

Tutorial C# con programacion NET, incluye programas aspx, ejercicio, ejemplo y aplicacion con arreglos y base de datos en Access.

[WWW.PROGRAMACIONFACIL.COM](http://www.programacionfacil.com)

[I.- ELEMENTOS BASICOS](#)

[1.- INTRODUCCION](#)

[2.- MODELO DE SOLUCION](#)

[3.- VARIABLES](#)

[4.- DECLARACION Y TIPOS DE DATOS](#)

[5.- OPERADORES ARITMETICOS](#)

[6.- OPERADOR CAST](#)

[7.- JERARQUIA DE OPERACIONES](#)

[8.- CONCEPTOS BASICOS DE OOP](#)

[9.- MODELOS DE PROGRAMACION EN INTERNET](#)

[10.- INTRODUCCION A CSHARP NET:\(1\)](#)

[11.- CSHARP NET:\(2\)](#)

[12.- CSHARP NET:\(3\)](#)

[APENDICE 1A: OBJETOS PROPIOS HTML](#)

[APENDICE 1B: PALABRAS RESERVADAS CSHARP](#)

[APENDICE 1C: OBJETOS WEBCONTROLS](#)

APENDICE 1D: EXTRA WEBCONTROLS

II.- INSTRUCCIONES DE CONTROL DE PROGRAMA

1.- INTRODUCCION

2.- INSTRUCCIONES DE CONTROL DE PROGRAMA

3.- INSTRUCCIONES CONDICIONALES

4.- CONDICIONES SIMPLES

5.- OPERADORES RELACIONALES

6.- INSTRUCCION IF

7.- CONDICIONES COMPUESTAS

8.- INSTRUCCION SWITCH

9.- COMPONENTE LISTBOX Y DROPDOWNLIST

10.- COMPONENTE CHECKBOX Y CHECKBOXLIST

11.- COMPONENTE RADIOBUTTON Y RADIOBUTTONLIST

12.- CICLO FOR

13.- CICLO WHILE

14.- CICLO DO...WHILE

15.- CONCLUSIONES ACERCA DE CICLOS

III.- ARREGLOS

1.- INTRODUCCION

2.- ARREGLOS

3.- ARREGLOS TIPO LISTAS

4.- ARREGLOS TIPO TABLA

5.-ARREGLOS IRREGULARES

6.- LISTAS VISUALES(LISTBOX)

7.- TABLAS VISUALES(TABLE)

8.- APENDICE ASP.OLD

IV.- PROCEDIMIENTOS Y FUNCIONES

4.1.- PROCEDIMIENTOS

4.2.- PARAMETROS

4.3.- VARIABLES LOCALES Y GLOBALES

4.4.-FUNCIONES

4.5.- ARREGLOS COMO PARAMETROS

IV.- INT A LAS BASES DE DATOS

1.- INTRODUCCION

2.- MODELOS DE ALMACENAMIENTO DE DATOS

3.- TABLAS

[4.- TABLAS\(CONTINUACION\)](#)

[5.- ACCESS](#)

[6.- ADO.NET ACTIVE DATA OBJECT](#)

[7.- SELECCION O DESPLIEGUE](#)

[8.- INSERCCION O ADICION DE REGISTROS](#)

[9.- BUSQUEDAS](#)

[10.- FILTROS](#)

[11.- OPERACIONES CON CAMPOS](#)

[12.- BAJAS](#)

[13.- EDICION DE REGISTROS](#)

[14.- GRAFICOS O IMAGENES](#)

[APENDICE ADO](#)

UNIDAD I ELEMENTOS BÁSICOS

TEMA 1: INTRODUCCIÓN

From Ugly Duckling to Swan

Eli Cohen, Reconceptualizing Information Systems as a Field of the Transdiscipline Informing Science: From Ugly Duckling to Swan, Journal of Computing and Information Technology. 7 (3) 1999, 213-219

Hans Christian Anderson wrote a tale in which all the young ducks made fun of another. They made the duckling feel inadequate because he was different. One day a swan, the most beautiful of the fowl, declared that the youngster was in fact a young swan and a fine one at that.

*"Information Systems is the field of inquiry that attempts to **provide the business client with information in a form, format, and schedule** that maximizes its effectiveness."*

Información y Conocimiento son los dos elementos claves del nuevo milenio ninguna sociedad podrá alcanzar ni puede ignorar este nuevo esquema ya las naciones no se miden por su riqueza industrial, ni sus activos físicos, ni por su poder militar, sino por la cantidad de información que produce y consume, así como por la recombinación de información nueva en un conocimiento de grado superior.

Nuevos sistemas de información, tienden a ser cada vez de mayor alcance y complejidad sobre todo cuando se toman en cuenta la nuevas necesidades de información y conocimiento que demandan las nuevas organizaciones.

Nuevos sistemas de información son costosos en tiempos y recursos, la solución moderna de sistemas de información exigen herramientas y metodologías que resuelvan rápida, económica, eficiente y de manera global, problemas de información y conocimiento planteados por las

organizaciones.

Ademas el pleno potencial del hardware tampoco es aprovechado plenamente y existe un considerable retraso con el software y sus aplicaciones generando lo que se conoce como "crisis del software".

Actualmente el paradigma de programación se ha enfocado a nuevas necesidades de modernos y globales sistemas de información basados en redes y mas aun en la red global de internet, actualmente es mas importante poder concebir y construir sistemas de información con estas nuevas tecnologias de programación.

C SHARP es un lenguaje de programación desarrollado por Microsoft muy apropiado para construir sistemas de información basados en red o mejor aun en internet.

NET es la nueva tecnologia desarrollada y ofrecida por Microsoft que permite hacer mas facil la construcción y desarrollo de programas y aplicaciones para Internet.

El proposito del presente curso es enfocarse al ultimo modelo y les deseo mucha suerte a mis alumnos



WWW.PROGRAMACIONFACIL.COM

UNIDAD I ELEMENTOS BÁSICOS

TEMA 2: MODELO DE SOLUCIÓN

En general un problema de información es posible entenderlo analizarlo y descomponerlo en todos sus componentes o partes que de una u otra manera intervienen tanto en su planteamiento como en su solución.

Una herramienta rápida que nos permite descomponer en partes un problema para su solución es el llamado modelo de solución, esta consiste de una pequeña caja que contiene los tres elementos más básicos en que se puede descomponer cualquier problema sencillo de información, estas tres partes son:

LA PRIMERA PARTE son todos los datos que el computador ocupa para resolver el problema, estos datos son almacenados internamente en la memoria del computador en las llamadas variables de entrada.

LA SEGUNDA PARTE son todas las operaciones generalmente algebraicas necesarias para solucionar el problema, generalmente esta parte del modelo es una ecuacion algebraica o formula (o igualdad matemática, ej. $X = y + 5$).

LA TERCERA PARTE es el resultado o solución del problema que generalmente se obtiene de la parte de operaciones del modelo y dichos datos están almacenados en las llamadas variables de salida.

En resumen para todo problema sencillo de información es necesario plantearse las siguientes preguntas:

Que datos ocupa conocer el computador para resolver el problema y en cuales variables de entrada se van a almacenar?

Que procesos u operaciones debe realizar el computador para resolver el problema planteado?

Que información o variables de salida se van a desplegar en pantalla para responder al problema planteado originalmente?

Como nota importante no confundir los términos datos, variables e información:

Datos se refiere a información en bruto no procesada ni catalogada, por ejemplo "Tijuana", "calle primera # 213", "15 años", "\$2,520.00", etc.

Variables es el nombre de una localidad o dirección interna en la memoria del computador donde se almacenan los datos, ejemplo de variables para los casos del inciso anterior, CIUDAD, DIRECCIÓN, EDAD, SUELDO, ETC.

Información son datos ya procesados que resuelven un problema planteado.

EJEMPLO DE MODELO DE SOLUCIÓN

Construir un modelo de solución que resuelva el problema de calcular el área de un triángulo con la formula área igual a base por altura sobre dos.

| Variable(s) de Entrada | Proceso u Operación | Variable(s) de Salida |
|------------------------|-------------------------|-----------------------|
| BASE ALTURA | AREA= BASE * ALTURA / 2 | AREA |

PROBLEMA 2.- CONVERTIR LA EDAD EN AÑOS DE UNA PERSONA A MESES.

PROBLEMA 3.- CONVERTIR PESOS A DÓLARES.

PROBLEMA 4.- CALCULAR EL ÁREA DE UN CIRCULO CON LA FORMULA

$$AREA = PI * RADIO^2$$

PROBLEMA 5.- EVALUAR LA FUNCIÓN $Y = 5X^2 - 3X + 2$ PARA CUALQUIER VALOR DE X.

- Observar para el caso de constantes fijas o conocidas (PI) no se debe dar como dato de entrada su valor en cambio colocar directamente su valor dentro de la formula en la parte de operaciones del problema.
- Pero recordar también que existirán problemas sencillos donde:
- No se ocupan entradas o no se ocupan operaciones, pero todos ocupan salida.
- Una formula grande o muy compleja puede ser más segura y fácil de resolver, si es descompuesta y resuelta en partes, juntando al final los parciales para obtener el resultado final.
- Un problema puede tener más de una solución correcta.
- El problema no esta suficientemente explicado o enunciado, entonces, estudiarlo, analizarlo y construirlo de manera genérica.

PROBLEMAS SUGERIDOS

Construir los modelos de solución de los siguientes problemas:

PROBLEMA 6.- Convertir millas a kilómetros (caso normal)

PROBLEMA 7.- Convertir 125 metros a centímetros (no ocupa entradas)

PROBLEMA 8.- Se calcula que en promedio hay 4.7 nidos en cada árbol en la UABC, también se calcula que en cada nido existen un promedio de 5.8 pájaros, se pide calcular la cantidad total de nidos y de pájaros en los 227 arboles que existen en la UABC. (no ocupa entradas)

PROBLEMA 9.- La gorda Sra. López y sus 8 hijos solo compran una vez al mes su mandado en conocido supermercado, en dicha tienda el kilogramo de frijol cuesta \$8.75, el paquete de tortillas cuesta \$3.55 y el frasco de café vale \$14.25, si solo compran de estos tres productos para su mandado, calcular su gasto total.(problema no claro)

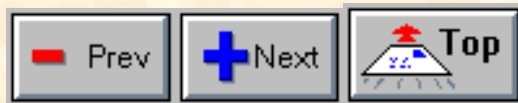
PROBLEMA 10.- Capturar y desplegar los cinco datos mas importantes de un automóvil (no ocupa operaciones)

PROBLEMA 11.- La distancia Tijuana - Ensenada es de 110 Kms, si un automóvil la recorre a una velocidad constante de 30 millas por hora,

cuanto tiempo tarda en llegar. (1 milla =1.609 Km.) (dos maneras correctas de resolverlo).

PROBLEMA 12.-Evaluar la función $y = 3x^2 + 2x - 5$ para cualquier valor de x.
(caso normal).

PROBLEMA 13.-Evaluar la función $y = -5x^3 - 3x^2 + 8$ para cuando x vale 4 .
(no ocupa entradas).



UNIDAD I: ELEMENTOS BÁSICOS

TEMA 3: VARIABLES

Identificadores son conjuntos de letras y/o números que se utilizan para simbolizar todos los elementos que en un programa son definibles por el usuario (programador o ingeniero de software) del mismo, como son las variables donde se almacenan datos, funciones(pequeños módulos con código), etiquetas, clases, objetos, etc.

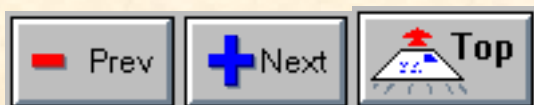
Una variable se define como un identificador que se utiliza para almacenar todos los datos generados durante la ejecución de un programa.

Existen ciertas reglas en cuanto a variables:

- Claras y con referencia directa al problema.
- No espacios en blanco, ni símbolos extraños en ellas.
- Se pueden usar abreviaturas, pero solo de carácter general.
- No deben ser palabras reservadas del lenguaje.

Ejemplos de buenas variables:

Nombre, Edad, SdoDiario, IngMensual, Perímetro, Calif1, etc.



UNIDAD I: ELEMENTOS BÁSICOS**TEMA 4: DECLARACION Y TIPO DE VARIABLES**

A toda variable que se use en un programa, se debera declarar de preferencia al principio del programa.

