBox de Visual Basic
Confirm(“mensaje”)	Muestra un mensaje y da la opción de aceptar o cancelar para confirmar una acción
window.open(“url”)	Abre una ventana PopUp, de esas que tanto nos molestan.

window.close()	Cierra la ventana actual
window.print()	Nos va a dejar imprimir la ventana actual
document.oncontextmenu = function(){return false}	Con esto deshabilitamos el botón derecho del mouse