En C SHARP (tambien se le conoce como C#) existen los siguientes tipos de variables:

| C# Tipo | .Net Framework (System) type | Signed? | Bytes en Ram | Rango |
|--------------------|-----------------------------------------|----------------|-------------------------|----------------------------------------------------------------------------------------|
| sbyte | System.Sbyte | Yes | 1 | -128 a 127 |
| short | System.Int16 | Yes | 2 | -32768 a 32767 |
| int | System.Int32 | Yes | 4 | -2147483648 a 2147483647 |
| long | System.Int64 | Yes | 8 | -9223372036854775808 a 9223372036854775807 |
| byte | System.Byte | No | 1 | 0 a 255 |
| ushort | System.Uint16 | No | 2 | 0 a 65535 |
| uint | System.UInt32 | No | 4 | 0 a 4294967295 |
| ulong | System.Uint64 | No | 8 | 0 a 18446744073709551615 |
| float | System.Single | Yes | 4 | Aprox. $\pm 1.5 \times 10^{-45}$ a $\pm 3.4 \times 10^{38}$ con 7 decimales |
| double | System.Double | Yes | 8 | Aprox. $\pm 5.0 \times 10^{-324}$ a $\pm 1.7 \times 10^{308}$ con 15 o 16 decimales |
| decimal | System.Decimal | Yes | 12 | Aprox. $\pm 1.0 \times 10^{-28}$ a $\pm 7.9 \times 10^{28}$ con 28 o 29 decimales |
| char | System.Char | N/A | 2 | Cualquier caracter Unicode |

| | | | | |
|------|----------------|-----|-------|--------------|
| bool | System.Boolean | N/A | 1 / 2 | true o false |
|------|----------------|-----|-------|--------------|

En particular cada tipo de dato que se menciona aqui es en realidad un OBJETO, que se deriva a su vez de una clase que provee el framework de microsoft.net es por eso que se incluye la clase de la cual proviene el tipo de dato.

Es decir en un programa se podra declarar una variable por ejemplo `float pi;` o tambien se podra declarar y crear un objeto derivado de esa clase, por ejemplo `System.Float alfa = new System.Float();` para este caso observar y tener en cuenta dos cosas:

Observar como se declara y crea un objeto (este formato de creación de objetos aprenderlo bien).

Como objeto alfa podra usar todas las propiedades y metodos asociadas al objeto, mas adelante se ve un tema donde se analiza mas a fondo el concepto de clases y objetos.

Signed significa que se puede usar el signo + o - al usar la variable.

Por ultimo variables strings o variables cadena, se podran crear usando la clase STRING que creara un objeto de dicho tipo.

Para declarar un variable en un script o programa solo usar el siguiente formato:

Tipo de dato lista de variables; ejemplo:

```
string nombre, ciudad;
```

```
int alfa, beta;
```

```
string ciudad="tijuana";
```

```
float pi=3.1416;
```


Para el caso de objetos numericos derivados de la clase respectiva, solo usar el formato que se indico en los parrafos de arriba.

Recordar que csharp (c#) es case-sensitive, es decir reconoce la diferencia que hay entre mayusculas y minusculas, en otras palabras no declarar alfa e intentar capturar o desplegar ALFA.

Para convertir numeros a strings no hay problema, solo cargar o asignar el numero o variable numerica a la variable string, pero para convertir strings a numeros existen y deberan usarse los metodos Parse de las clases respectivasejemplo;

```
String beta1="100";
```

```
Int beta2 = System.Int32.Parse(beta1);
```

Las secuencias de escape que reconoce c#(csharp) y que se pueden usar dentro de una string son:

| Character | Escape Sequence |
|-----------------------------------------------------------------|-----------------|
| ' | \' |
| " | \" |
| \ | \\ |
| Alert | \a |
| Backspace | \b |
| Form feed | \f |
| New Line | \n |
| Carriage Return | \r |
| Horizontal Tab | \t |
| Vertical Tab | \v |
| A unicode character specified by its number e.g. \u200 | \u |
| A unicode character specified by its hexadecimal code e.g. \xc8 | \x |

| | |
|------|-----------|
| Null | \0 (zero) |
|------|-----------|



Prev



Next



Top

UNIDAD I: ELEMENTOS BÁSICOS**TEMA 5: OPERADORES ARITMÉTICOS**

Un operador es un símbolo especial que indica al compilador que debe efectuar una operación matemática o lógica.

CSharp reconoce los siguientes operadores:

| Category | Name | Syntax Example | Overloadable? |
|----------|------------------------------|--------------------------------|---------------|
| Primary | Grouping | (a+b) | No |
| | Member | A.B | No |
| | Struct pointer member access | A->B | No |
| | Method call | f(x) | No |
| | Post increment | c++ | Yes |
| | Post decrement | c-- | Yes |
| | Constructor call | c = new Coord(); | No |
| | Array stack allocation | int* c = stackalloc int[10] | No |
| | Struct size retrieval | sizeof (int) | No |
| | Arithmetic check on | checked {byte c = (byte) d;} | No |
| | Arithmetic check off | unchecked {byte c = (byte) d;} | No |
| Unary | Positive value | +10 | Yes |
| | Negative value | -10 | Yes |
| | Not | !(c==d) | Yes |

| | | | |
|------------------------|-------------------------------|---------------------|-----|
| | Bitwise complement | ~(int x) | Yes |
| | Pre increment | ++c | Yes |
| | Pre decrement | --c | Yes |
| | Type cast | (myType)c | No |
| | Value at address | int* c = d; | No |
| | Address value of | int* c = &d; | No |
| Type operators | Type equality / compatibility | a is String | No |
| | Type retrieval | typeof (int) | No |
| Arithmetic | Multiplication | c*d | Yes |
| | Division | c/d | Yes |
| | Remainder | c%d | Yes |
| | Addition | c+d | Yes |
| | Subtraction | c-d | Yes |
| | Shift bits right | c>>3 | Yes |
| | Shift bits left | c<<3 | Yes |
| Relational and Logical | Less than | c<d | Yes |
| | Greater than | c>d | Yes |
| | Less than or equal to | c<=d | Yes |
| | Greater than or equal to | c>=d | Yes |
| | Equality | c==d | Yes |
| | Inequality | c!=d | Yes |
| | Bitwise and | c&d | Yes |
| | Bitwise or | c d | Yes |
| | Logical and | c&& d | No |
| | Logical or | c d | No |
| | Conditional | int c=(d<10) ? 5:15 | No |

De momento nos concentramos en los operadores aritmeticos, pero

dejamos esta tabla de operadores para usarla a lo largo del curso.

Recordar que en c# cuando se dividen dos enteros, csharp trunca la parte residual, es decir si se realiza la siguiente operación:

```
float alfa= 10 / 3;
```

desplegar alfa-->sale 3 en pantalla

Es decir csharp truncó el residuo de la division entre enteros, no importa el tipo de variable (primero se realiza la operación a la derecha y luego carga la variable a la izquierda, es decir para cuando quiere cargar la variable el residuo ya no existe) para arreglar este problema usar el siguiente metodo:

```
float alfa= 10/3.0; observar que ya no esta dividiendo enteros, sino un entero entre un decimal.
```

El operador modulo o remanente (%) devuelve el residuo entero de una división entre enteros, ejemplo;

```
// área de declaración
```

```
int o float alfa;
```

```
// área de operaciones
```

```
alfa = 23 % 4;
```

```
// área de despliegue
```

```
desplegar alfa; ---> El resultado en pantalla es 3
```

Otro ejemplo;

```
alfa = 108 % 10;
```

```
desplegar alfa; --> El resultado en pantalla es 8
```

Para resolver los problemas de potencias y raíces se usan ciertas instrucciones especiales que proporciona el lenguaje llamadas funciones matemáticas, en chsarp existe toda una librería o mas

correctamente dicho, una clase especializada en instrucciones o funciones matemáticas (System.Math).

Recordar que todas las funciones reciben uno o más datos o valores y regresan siempre un resultado, una de estas funciones matemáticas es:

a) Potencias por ejemplo 5^2 se resuelve usando el objeto MATH y su metodo Pow(base,exp).

System.field o metodo

Public Fields

E Represents the natural logarithmic base, specified by the constant, e.

PI Represents the ratio of the circumference of a circle to its diameter, specified by the constant, π .

Public Methods

Abs Overloaded. Returns the absolute value of a specified number.

Acos Returns the angle whose cosine is the specified number.

Asin Returns the angle whose sine is the specified number.

Atan Returns the angle whose tangent is the specified number.

Atan2 Returns the angle whose tangent is the quotient of two specified numbers.

Ceiling Returns the smallest whole number greater than or equal to the specified number.

Cos Returns the cosine of the specified angle.

Cosh Returns the hyperbolic cosine of the specified angle.

Exp Returns e raised to the specified power.

Floor Returns the largest whole number less than or equal to the specified number.

IEEERemainder Returns the remainder resulting from the division of a specified number by another specified number.

Log Overloaded. Returns the logarithm of a specified number.

Log10 Returns the base 10 logarithm of a specified number.

Max Overloaded. Returns the larger of two specified numbers.

Min Overloaded. Returns the smaller of two numbers.

Pow Returns a specified number raised to the specified power.

Round Overloaded. Returns the number nearest the specified value.

Sign Overloaded. Returns a value indicating the sign of a number.

Sin Returns the sine of the specified angle.

Sinh Returns the hyperbolic sine of the specified angle.

Sqrt Returns the square root of a specified number.

Tan Returns the tangent of the specified angle.

Tanh Returns the hyperbolic tangent of the specified angle.

Fuente: documentacion de .NET FRAMEWORK

Para el ejemplo a seguir, el de potencias esta función ocupa dos valores o datos(base y exp) ambos de tipo double y regresa un

resultado también de tipo double, ejemplo;

```
double pot;
```

```
pot = System.Math.Pow(5,2);
```

```
desplegar pot;
```

b) Raíces solo recordar la ley de exponentes que dice:

$$\sqrt[n]{a^m} = a^{m/n}$$

$\sqrt[3]{5^8}$ = usar `System.Math.Pow(5,8/3.0);`

$\sqrt{9}$ = usar `System.Math.Pow(9,1/2.0);`

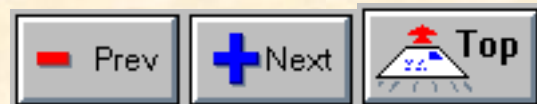
PROBLEMAS SUGERIDOS

1.- expresar las siguientes funciones en notacion algebraica de csharp

1.- $y = 4x^3 - 3x^2 + 2x - 5$

2.- $y = 5\sqrt[3]{x} + 4x^2 + 6$

3.- $y = -8\sqrt[5]{x^3} - 3x^3 + 6x^2 - 7$



UNIDAD 1: ELEMENTOS BASICOS

TEMA 6: OPERADOR CAST

Se puede forzar un dato, variable o una expresión a convertirse o cambiarse a un nuevo tipo de dato.

El operador cast realiza este proceso es decir convierte datos, variables o expresiones a un nuevo tipo de dato, su formato es:

```
(nvotipo) dato, var, exp;
```

Ejemplo:

```
// declaración  
  
int alfa;  
  
// Asignación  
  
alfa=20;  
  
// Cambio de tipo  
  
(double)alfa;
```

Ejemplo:

```
(int)3.1416;
```

En este ejemplo se está convirtiendo un float a int recordar que en este caso ya no se tendrán los decimales.

Como nota importante este operador resuelve los dos problemas pendientes:

- a. El de la división entre enteros.
- b. El tipo de dato específico que requieren las funciones.

Ejemplos:

a) // Declaración

```
float alfa;  
  
// Operación  
  
alfa = float (23)/5;  
  
// Pero en estos casos es preferible usar un puntodecimal  
  
alfa=23/5.0;
```

En toda división recordar agregar a uno de los dos valores el (.0), solo que los dos elementos sean variables entonces usar el operador cast con una de ellas.

b)// Declaración

```
double potencia;  
  
// Operación
```



```
potencia = Pow ( (double)5, (double)3);
```

Como se observa en el ejemplo se puede usar Pow() directamente con los datos, argumentos o parámetros requeridos si estos son numéricos pero transformándolos con el operador cast.

Recordar que es **Pow <-- P GRANDOTA.**

Esto también va a permitir evaluar expresiones matemáticas de manera mas directa y sencilla, solo recordando usar un Pow() por cada potencia y cada raíz de la ecuación, ejemplo:

Sea $y = 3x^3 - \sqrt[3]{x} + 4x^2$ dentro de un programa esto se resuelve con;

```
// área de declaración de variables
```

```
double y, x;
```

```
// área de captura de datos
```

```
capturar el valor de x;
```

```
// área de operaciones
```

```
y = 3 * Pow(x, (double)3) - Pow(x, (1/3.0)) + 4 * Pow(x, (double)2 );
```

```
// área de despliegue de resultados
```

```
desplegar x, y
```

PROBLEMAS SUGERIDOS POW():

1. $y = 4x^3 - 3x^2 + 2x - 5$

2. $y = 5\sqrt[3]{x} + 4x^2 + 6$

3. $y = -8\sqrt[5]{x^3} - 3x^3 + 6x^2 - 7$



WWW.PROGRAMACIONFACIL.COM

UNIDAD I: ELEMENTOS BÁSICOS

TEMA 7: JERARQUÍA DE OPERACIONES

El problema de no tomar en cuenta la jerarquía de los operadores al plantear y resolver una operación casi siempre conduce a resultados muchas veces equivocados como estos:

Ejemplos: a) $2 + 3 * 4 = 20$ (incorrecto)

.....= 14 (correcto)

b) si $\text{calif1}=60$ y $\text{calif2}=80$

entonces si en programa se usa

$\text{promedio} = \text{calif1} + \text{calif2} / 2$

da como resultado promedio = 100

Recordar siempre que antes de plantear una formula en un programa se deberá evaluar contra el siguiente:

Orden de operaciones:

1.- Paréntesis

2.- Potencias y raíces

3.- Multiplicaciones y divisiones

4.- Sumas y restas

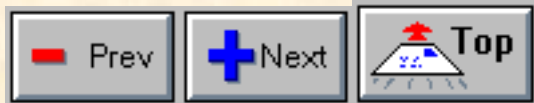
5.- Dos o más de la misma jerarquía u orden entonces resolver de izquierda a derecha

Nota: Si se quiere alterar el orden normal de operaciones entonces usar paréntesis.

Nota: Tampoco es bueno usar paréntesis de mas en una operación esto solo indica que no se evaluó bien la formula como en el siguiente ejemplo;

$$\text{área} = (\text{base} * \text{altura}) / 2$$

Aquí los paréntesis están de mas porque por orden de operaciones, multiplicación y división tienen la misma jerarquía y entonces se resuelven de izquierda a derecha, en otras palabras ni que falten paréntesis ni que sobren paréntesis.



WWW.PROGRAMACIONFACIL.COM

UNIDAD I: ELEMENTOS BÁSICOS

TEMA 8: CONCEPTOS BÁSICOS DE OOP

Para nuestro propósito en general un objeto puede definirse como cualquier cosa, ente o entidad física o lógica de información.

En este sentido todos los elementos materiales o inmateriales pueden clasificarse como objetos.

En particular cualquier objeto considerado presenta los siguientes tres elementos:

a) **Propiedades:** Son las características propias de un objeto estas propiedades o atributos son los que permiten diferenciar o individualizar un objeto de otro objeto ya sea de la misma o diferente clase o categoría.

Las propiedades mas generales son forma, color, tamaño, peso, etc., pero ya en particular:

Chamarra -> Marca, material, precio, color, tamaño, etc

Alumno -> Matricula, nombre, edad, domicilio, etc.

Gato -> Raza, nombre, color, edad, etc.

VentanaWindows-->Tamaño, Color, font, etc.

b) **Métodos:** Son las conductas propias de la naturaleza del objeto.

Así como las propiedades son el ser (que es) del objeto, los métodos son el hacer (que hacer) del objeto.

ejemplo de métodos:

Gato ---> Maullar(), comer(), correr(), saltar(), etc.

Alumno---> Estudiar(), comer(), asistir clase(), pintar()

Cuaderno-->Esescrito(), esrayado(), esborrado(), etc.

VentanaWindows--> Abrir(), cerrar(), maximizar(), etc....

c) **Eventos:** Es la relación (de varias maneras) que se puede dar entre dos objetos ya sean de la misma o diferente clase.

Un evento se manifiesta como un interacción entre dos objetos, en general al momento de la relación al mismo tiempo se dará una reacción o respuesta por parte de los dos objetos que se manifiestan como una serie, cadena o conjuntos de métodos propios que se activan o disparan, ejemplo:

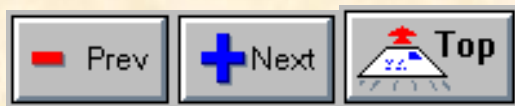
| Evento | Relación | Métodos que se activan |
|-----------------------|----------|---------------------------------|
| gato detecta gata | detectar | maullar(), correr(), oler() |
| gato detecta perro | detectar | bufar(), saltar(), correr() |
| maestro enseña alumno | Enseñar | pasar lista(), preguntar(), etc |
| Raton click Windows | click | maximizar(), cerrar() |
| Raton dblclk Windows | dblclk | minimizar(), etc |

Un Programa o un SCRIPT en csharp se puede considerar como un conjunto

de una o mas paginas o formas, donde cada una de ellas contiene un conjunto de objetos, componentes o controles.

Un componente o propiamente dicho un control es un objeto que se especializa en una tarea especifica por ejemplo hay controles especializados en desplegar textos o mensajes, otros controles se especializan en desplegar imágenes o vídeos, otros en manipular directorios o archivos en disco, etc.

Pero en general tanto las formas como los controles no dejan de ser objetos en programación y por tanto tienen sus propiedades, métodos y están sujetos a eventos.



WWW.PROGRAMACIONFACIL.COM

UNIDAD 1: ELEMENTOS BASICOS

TEMA 9: MODELO DE PROGRAMACION EN INTERNET

CSHARP C# (de momento lo entenderemos como programas en Csharp) y ASP.NET (Active Server Pages) son programas hechos para ejecutarse en la red de redes es decir en internet y mas apropiado en servidores de paginas(web server).

En este modelo, minimo se ocupan dos computadoras a la primera le llamamos **servidor** y es su mision proporcionar paginas y algunos servicios a las segundas maquinas, este servidor tiene en ejecución constante un programa llamado servidor de paginas (web server).

La segunda maquina, le llamamos **cliente** y el unico programa que tiene en ejecución es un programa o compilador llamado browser de los cuales el mas comun es el internet explorer, mediante el browser el cliente sube a una maquina servidora y pide una pagina.html almacenada en la servidora y el propio cliente a esta pagina.html la compila y la despliega dentro de la ventana del browser.

Para entender ASP.NET, tenemos que entender algunas cosas elementales de esta tecnologia de MicroSoft.

En principio ASP es un programa que es ejecutado por un servidor de paginas y sus resultados son enviados a el browser de la maquina cliente.

Este programa que por cierto tiene por extensión aspx (ejemplo prog15.aspx) es un conjunto de objetos que pueden provenir de varias fuentes distintas, las mas comunes son:

A) Objetos HTML.- Son los objetos mas elementales que puede contener o construirse en cualquier pagina o forma html en internet, por

ejemplo los input text, input submits, etc, (recordar que como objetos tienen sus propiedades y metodos que hay que cargar o programar ver apendice a final del capitulo).

B) Objetos ASP.- Son objetos propios de esta tecnologia y generalmente estan especializados en comunicacion entre formas o paginas html, pero tambien tiene objetos especializados en archivos, directorios, etc ver apendice al final.

C) Objetos ACTIVEX.- Componentes o controles especializados en muchas tareas hechos por MicroSoft para sus lenguajes visuales de programacion, ejemplos textbox, combobox, grids, etc ver apendice al final de la unidad.

D) Objetos ADO.- Active Data Object, componentes, controles u objetos especializados en la manipulaci3n de bases de datos, entre ellas sqlserver, access, etc.

E) Objeto DOM.- Document object model, un objeto estandar y especializado en manipular una pagina html.

F) Objetos NET.- Nuevos objetos que facilitan aun mas la construcci3n de programas en internet, de momento entender que son una combinaci3n de asp-activex.

Sin embargo recordar que todos estos objetos de distinta fuente deberan ser manipulados por algun lenguaje de programaci3n, es mediante instrucciones en este lenguaje que se puede cargar propiedades o activar metodos o programar eventos.

Los lenguajes basicos que microsoft incluyo en su primera versi3n de ASP, son los llamados LENGUAJES SCRIPTS y los mas comunes fueron al principio VisualBasicScript, JavaScript y por esfuerzos de personas y compa1as muy responsables PERLSCRIPT y actualmente JSCRIPT.

En enero del 2002 microsoft libero su nueva tecnologia que denomino ASP.NET que a diferencia de los 7 objetos del asp viejo, incorpora mas de 3700 objetos y a diferencia de los tres lenguajes scripts que soportaba el asp viejo, asp.net soporta muchos lenguajes de programaci3n incluyendo cobol (estarse pendiente de www.programacionfacil.com que pronto incluire muchos cursos de asp.net en muchos lenguajes de programaci3n diferentes)

En este curso nos especializamos en la construcción de programas usando la mas nueva tecnologia de Microsoft.Net y como lenguaje de trabajo C SHARP



UNIDAD 1: ELEMENTOS BASICOS

TEMA 10: INTRODUCCION A CSHARP NET(1)

Tomar nota que el modelo de programación que vamos a seguir, indica que un programa es un conjunto de objetos provenientes de cualquier fuente (html, activex, asp, ado, dom, asp.net) y se utiliza el lenguaje CHSARP NET para programar sus propiedades, metodos y eventos.

El primero modelo a aprender y programar, es el mas sencillo de todos solo contiene instrucciones y algunos objetos HTML asi como codigo en csharp, para programar estos objetos html y solucionar el problema.

Codigo progl.aspx:

```
<HTML>
```

```
<H1>BIENVENIDO A CSHARP </H1>
```

```
<B>MI PRIMER ASPX<BR>
```

```
<FORM RUNAT=SERVER>
```

```
EDAD<INPUT TYPE=TEXT ID=EDAD RUNAT=SERVER/><BR>
```

```
MESES<INPUT TYPE=TEXT ID=MESES RUNAT=SERVER /><BR>
```

```
<INPUT TYPE=BUTTON TEXT=OK ONSERVERCLICK=EVENTO1 VALUE=OK RUNAT=SERVER/>
```

```
</FORM><BR>
```

```
SALUDOS Y DESPEDIDA
```

```
</HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
int edad = Int32.Parse(EDAD.Value);
```

```
edad=edad*12;
```

```
MESES.Value=edad.ToString();
```

```
}
```

```
</script>
```

Notas:

1.- Lo primero y mas importante a recordar es que en este modelo de programación, el codigo en Csharp esta junto con codigo Html.
(empezar consiguiendo y estudiando un buen tutorial de html).

2.- Este codigo mezcla instrucciones de dos lenguajes de programación diferentes, ellos son HTML y CSHARP.

En otras palabras aparte de aprender el lenguaje de programación csharp, tambien por el mismo precio van a aprender el lenguaje de programacion HTML, en ninguna parte del mundo se proporciona esta oferta.

3.- Para crear este programa deberan usar el mejor editor de programas del mundo el NOTEPAD o el WORDPAD DE WINDOWS cargarlo, escribir el programa y grabarlo como prog1.aspx, tener mucho cuidado que la extensión sea .aspx, notepad y wordpad tienen la costumbre de agregarles aparte la extension .txt o .doc, es decir si el programa queda grabado como prog1.aspx.txt, **DICHO PROGRAMA NO SE VA A EJECUTAR**, revisarlo desde una consola desde el MSDOS con una orden

DIR y si es necesario usar RENAME.

4.- La primera parte del programa (el codigo html) empieza creando un objeto o componente FORM(tambien de html) que contendra todos los demas objetos o componentes de html, luego se crean dos objetos tambien provenientes de html (input text) objetos o componentes que se especializan en manipular datos, es decir se usan para capturar o desplegar datos.

5.- Despues se crea otro objeto o componente HTML(input button) que se puede definir como el objeto o componente de orden o comando, es decir en su evento onclick(cuando el usuario hace un click dentro de el) se activa la forma y el metodo en csharp asociado a ella, **observar y aprender y respetar el formato** de como asociarle codigo en csharp al evento onclick del componente botton.

6.- Hasta esta parte del programa, se esta manipulando puros objetos html (form, input text, input button) al finalizar este capitulo viene un apendice con todos los objetos html incluyendo sus propiedades y sus metodos, favor de revisarlos y estudiarlos.

7.- Ya que estan de regreso de analizar los objetos HTML, los mas observadores de ustedes notaran que en todos estos objetos que estan en progl.aspx, usan una serie de propiedades que en el apendice no se muestran, esta es la primera aportacion que la tecnologia Microsoft. Net agrega a este modelo de programación.

8.- El problema es que los objetos HTML no tienen suficientes propiedades y metodos para resolver muchos problemas de programación, por esta razon Microsoft.Net crea unas **cubiertas?(WRAPPERS)** para cada uno de ellos, en su libreria **System.Web.UI.Control**, o coleccion **HTMLCONTROL**, estos nuevos objetos o controles son:

HtmlInputButton, HtmlInputCheckBox, HtmlInputFile, HtmlInputHiden, HtmlInputImage, HtmlInputRaddioButton, HtmlInputText, HtmlAnchor, HtmlButton, HtmlForm, HtmlGenericControl, HtmlSelect, HtmlTable, HtmlTableCell, HtmlTableRow, HtmlTextArea, HtmlImage.

Para los observadores, notar que hay un objeto correspondiente para cada objeto HTML pero tambien EXISTEN ALGUNOS NUEVOS QUE FACILITAN LA CONSTRUCCION DE PAGINAS o FORMAS aspx.

Recordar tambien que estos controles aceptan o pueden usar todas las propiedades originales de los objetos HTML y algunas nuevas propiedades y metodos como se observa en el programa ejemplo.

La unica propiedad que no se menciona y se usa en todos estos objetos `htmlcontrol`, es `runat="server"` que se utiliza para indicarle al servidor de paginas que debe compilar y ejecutar el `programa.aspx` usando el lenguaje apropiado y solo mandarle el resultado de esta compilación a la maquina cliente que lo pidio.

9.- El script o miniprograma empieza con `<SCRIPT` y termina con `</SCRIPT>`.

10.- Recordar que CSHARP es case-sensitive es decir diferencia entre mayusculas y minusculas por tanto se debera respetar todas las ID de los objetos y las variables que se hayan declarado.

11.- La primera sorpresa agradable comparada con los otros modelos de programación es que de los objetos `htmlcontrols` se pueden manipular directamente sus propiedades y metodos, es decir ya no hay necesidad (aunque se puede) de usar los objetos tradicionales de ASP.

12.- Se empieza creando una variable entera normal para almacenar el valor de la edad solo recordar que el objeto `EDAD.Value` tiene su dato o valor de tipo `string` y por tanto se tiene que convertir a un valor entero para poder realizar aritmetica dentro del programa, esta conversión como se indico en el tema de TIPOS DE DATOS usa el metodo `Parse`, observar y respetar el formato.

13.- Ya realizado el calculo, se manda el resultado al objeto `MESES.Value` y recordar que la propiedad `VALUE` solo manipula puras strings, por eso la variable numerica se convierte a `ToString()`.

Los mas observadores de ustedes se preguntaran como es posible asignarle o asociarle un metodo a una variable normal, esto se hace porque CSHARP no tiene tipos de datos normales todos los tipos vistos en el tema correspondiente de tipos de datos son en realidad objetos derivados de una clase numerica misma que se puso a un lado del tipo

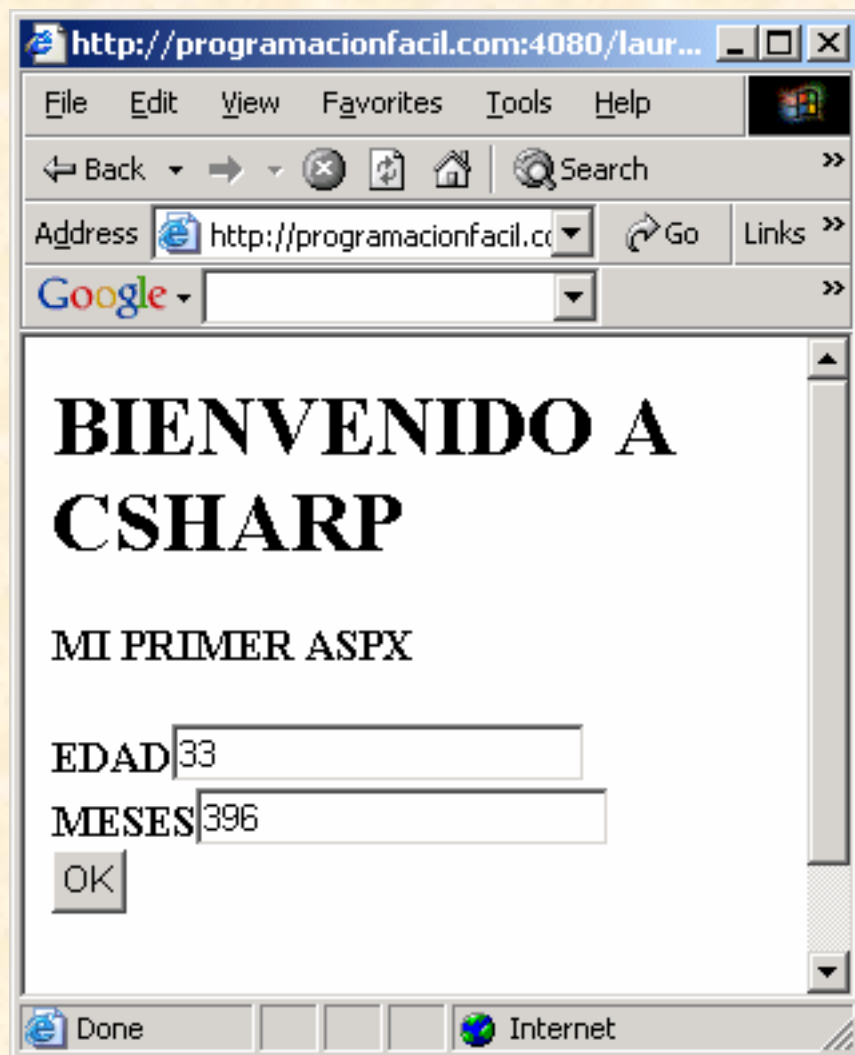
en ese tema ya analizado.

14.- Para ejecutar este programa, solo basta crearlo con el wordpad o notepad, grabarlo como prog1.aspx y para verlo en ejecución:

Subirlo a tu sitio en [programaciónfacil.com](http://programacionfacil.com) con el ftp del internet explorer y llamarlo desde cualquier parte del mundo con la dirección: <http://programaciónfacil.com:4080/tusitio/prog1.aspx>

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

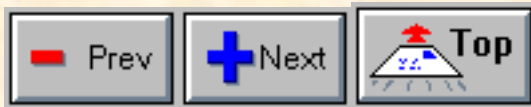
El programa en ejecución:



PROBLEMAS SUGERIDOS:

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- Construir programas para la primera mitad de los problemas del modelo de solución.



UNIDAD 1: ELEMENTOS BASICOS

TEMA 11: CSHARP(2)

En este segundo modelo de CHSARP se introduce una nueva coleccion de objetos denominada WEBCONTROLS.

Aunque los objetos derivados de HTMLCONTROL facilitan la tarea de manipular los objetos originales de HTML, Microsoft decidio que un nuevo conjuntos denominado WEBCONTROLS derivados de la libreria SYSTEM.WEB.UI. CONTROL permite una mejora mayor y mas funcionalidad para el trabajo con paginas, estos nuevos objetos son:

[VER APENDICEWEBCONTROLS AL FINAL DEL CAPITULO:](#)

Como se puede apreciar existen muchos nuevos objetos de mucha utilidad para la construccion de sistemas de informaci3n, todos estos objetos **tienen sus propiedades especificas**, sin embargo muchos de ellos comparten las siguientes propiedades:

| PROPIEDAD | DESCRIPCION |
|-------------|----------------------------------------|
| BackColor | Carga o lee el color de background |
| BorderColor | Carga o lee el color del marco(border) |
| BorderStyle | Carga o lee el estilo del marco |
| BorderWidth | Carga o lee el tama1o del marco |
| Font | Carga o lee el font |
| ForeColor | Carga o lee el color del foreground |
| Height | Carga o lee la altura del control |

| | |
|----------|------------------------------------------|
| ID | Carga o lee el identificador del control |
| TabIndex | Carga o lee el tab index |
| ToolTip | Carga o lee el tooltip del control |
| Visible | Carga o lee su estado visible |
| Width | Carga o lee la anchura del control |

Interesante verdad?

Prog2.aspx

```
<HTML>
```

```
<H1>BIENVENIDO A CSHARP </H1>
```

```
<B>MI SEGUNDO ASPX<BR>
```

```
<FORM RUNAT=SERVER>
```

```
EDAD.....:<ASP:TEXTBOX ID=EDAD RUNAT=SERVER/><BR>
```

```
MESESLABEL...:<ASP:LABEL ID=MESES1 RUNAT=SERVER/><BR>
```

```
MESESLITERAL:<ASP:LITERAL ID=MESES2 RUNAT=SERVER/><BR>
```

```
<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/>
```

```
</FORM><BR>
```

```
SALUDOS Y DESPEDIDA
```

```
</HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
int edad = Int32.Parse(EDAD.Text);
```

```
edad=edad*12;

MESES1.Text=edad.ToString();

MESES2.Text=edad.ToString();

}

</script> </html>
```

- 1.- Lo primero y mas importante a recordar que los scripts deben estar embebidos o empotrados en una pagina aspx y deben ir entre los tags `<script>codigocsharp</script>`.
- 2.- Este codigo mezcla instrucciones y objetos de dos lenguajes de programación diferentes ellos son HTML y Csharp.
- 3.- Considerar la pagina como una sola forma o ventana empotrada en el browser esta forma contiene tres objetos, controles o componentes provenientes de WEBCONTROLS (TEXTBOXS, LABEL, BUTTON) este ultimo control (button) se usa para activar el codigo o script del programa.
- 4.- Observar que se debera usar el tag `<ASP: WEBCONTROL PROPIEDADES />` para poner cada webcontrol en la pagina.
- 5.- Como se observa todos estos controles son objetos y por tanto tienen propiedades y metodos que son los que usamos dentro del programa observar el formato para procesarlos, es decir: objeto.propiedad o metodo.
- 6.- Se muestran dos tipos de controles para desplegar datos, textos o mensajes estaticos, ellos son LABEL y LITERAL (la diferencia entre ellos revisar en el apendice pero en general tiene mas y mejores propiedades LABEL), todos estos controles incluyendo TEXTBOX estan usando su propiedad TEXT para procesar los datos, del control BUTTON estamos usando su evento onclick, para pegarle el proceso u operación.
- 7.- Observar que lo primero que hace csharp, es detectar el EVENTO1 onclick del boton y realizar directamente las operaciones con los objetos webcontrol.
- 8.- Observar con cuidado todas las partes en minusculas, ES DECIR ES IMPORTANTE RECORDAR QUE DENTRO DEL CODIGO HTML NO IMPORTAN MAYUSCULAS O

MINUSCULAS, PERO DENTRO DEL CODIGO EN CSHARP SI IMPORTAN MAYUSCULAS Y MINUSCULAS.

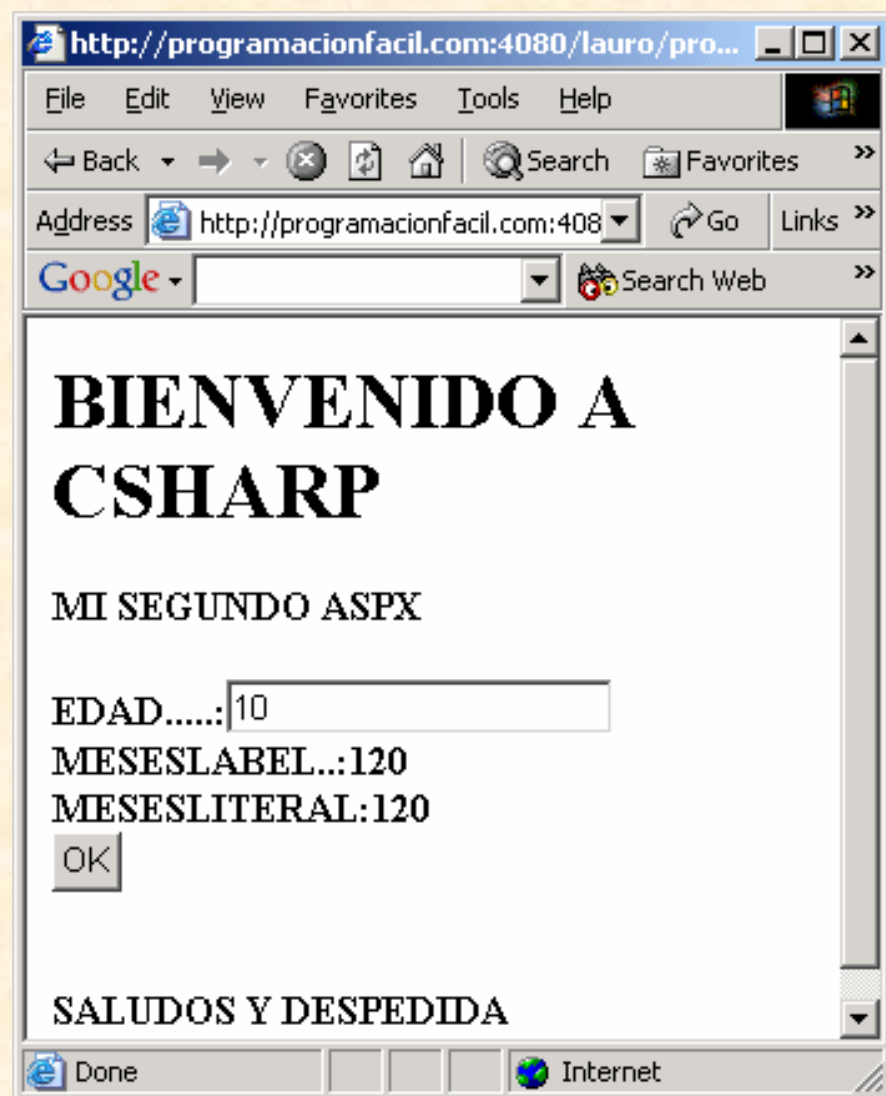
Para ejecutarlo:

1.- Solo subir prog2.aspx a tu sitio web en nuestro servidor de paginas y para verlo, bajarlo y ejecutarlo desde una maquina cliente, solo usar la siguiente dirección:

`http://programacionfacil.com:4080/tusitio/prog2.aspx`

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

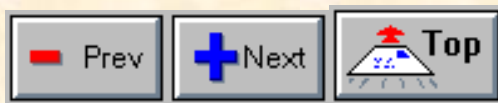
El programa en ejecución:



PROBLEMAS SUGERIDOS:

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- Construir scripts para la segunda mitad de los problemas que se vierón en el tema de modelo de solución, usar labels en unos y literal en otros.



WWW.PROGRAMACIONFACIL.COM

UNIDAD 1: ELEMENTOS BASICOS

TEMA 12: CSHARP(3):

Tercer modelo de script, PARA ESTE MODELO SE SEPARAN LA PARTE DE INTERFASE CON EL USUARIO (LA FORMA Y SUS CONTROLES) Y LA PARTE PROGRAMATICA en programas o archivos diferentes.

Este modelo es mas comun y practico para la construcción modular de programas.

Prog3.aspx

```
<%@ PAGE INHERITS=PROG3 SRC=PROG3.CS %>

<HTML>

<H1>BIENVENIDO A CSHARP </H1>

<B>MI TERCER ASPX<BR>

<FORM RUNAT=SERVER>

EDAD<ASP:TEXTBOX ID=EDAD RUNAT=SERVER/><BR>

MESES<ASP:TEXTBOX ID=MESES RUNAT=SERVER/><BR>

<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/>

</FORM><BR>

SALUDOS Y DESPEDIDA

</HTML>
```

notas:

1.- Comienza con una directiva o instrucción PAGE que en principio es el compilador de paginas de ASP.NET y su atributo INHERITS le indica a dicho compilador que use prog3.aspx y el codigo fuente o SouRcE que esta en el archivo llamdo prog3.cs.

Aparte de la directiva PAGE, existen las siguientes:

ASP.NET PAG-LEVEL DIRECTIVES:

| DIRECTIVE | DESCRIPCION |
|---------------|-------------------------------------------------------------------------------|
| @ Page | Define atributos usados para compilar paginas ASP.NET |
| @ Control | Define atributos usados para compilar controles de usuarios |
| @ Import | Imporrta NAMESPACES desde la libreria de clases de .NET |
| @ Register | Define alias, tags, y otros parametros para constroles de usuarios y normales |
| @ Assembly | Identifica otros archivos (assemblies) para enlazar a esta pagina |
| @ OutputCache | Define parametros para el cache de salida de HTML |

2.- El resto de instrucciones son las normales vistas en el tema anterior.

Prog2.cs

```
using System;
```

```
using System.Web;
```

```
using System.Web.UI;
```

```
using System.Web.UI.WebControls;

public class PROG3 : Page

{

// CREANDO Y ENLAZANDO CONTROLES A FORM.ASPX

protected TextBox EDAD;

protected TextBox MESES;

// programando evento clik de prog3.aspx

public void EVENTO1 (Object sender, EventArgs e)

{

int edad = Int32.Parse(EDAD.Text);

edad=edad*12;

MESES.Text=edad.ToString();

}

}
```

notas:

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- Este programa empieza importando o usando las librerias de clases apropiadas para este problema:

SYSTEM:= libreria mas generica de chsharp, incluye todas las definiciones del lenguaje csharp.

SYSTEM.WEB.UI.WEBCONTROLS:= Como se indico en tema anterior esta libreria

incluye las definiciones de todos los objetos o controles de tipo WEBCONTROL.

2.- Empieza el programa .CS creando y enlazando los objetos que se tienen en PROG3.ASPX y observar que su ID es el mismo tanto en prog3.aspx como en prog3.cs

3.- Un programa .CS debe ser una clase, en este caso la clase se llama PROG3 y es de tipo o se deriva de la clase PAGE.

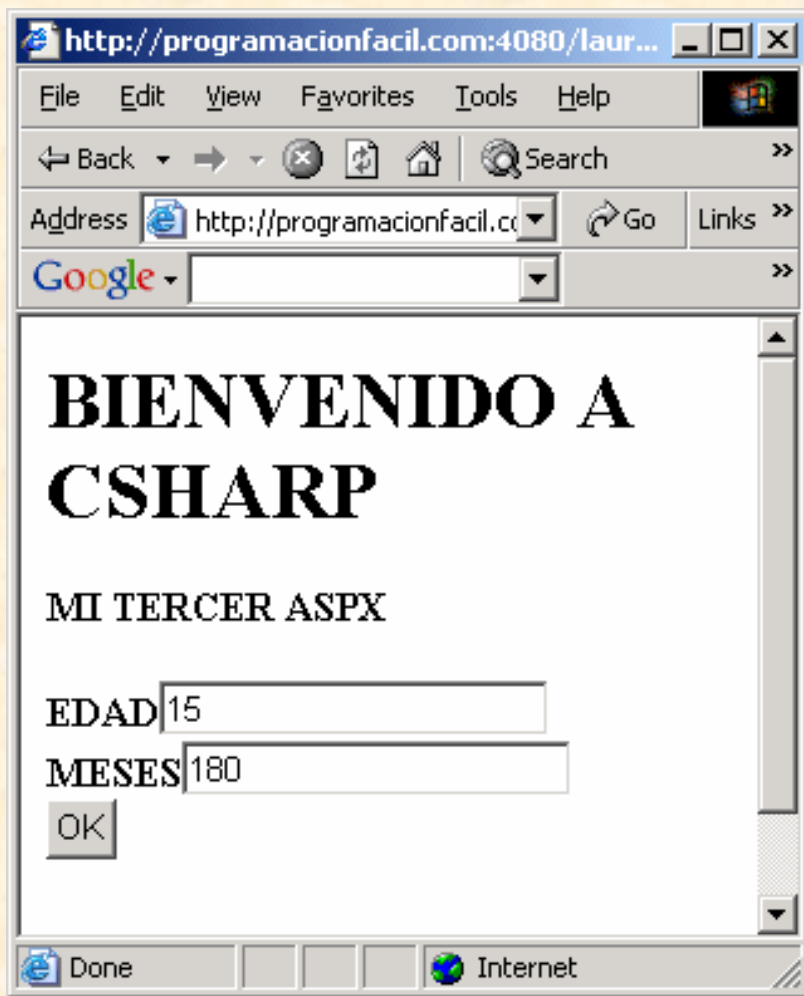
3.- Toda clase debe llevar metodos (de preferencia MAIN()), sin embargo para este caso basta crear el metodo onclick del componente button de aspx usando el nombre del este evento(EVENTO1).

4.- Este metodo lleva el codigo normal de chsharp que ya se analizo en los dos temas anteriores, CON LA EXCEPCION DE QUE AHORA USAMOS DOS TEXTBOXS, EN LUGAR DE UN LABEL PERO ES INDISTINTO USAR TEXTBOX O LABEL PAA DESPLIEGUE, AUNQUE ES MAS CORRECTO USAR LABEL'S PARA DESPLIEGUES.

5.- Ambos programas (prog3.aspx y prog3.cs) deberan subirse a tu sitio en programaciónfacil.com y pedir el aspx de manera normal, es decir <http://programacionfacil.com:4080/tusitio/prog3.aspx>.

6.- Recordar que cuando un cliente solicita este tipo de aspx, el servidor enlaza, compila y ejecuta los dos programas a la vez y le manda el resultado al browser del cliente.

Corrida prog3.aspx y prog3.cs

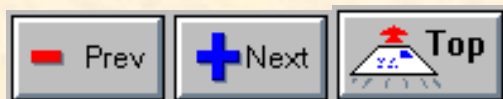


Aunque es mas laborioso tener el programa en dos archivos o fuentes diferentes, a la larga es mejor este esquema de trabajo sobre todo en aquellos sistemas muy grandes es decir, por ejemplo un problema de programación muy grande es mejor resolverlo construyendo muchos CS's y desde unos cuantos ASPX's estar llamando y enlazando los necesarios y en el momento que realmente se ocupe.

PROBLEMAS SUGERIDOS:

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- Construir programas (cs's) usando controles label, text, combobox, y otros para los problemas impares del del modelo de solución.



UNIDAD 1: ELEMENTOS BASICOS

APENDICE 1A: OBJETOS PROPIOS HTML

Button, Submit, Reset, CheckBox, Radio, Password, Text, TextArea, Select, Hidden

Controles Button, Submit y Reset.

| <i>Propiedades</i> | <i>Eventos</i> | <i>Metodos</i> |
|--------------------|----------------|----------------|
| form | OnClick | click |
| name | OnFocus | focus |
| value | | |
| enabled | | |

Control CheckBox.

| <i>Propiedades</i> | <i>Eventos</i> | <i>Metodos</i> |
|--------------------|----------------|----------------|
| form | OnClick | click |
| name | OnFocus | focus |
| value | | |
| enabled | | |
| checked | | |
| defaultchecked | | |

Control Radio.

| <i>Propiedades</i> | <i>Eventos</i> | <i>Metodos</i> |
|--------------------|----------------|----------------|
| form | OnClick | click |

| | | |
|---------|---------|-------|
| name | OnFocus | focus |
| value | | |
| enabled | | |
| checked | | |

Control Password.

| | | |
|--------------------|----------------|----------------|
| <i>Propiedades</i> | <i>Eventos</i> | <i>Metodos</i> |
| form | OnBlur | blur |
| name | OnFocus | focus |
| value | | |
| enabled | | |

Controles Text y Textareas.

| | | |
|--------------------|----------------|----------------|
| <i>Propiedades</i> | <i>Eventos</i> | <i>Metodos</i> |
| form | OnBlur | blur |
| name | OnFocus | focus |
| value | | |
| enabled | | |

Control Select.

| | | |
|--------------------|----------------|----------------|
| <i>Propiedades</i> | <i>Eventos</i> | <i>Metodos</i> |
| length | OnBlur | blur |
| options | OnFocus | focus |
| selectedIndex | OnChange | |

Control Hidden.

| | | |
|--------------------|----------------|----------------|
| <i>Propiedades</i> | <i>Eventos</i> | <i>Metodos</i> |
| name | | |
| value | | |



www.programacionfacil.com

LENGUAJE CSHARP

PALABRAS RESERVADAS

Keywords are predefined reserved identifiers that have special meanings to the compiler. They cannot be used as identifiers in your program unless they include @ as a prefix. For example, @if is a legal identifier but if is not because it is a keyword.

| | | | |
|----------|-----------|------------|-----------|
| abstract | event | new | struct |
| as | explicit | null | switch |
| base | extern | object | this |
| bool | false | operator | throw |
| break | finally | out | true |
| byte | fixed | override | try |
| case | float | params | typeof |
| catch | for | private | uint |
| char | foreach | protected | ulong |
| checked | goto | public | unchecked |
| class | if | readonly | unsafe |
| const | implicit | ref | ushort |
| continue | in | return | using |
| decimal | int | sbyte | virtual |
| default | interface | sealed | volatile |
| delegate | internal | short | void |
| do | is | sizeof | while |
| double | lock | stackalloc | |
| else | long | static | |
| enum | namespace | string | |

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

C# Operators

C# provides a large set of operators, which are symbols that specify which operations to perform in an expression. C# predefines the usual arithmetic and logical operators, as well as a variety of others as shown in the following table. In addition, many operators can be overloaded by the user, thus changing their meaning when applied to a user-defined type.

| Operator category | Operators |
|------------------------------------|-----------------------------------|
| Arithmetic | + - * / % |
| Logical (boolean and bitwise) | & ^ ! ~ && true false |
| String concatenation | + |
| Increment, decrement | ++ -- |
| Shift | << >> |
| Relational | == != < > <= >= |
| Assignment | = += -= *= /= %= &= = ^= <<= >>= |
| Member access | . |
| Indexing | [] |
| Cast | () |
| Conditional | ?: |
| Delegate concatenation and removal | + - |
| Object creation | new |
| Type information | is sizeof typeof |
| Overflow exception control | checked unchecked |
| Indirection and Address | * -> [] & |

Arithmetic Overflow

The arithmetic operators (+, -, *, /) can produce results that are outside the range of possible values for the numeric type involved. You should refer to the *C# Language Reference* section on a particular operator for details, but in general:

- Integer arithmetic overflow either throws an `OverflowException` or discards the most significant bits of the result (see below). Integer division by zero always throws a `DivideByZeroException`.
- Floating-point arithmetic overflow or division by zero never throws an exception, because floating-point types are based on IEEE 754 and so have provisions for representing infinity and NaN (Not a Number).
- Decimal arithmetic overflow always throws an `OverflowException`. **Decimal** division by zero always throws a `DivideByZeroException`.

When integer overflow occurs, what happens depends on the execution context, which can be checked or unchecked. In a checked context, an `OverflowException` is thrown. In an unchecked context, the most significant bits of the result are discarded and execution continues. Thus, C# gives you the choice of handling or ignoring overflow.

In addition to the arithmetic operators, integral-type to integral-type casts can cause overflow (for example, casting a long to an int) and are subject to checked or unchecked execution. Also note that bitwise operators and shift operators never cause overflow.

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

Types Reference Tables

The following reference tables summarize the C# types:

- Built-in types
- Integral types
- Floating-point types
- Default values
- Value types
- Implicit numeric conversions
- Explicit numeric conversions

For information on formatting the output of numeric types, see [Formatting Numeric Results Table](#).

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

Built-in Types Table

The following table shows the keywords for built-in C# types, which are aliases of predefined types in the **System** namespace.

| C# Type | .NET Framework type |
|---------|-----------------------|
| bool | System.Boolean |
| byte | System.Byte |
| sbyte | System.SByte |
| char | System.Char |
| decimal | System.Decimal |
| double | System.Double |
| float | System.Single |
| int | System.Int32 |
| uint | System.UInt32 |
| long | System.Int64 |
| ulong | System.UInt64 |

| | |
|---------------|----------------------|
| object | System.Object |
| short | System.Int16 |
| ushort | System.UInt16 |
| string | System.String |

Remarks

All of the types in the table, except **object** and **string**, are referred to as simple types.

The C# type keywords and their aliases are interchangeable. For example, you can declare an integer variable by using either of the following declarations:

```
int x = 123;
System.Int32 x = 123;
```

To display the actual type for any C# type use the system method GetType(). For example, the following statement displays the system alias that represents the type of myVariable:

```
Console.WriteLine(myVariable.GetType());
```

You can also use the typeof operator

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

Integral Types Table

The following table shows the sizes and ranges of the integral types, which constitute a subset of simple types.

| Type | Range | Size |
|--------|---------------------------------------------------------|--------------------------|
| sbyte | -128 to 127 | Signed 8-bit integer |
| byte | 0 to 255 | Unsigned 8-bit integer |
| char | U+0000 to U+ffff | Unicode 16-bit character |
| short | -32,768 to 32,767 | Signed 16-bit integer |
| ushort | 0 to 65,535 | Unsigned 16-bit integer |
| int | -2,147,483,648 to 2,147,483,647 | Signed 32-bit integer |
| uint | 0 to 4,294,967,295 | Unsigned 32-bit integer |
| long | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 | Signed 64-bit integer |
| ulong | 0 to 18,446,744,073,709,551,615 | Unsigned 64-bit integer |

Remarks

If the value represented by an integer literal exceeds the range of **ulong**, a compilation error will occur.

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

Floating-Point Types Table

The following table shows the precision and approximate ranges for the floating-point types.

| Type | Approximate range | Precision |
|--------|---------------------------------------------------------|--------------|
| float | $\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$ | 7 digits |
| double | $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ | 15-16 digits |

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

Default Values Table

The following table shows the default values of value types returned by the default constructors. Default constructors are invoked by using the **new** operator, for example:

```
int myInt = new int();
```

The preceding statement has the same effect as the following statement:

```
int myInt = 0;
```

Remember that using uninitialized variables in C# is not allowed.

| Value type | Default value |
|------------|----------------------------------------------------------------------------|
| bool | false |
| byte | 0 |
| char | '\0' |
| decimal | 0.0M |
| double | 0.0D |
| enum | The value produced by the expression (E)0, where E is the enum identifier. |
| float | 0.0F |
| int | 0 |
| long | 0L |

| | |
|--------|----------------------------------------------------------------------------------------------------------------------------|
| sbyte | 0 |
| short | 0 |
| struct | The value produced by setting all value-type fields to their default values and all reference-type fields to null . |
| uint | 0 |
| ulong | 0 |
| ushort | 0 |

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

Value Types Table

The following table lists the C# value types by category.

| Value type | Category |
|------------|-----------------------------|
| bool | Boolean |
| byte | Unsigned, numeric, integral |
| char | Unsigned, numeric, integral |
| decimal | Numeric, decimal |
| double | Numeric, floating-point |
| enum | Enumeration |
| float | Numeric, floating-point |
| int | Signed, numeric, integral |
| long | Signed, numeric, integral |
| sbyte | Signed, numeric, integral |
| short | Signed, numeric, integral |
| struct | User-defined structure |
| uint | Unsigned, numeric, integral |
| ulong | Unsigned, numeric, integral |
| ushort | Unsigned, numeric, integral |

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

Implicit Numeric Conversions Table

The following table shows the predefined implicit numeric conversions. Implicit conversions might occur in many situations, including method invoking and assignment statements.

| From | To |
|-------|---------------------------------------------|
| sbyte | short, int, long, float, double, or decimal |

| | |
|--------|------------------------------------------------------------------|
| byte | short, ushort, int, uint, long, ulong, float, double, or decimal |
| short | int, long, float, double, or decimal |
| ushort | int, uint, long, ulong, float, double, or decimal |
| int | long, float, double, or decimal |
| uint | long, ulong, float, double, or decimal |
| long | float, double, or decimal |
| char | ushort, int, uint, long, ulong, float, double, or decimal |
| float | double |
| ulong | float, double, or decimal |

Remarks

- The conversions from **int**, **uint**, or **long** to **float** and from **long** to **double** may cause a loss of precision, but not a loss of magnitude.
- There are no implicit conversions to the **char** type.
- There are no implicit conversions between floating-point types and the **decimal** type.
- A constant expression of type **int** can be converted to **sbyte**, **byte**, **short**, **ushort**, **uint**, or **ulong**, provided the value of the constant expression is within the range of the destination type.

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

Explicit Numeric Conversions Table

Explicit numeric conversion is used to convert any numeric type to any other numeric type, for which there is no implicit conversion, by using a cast expression. The following table shows these conversions.

| From | To |
|---------|-----------------------------------------------------------------------------|
| sbyte | byte, ushort, uint, ulong, or char |
| byte | sbyte or char |
| short | sbyte, byte, ushort, uint, ulong, or char |
| ushort | sbyte, byte, short, or char |
| int | sbyte, byte, short, ushort, uint, ulong, or char |
| uint | sbyte, byte, short, ushort, int, or char |
| long | sbyte, byte, short, ushort, int, uint, ulong, or char |
| ulong | sbyte, byte, short, ushort, int, uint, long, or char |
| char | sbyte, byte, or short |
| float | sbyte, byte, short, ushort, int, uint, long, ulong, char, or decimal |
| double | sbyte, byte, short, ushort, int, uint, long, ulong, char, float, or decimal |
| decimal | sbyte, byte, short, ushort, int, uint, long, ulong, char, float, or double |

Remarks

- The explicit numeric conversion may cause loss of precision or result in throwing exceptions.
- When you convert a **float**, **double**, or **decimal** value to an integral type, this value is rounded towards zero to the nearest integral value. If the resulting integral value is outside the range of the destination type, an **InvalidCastException** is thrown.
- When you convert **double** to **float**, the **double** value is rounded to the nearest **float** value. If the **double** value is too small or too large to fit into the destination type, the result will be zero or infinity.
- When you convert **float** or **double** to **decimal**, the source value is converted to **decimal** representation and rounded to the nearest number after the 28th decimal place if required. Depending on the value of the source value, one of the following results may occur:
 - If the source value is too small to be represented as a **decimal**, the result becomes zero.
 - If the source value is NaN (not a number), infinity, or too large to be represented as a **decimal**, an **InvalidCastException** is thrown.
- When you convert **decimal** to **float** or **double**, the **decimal** value is rounded to the nearest **double** or **float** value.

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

Formatting Numeric Results Table

You can format numeric results by using the **String.Format** method, or through the **Console.Write** method, which calls **String.Format**. The format is specified using format strings. The following table contains the supported standard format strings. The format string takes the form *Axx*, where *A* is the *format specifier* and *xx* is the *precision specifier*. The format specifier controls the type of formatting applied to the numerical value, and the precision specifier controls the number of significant digits or decimal places of the formatted output.

For more information on standard and custom formatting strings, see [Formatting Overview](#). For more information on the **String.Format** method, see [String.Format Method](#).

| Character | Description | Examples | Output |
|-----------|-------------|----------------------------------|---------------|
| C or c | Currency | Console.Write("{0:C}", 2.5); | \$2.50 |
| | | Console.Write("{0:C}", -2.5); | (\$2.50) |
| D or d | Decimal | Console.Write("{0:D5}", 25); | 00025 |
| E or e | Scientific | Console.Write("{0:E}", 250000); | 2.500000E+005 |
| F or f | Fixed-point | Console.Write("{0:F2}", 25); | 25.00 |
| | | Console.Write("{0:F0}", 25); | 25 |
| G or g | General | Console.Write("{0:G}", 2.5); | 2.5 |
| N or n | Number | Console.Write("{0:N}", 2500000); | 2,500,000.00 |
| | | Console.Write("{0:X}", 250); | FA |
| X or x | Hexadecimal | | |
| | | Console.Write("{0:X}", 0xffff); | FFFF |

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

WEBCONTROLS

WWW.PROGRAMACIONFACIL.COM

System.Web.UI.WebControls Namespace

See Also

.NET Framework Class Library

The System.Web.UI.WebControls namespace is a collection of classes that allow you to create Web server controls on a Web page. Web server controls run on the server and include form controls such as buttons and text boxes. They also include special purpose controls such as a calendar. Because Web server controls run on the server, you can programmatically control these elements. Web server controls are more abstract than HTML server controls. Their object model does not necessarily reflect HTML syntax.

Namespace hierarchy

Classes

| Class | Description |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AdCreatedEventArgs | Provides data for the AdCreated event of the AdRotator control. This class cannot be inherited. |
| AdRotator | Displays an advertisement banner on a Web page. |
| BaseCompareValidator | Serves as the abstract base class for validation controls that perform typed comparisons. |
| BaseDataList | Serves as the abstract base class for data listing controls, such as the DataList and DataGrid. This class provides the methods and properties common to all data listing controls. |
| BaseValidator | Serves as the abstract base class for validation controls. |
| BoundColumn | A column type for the DataGrid control that is bound to a field in a data source. |
| Button | Displays a push button control on the Web page. |

ButtonColumn

A column type for the DataGrid control that contains a user-defined command button, such as **Add** or **Remove**, that corresponds with each row in the column.

Calendar

Displays a single month calendar that allows the user to select dates and move to the next or previous month.

CalendarDay

Represents a date in the Calendar control.

CheckBox

Displays a check box that allows the user to select a **true** or **false** condition.

CheckBoxList

Creates a multi selection check box group that can be dynamically created by binding the control to a data source.

CommandEventArgs

Provides data for the **Command** event.

CompareValidator

Compares the value entered by the user into an input control with the value entered into another input control or a constant value.

CustomValidator

Performs user-defined validation on an input control.

DataGrid

A data bound list control that displays the items from data source in a table. The **DataGrid** control allows you to select, sort, and edit these items.

DataGridColumn

Serves as the base class for the different column types of the DataGrid control.

DataGridColumnCollection

A collection of DataGridColumn derived column objects that represent the columns in a DataGrid control. This class cannot be inherited.

DataGridCommandEventArgs

Provides data for the CancelCommand, DeleteCommand, EditCommand, ItemCommand, and UpdateCommand events of the DataGrid control. This class cannot be inherited.

DataGridItem

Represents an item (row) in the DataGrid control.

DataGridItemCollection

Represents a collection of DataGridItem objects in a DataGrid control.

DataGridItemEventArgs

Provides data for the ItemCreated and ItemDataBound events of the DataGrid control. This class cannot be inherited.

DataGridPageChangedEventArgs

Provides data for the PageIndexChanged event of the DataGrid control. This class cannot be inherited.

DataGridPagerStyle

Specifies the style for the pager of the DataGrid control. This class cannot be inherited.

DataGridSortCommandEventArgs

Provides data for the SortCommand event of the DataGrid control. This class cannot be inherited.

DataKeyCollection

Represents a collection that contains the key field of each record in a data source. This class cannot be inherited.

| | |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DataList | A data bound list control that displays items using templates. |
| DataSourceID | Provides data for the CancelCommand, DeleteCommand, EditCommand, ItemCommand, and UpdateCommand events of the DataList control. This class cannot be inherited. |
| DataListItem | Represents an item in the DataList control. |
| DataListItemCollection | Represents the collection of DataListItem objects in the DataList control. This class cannot be inherited. |
| DataListItemEventArgs | Provides data for the ItemCreated and ItemDataBound events of a DataList control. This class cannot be inherited. |
| DayRenderEventArgs | Provides data for the DayRender event of the Calendar control. This class cannot be inherited. |
| DropDownList | Represents a control that allows the user to select a single item from a drop-down list. |
| EditCommandColumn | A special column type for the DataGrid control that contains the Edit command buttons for editing data items in each row. |
| FontInfo | Encapsulates the font properties of text. This class cannot be inherited. |
| FontNamesConverter | Converts a string containing a list of font names to an array of strings containing the individual names. It also performs the reverse function. |
| FontUnitConverter | Converts a FontUnit to an object with another data type. It also converts an object with another data type to a FontUnit . |
| HyperLink | A control that displays a link to another Web page. |
| HyperLinkColumn | A column type for the DataGrid control that contains a hyperlink for each item in the column. |
| HyperLinkControlBuilder | Interacts with the parser to build a HyperLink control. |
| Image | Displays an image on a Web page. |
| ImageButton | A control that displays an image and responds to mouse clicks on the image. |
| Label | Represents a label control, which displays text on a Web page. |
| LabelControlBuilder | Interacts with the parser to build a Label control. |
| LinkButton | Displays a hyperlink style button control on a Web page. |
| LinkButtonControlBuilder | Interacts with the parser to build a LinkButton control. |
| ListBox | Represents a list box control that allows single or multiple item selection. |

| | |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------|
| ListControl | Serves as the abstract base class that defines the properties, methods, and events common for all list-type controls. |
| ListItem | Represents a data item in a data-bound list control. This class cannot be inherited. |
| ListItemCollection | A collection of ListItem objects in a list control. This class cannot be inherited. |
| ListItemControlBuilder | Interacts with the parser to build a ListItem control. |
| Literal | Reserves a location on the Web page to display static text. |
| LiteralControlBuilder | Interacts with the parser to build a Literal control. |
| MonthChangedEventArgs | Provides data for the VisibleMonthChanged event of a Calendar. This class cannot be inherited. |
| PagedDataSource | Encapsulates the properties of the DataGrid control that allow it to perform paging. This class cannot be inherited. |
| Panel | Represents a control that acts as a container for other controls. |
| Placeholder | A container to store dynamically added server controls on the Web page. |
| PlaceholderControlBuilder | Interacts with the parser to build a Placeholder control. |
| RadioButton | Represents a radio button control. |
| RadioButtonList | Represents a list control that encapsulates a group of radio button controls. |
| RangeValidator | Checks whether the value of an input control is within a specified range of values. |
| RegularExpressionValidator | Validates whether the value of an associated input control matches the pattern specified by a regular expression. |
| Repeater | A data-bound list control that allows custom layout by repeating a specified template for each item displayed in the list. |
| RepeaterCommandEventArgs | Provides data for the ItemCommand event of a Repeater. This class cannot be inherited. |
| RepeaterItem | Represents an item in the Repeater control. |
| RepeaterItemCollection | Represents a collection of RepeaterItem objects in the Repeater control. This class cannot be inherited. |
| RepeaterItemEventArgs | Provides data for the ItemCreated and ItemDataBound events of a Repeater. |
| RepeatInfo | Encapsulates the information used to render a list control that repeats a list of items. This class cannot be inherited. |

| | |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RequiredFieldValidator | Makes the associated input control a required field. |
| SelectedDatesCollection | Encapsulates a collection of System.DateTime objects that represent the selected dates in a Calendar control. This class cannot be inherited. |
| ServerValidateEventArgs | Provides data for the ServerValidate event of the CustomValidator control. This class cannot be inherited. |
| Style | Represents the style of a Web server control. |
| Table | Constructs a table and defines its properties. |
| TableCell | Represents a cell in a Table control. |
| TableCellCollection | Encapsulates a collection of TableCell and TableCell objects that make up a row in a Table control. This class cannot be inherited. |
| TableCellControlBuilder | Interacts with the parser to build a TableCell control. |
| TableHeaderCell | Represents a heading cell within a Table control. |
| TableItemStyle | Represents the style properties for an element of a control that renders as a TableRow or TableCell. |
| TableRow | Represents a row in a Table control. |
| TableRowCollection | Encapsulates a collection of TableRow objects that represent a single row in a Table control. This class cannot be inherited. |
| TableStyle | Represents the style for a table control. This class is primarily used by control developers. |
| TargetConverter | Converts a value representing the location (target) to display the content resulting from a Web navigation to a string. It also converts a string to a target value. |
| TemplateColumn | Represents a column type for the DataGrid control that allows you to customize the layout of controls in the column. |
| TextBox | Constructs a text box and defines its properties. |
| TextBoxControlBuilder | Interacts with the parser to build a TextBox control. |
| UnitConverter | Converts a Unit to an object of another data type. It also converts an object of another data type to a Unit . |
| ValidatedControlConverter | Converts a control on the Web Forms page that can be validated with a validation control to a string. |
| ValidationSummary | Displays a summary of all validation errors inline on a Web page, in a message box, or both. |
| WebColorConverter | Converts a predefined color name or an RGB color value to and from a System.Drawing.Color. |
| WebControl | Serves as the base class that defines the methods, properties and events common to all controls in the System.Web.UI.WebControls namespace. |

Xml

Displays an XML document without formatting or using Extensible Stylesheet Language Transformations (XSLT).

Interfaces

| Interface | Description |
|------------------|---------------------------------------------------------------------------------------------------------------|
| IRRepeatInfoUser | Defines the properties and methods that must be implemented by any list control that repeats a list of items. |

Structures

| Structure | Description |
|-----------|----------------------------------|
| FontUnit | Represents the size of a font. |
| Unit | Represents a length measurement. |

Delegates

| Delegate | Description |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| AdCreatedEventHandler | Represents the method that will handle the AdCreated event of an AdRotator control. |
| CommandEventHandler | Represents the method that will handle the Command event. |
| DataGridCommandEventHandler | Represents the method that will handle the CancelCommand, DeleteCommand, EditCommand, ItemCommand, and UpdateCommand events of a DataGrid. |
| DataGridItemEventHandler | Represents the method that will handle the ItemCreated and ItemDataBound events of a DataGrid. |
| DataGridPageChangedEventHandler | Represents the method that will handle the PageIndexChanged event of the DataGrid control. |
| DataGridSortCommandEventHandler | Represents the method that will handle the SortCommand event of the DataGrid control. |
| DataListCommandEventHandler | Represents the method that will handle the CancelCommand, DeleteCommand, EditCommand, ItemCommand, and UpdateCommand events of a DataList control. |
| DataListItemEventHandler | Represents the method that will handle the ItemCreated and ItemDataBound events of the DataList control. |
| DayRenderEventHandler | Represents the method that will handle the DayRender event of the Calendar control. |

| | |
|-----------------------------|------------------------------------------------------------------------------------------------|
| MonthChangedEventHandler | Represents the method that handles the VisibleMonthChanged event of a Calendar. |
| RepeaterCommandEventHandler | Represents the method that will handle the ItemCommand event of a Repeater. |
| RepeaterItemEventHandler | Represents the method that will handle the ItemCreated and ItemDataBound events of a Repeater. |
| ServerValidateEventHandler | Represents the method that will handle the ServerValidate event of a CustomValidator control. |

Enumerations

| Enumeration | Description |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------|
| BorderStyle | Specifies the border style of a control. |
| ButtonColumnType | Specifies the button type for the ButtonColumn object. |
| CalendarSelectionMode | Specifies the date selection mode of the Calendar control. |
| DayNameFormat | Specifies the display format for the days of the week on a Calendar control. |
| FirstDayOfWeek | Specifies the day to display as the first day of the week on the Calendar control. |
| FontSize | Specifies the font sizes defined by HTML 4.0. |
| GridLines | Specifies the grid line styles for controls displaying items in a table structure. |
| HorizontalAlign | Specifies the horizontal alignment of items within a container. |
| ImageAlign | Specifies the alignment of an image in relation to the text of a Web page. |
| ListItemType | Specifies the type of an item in a list control. |
| ListSelectionMode | Specifies the selection mode of the ListBox control. |
| NextPrevFormat | Represents the display format for the previous and next month navigation controls within the Calendar. |
| PagerMode | Represents the mode of the pager for accessing various pages within the DataGrid control. |
| PagerPosition | Specifies the position of the pager for accessing various pages within the DataGrid control. |
| RepeatDirection | Specifies the direction in which items of a list control are displayed. |
| RepeatLayout | Specifies the layout of items in a list control. |
| TextAlign | Specifies whether the text associated with a check box or radio button control appears to the left or to the right of the control. |
| TextBoxMode | Specifies the behavior mode of the text box. |

| | |
|------------------------------|-----------------------------------------------------------------------------------------------|
| TitleFormat | Specifies the title format for the displayed month in the Calendar control. |
| UnitType | Specifies the unit of measurement. |
| ValidationCompareOperator | Specifies the validation comparison operators used by the CompareValidator control. |
| ValidationDataType | Specifies the validation data types used by the CompareValidator and RangeValidator controls. |
| ValidationSummaryDisplayMode | Specifies the validation summary display mode used by the ValidationSummary control. |
| ValidatorDisplay | Specifies the display behavior of error messages in validation controls. |
| VerticalAlign | Specifies the vertical alignment of an object or text in a control. |

See Also

.NET Framework Class Library

FUENTE: © 2001 Microsoft Corporation. All rights reserved.

Como se puede apreciar existen muchos nuevos objetos de mucha utilidad para la construccion de sistemas de información, todos estos objetos **tienen sus propiedades especificas**, sin embargo muchos de ellos comparten las siguientes propiedades:

| PROPIEDAD | DESCRIPCION |
|-------------|----------------------------------------|
| BackColor | Carga o lee el color de background |
| BorderColor | Carga o lee el color del marco(border) |
| BorderStyle | Carga o lee el estilo del marco |
| BorderWidth | Carga o lee el tamaño del marco |
| Font | Carga o lee el font |
| ForeColor | Carga o lee el color del foreground |
| Height | Carga o lee la altura del control |

| | |
|----------|------------------------------------------|
| ID | Carga o lee el identificador del control |
| TabIndex | Carga o lee el tab index |
| ToolTip | Carga o lee el tooltip del control |
| Visible | Carga o lee su estado visible |
| Width | Carga o lee la anchura del control |

WWW.PROGRAMACIONFACIL.COM

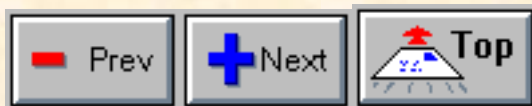
UNIDAD 2: INSTRUCCIONES DE CONTROL DE PROGRAMA

TEMA 1: INTRODUCCIÓN

En este capitulo se continua siguiendo el esquema de trabajo ya planteado en el capitulo anterior, es decir:

Construccion de programas basandonos en los dos modelos de red vistos es decir aspx's(cuando es una forma junto con su programa) y .cs (cuando estan separados las forma y el programa).

Esta definicion asp'x y .cs no es real, es solo para ir practicando los dos tipos de programas a lo largo del curso



WWW.PROGRAMACIONFACIL.COM

UNIDAD 2: INSTRUCCIONES DE CONTROL DE PROGRAMA

TEMA 2: INSTRUCCIONES DE CONTROL DE PROGRAMA

Instrucciones de control de programa permiten alterar la secuencia normal de ejecución de un programa.

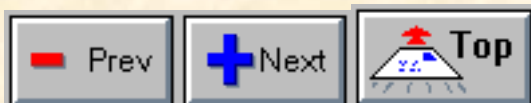
Estas instrucciones se dividen en tres grandes categorías:

1.- Instrucciones Condicionales que en Csharp se implementan con las instrucciones if y switch.

b) Instrucciones de ciclos con

- for
- while
- do...while
- foreach: ciclo especializado en procesar y manipular arreglos y colecciones, por tanto se vera hasta la siguiente unidad.

Muchas de ellas con sus correspondientes componentes visuales, tanto en html como en activex, htmlcontrols y webcontrols, pero para proposito del curso solo se usaran los WebControls



WWW.PROGRAMACIONFACIL.COM

UNIDAD 2: INSTRUCCIONES DE CONTROL DE PROGRAMA

TEMA 3: INSTRUCCIONES CONDICIONALES

Una de las mas poderosas características de cualquier computador es la capacidad que tiene de tomar decisiones.

Es decir al comparar dos alternativas diferentes el computador puede tomar una decisión basándose en la evaluación que hace de alguna condición.

ejemplo de instrucciones condicionales:

```
si sueldo > 3000
```

```
desplegar rico
```

```
si no
```

```
desplegar pobre
```

```
Fin-si
```

```
si sexo = 'm'
```

```
imprime mujer
```

```
si no
```

imprime hombre

Fin-si

De los ejemplos observar que los caminos a seguir por el computador dependerán de la evaluación que el computador hace con y de la condición.

Todo lenguaje de programación debe tener instrucciones que permitan formar condiciones e instrucciones que pueden evaluar esas condiciones.

Pero recordar que lenguajes modernos y orientados a clientes-servidores de igual forma tienen componentes que permiten del mismo modo al usuario tomar decisiones incluso directamente en pantalla, es decir también existen los objetos, controles o componentes de selección y decisión en html, htmlcontrols, activex, webcontrols.

El formato general de una instrucción condicional es:



Como se observa, son cuatro partes bien diferenciadas entre si;

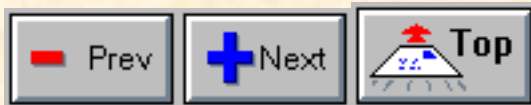
- La propia instrucción condicional en si
- La condición
- El grupo cierto de instrucciones
- El grupo falso de instrucciones

Cuando el computador evalúa una condición el resultado de esa evaluación solo es evaluado de dos maneras o la condición es CIERTA o la condición es FALSA.

Esto dependerá del valor que tenga asignado o que se haya capturado para la variable que esta en la condición, por ejemplo si se capturo 6000 en sueldo en el ejemplo a) entonces el computador indicaría que la condición es CIERTA pero en otro caso si a la variable sueldo primero se le asigno un valor de 250 entonces el computador indicaría que la condición es FALSA.

Ya dependiendo del resultado de la evaluación, el computador ejecuta las instrucciones contenidas en el grupo de cierto o falso respectivamente.

Empezaremos el análisis por la CONDICIÓN.

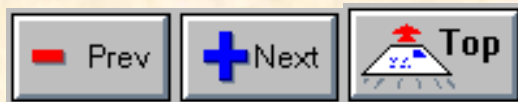


UNIDAD 2: INSTRUCCIONES DE CONTROL DE PROGRAMA**TEMA 4: CONDICIONES SIMPLES**

En general todas las condiciones se forman con;

| Variables | Operadores Relacionales | Constante o Variables |
|-----------|----------------------------|-----------------------|
| sexo | = | m |
| sueldo | > | 300,000 |

Una condición simple se define como el conjunto de variables y/o constantes unidas por los llamados operadores relacionales.



UNIDAD 2: INSTRUCCIONES DE CONTROL DE PROGRAMA**TEMA 5: OPERADORES RELACIONALES**

Los operadores relacionales que reconoce Csharp son:

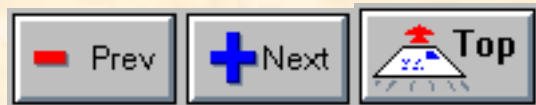
| Operador | Significado |
|----------|------------------------------------|
| == | Igual que |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual que |
| <= | Menor o igual que |
| != | No es igual que o es diferente que |

Observar y tener cuidado sobre todo con el operador de igualdad(=) y el operador relacional de comparación por igualdad(==) es decir:

`sueldo = 500`, Se esta pidiendo cargar o asignar la variable `sueldo` con el valor 500

`sueldo == 500`, Se esta pidiendo que se compare el valor o dato que se encuentra en la variable `sueldo`, contra el numero 500.

Solo este ultimo formato es valido dentro de una condición en una instrucción condicional.



WWW.PROGRAMACIONFACIL.COM

UNIDAD 2: INSTRUCCIONES DE CONTROL DE PROGRAMA

TEMA 6: INSTRUCCIÓN IF

Es la instrucción condicional mas usada en los diversos lenguajes de programación, su formato completo y de trabajo en Csharp es:

cargar o asignar la variable de condición;

```
if (condición)
```

```
{ grupo cierto de instrucciones;}
```

```
else
```

```
{ grupo falso de instrucciones; };
```

Primus.- Observar **donde van y donde no van** los puntos y comas;

Secundus.- La condición va entre paréntesis ;

Tertius.- Si un if no ocupa un grupo falso de instrucciones entonces no se pone el else y la llave antes del else si terminaría con punto y coma.

Ejemplos:

a) primer modelo, es decir un aspx con código csharp

Prog4.aspx

```
<HTML>
```

```
<FORM RUNAT=SERVER>
```

```
SUELDO<ASP:TEXTBOX ID=SUELDO RUNAT=SERVER/><BR>
```

```
RESULTADO<ASP:LABEL ID=RESULTADO RUNAT=SERVER/><BR>
```

```
<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/><BR>
```

```
<ASP:LINKBUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/><BR>
```

```
<ASP:IMAGEBUTTON IMAGEURL="OSO.JPG" ONCLICK=EVENTO2 RUNAT=SERVER/><BR>
```

```
</FORM></HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
if ( Int32.Parse(SUELDO.Text) >= 3000)
```

```
{RESULTADO.Text="RICO";}
```

```
else
```

```
{RESULTADO.Text="POBRE";};
```

```
}
```

```
void EVENTO2 (Object sender, ImageClickEventArgs e)
```

```
{
```

```
if ( Int32.Parse(SUELDO.Text) >= 3000)
```

```
{RESULTADO.Text="RICO";}
```

```
else
```

```
{RESULTADO.Text="POBRE";};
```



```
}  
  
</script>
```

Para ejecutarlo subirlo a tu sitio en programaciónfacil y pedirlo desde ahí, con la dirección:

`http://programacionfacil.com:4080/tusitio/prog4.aspx`

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Corrida prog4.aspx



notas:

En cuanto al if se esta respetando el formato que ya se indico.

Lo nuevo es que existen tres tipos de objetos o controles de orden o comando, ellos son BUTTON, LINKBUTTON, IMAGEBUTTON(efectivamente si hacen click dentro de la imagen de mi perrito(osito) se activa el metodo correspondiente y por tanto se evalua la condición.

En cuanto a button y linkbutton el primero es la cajita normal de ordenes que se a venido usando y el segundo es una liga normal de html, observar que pueden usar el mismo ONCLICK y metodo void etc()).

Pero imagebutton debe llevar su propio onclick y en su metodo void no se manda como parametro un EVENTARGS sino un IMAGECLICKEVENTARGS.

b) ejemplo 2 segundo modelo forma en prog5.aspx y codigo en prog5.cs

prog5.aspx

```
<%@ PAGE INHERITS=PROG5 SRC=PROG5.CS %>
```

```
<HTML>
```

```
<FORM RUNAT=SERVER>
```

```
<ASP: PANEL ID=PANEL1 BACKCOLOR=GREEN RUNAT=SERVER>
```

```
SUELDO<ASP: TEXTBOX ID=SUELDO BACKCOLOR=YELLOW RUNAT=SERVER/><BR>
```

```
RESULTADO<ASP: LABEL ID=RESULTADO BORDERSTYLE=2 RUNAT=SERVER/><BR>
```

```
<ASP: BUTTON TEXT=OK ONCLICK=EVENTO1 TOOLTIP="CLICK HERE POR FAVOR"  
RUNAT=SERVER/>
```

```
</ASP: PANEL>
```

```
</FORM><BR>
```

```
SALUDOS Y DESPEDIDA
```

```
</HTML>
```

nota observar como agregar atributos a los objetos o controles de tipo webcontrol.

Prog5.cs

```
using System;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Drawing;

public class PROG5 : Page

{

    // CREANDO Y ENLAZANDO CONTROLES A FORM.ASPX

    protected TextBox SUELDO;

    protected Label RESULTADO;

    protected Panel PANEL1;

    // programando evento clic de prog5.aspx

    public void EVENTO1 (Object sender, EventArgs e)

    {

        if ( Int32.Parse(SUELDO.Text)>= 3000 )

        {

            RESULTADO.Text="RICO";

        }

        else

        {

            RESULTADO.Text="POBRE";

        }

    }

}
```

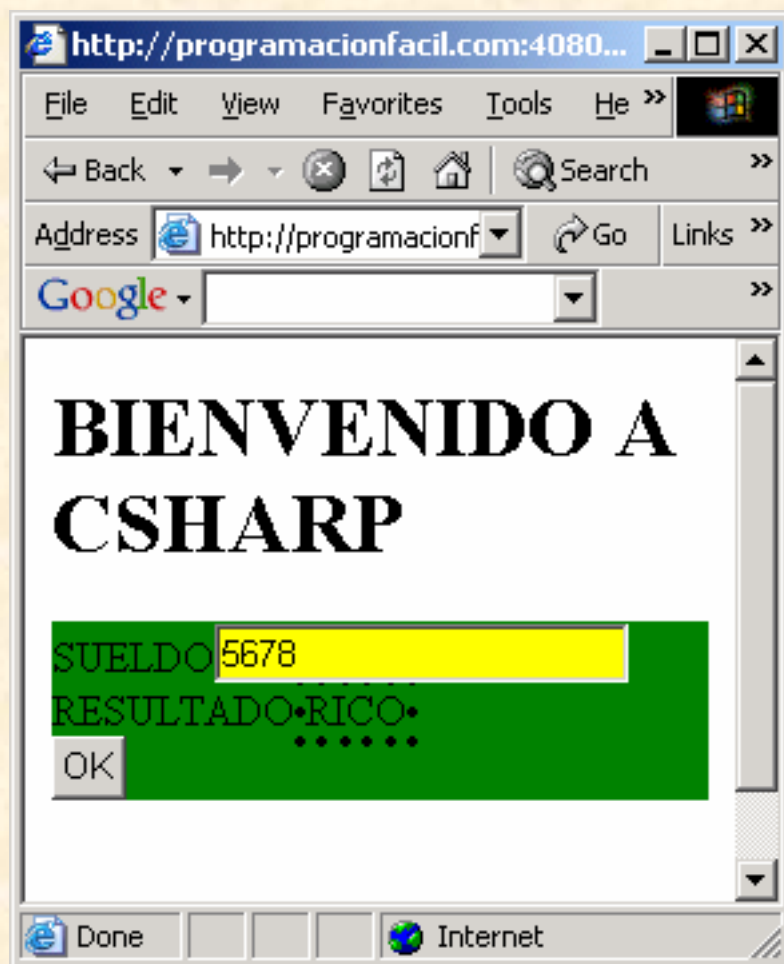


```
}}
```

subir ambos programas a tu sitio en programacionfacil y pedir desde el browser el aspx, es decir usar: `http://programacionfacil.com:4080/tusitio/prog5.aspx`

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

corrida:

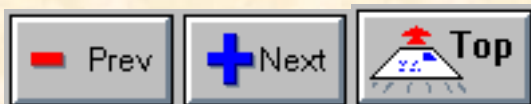


Recordar que es valido usar mas de una instrucción dentro del grupo cierto o falso del if.

PROBLEMAS SUGERIDOS

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

- 1.- Capturar un numero cualesquiera e informar si es o no es mayor de 100 (programar los dos modelos y usar button's diferentes)
- 2.- Capturar un numero entero cualesquiera e informar si es o no es múltiplo de 4 (recordar el operador mod(%), analizado en el tema de operadores aritméticos). (hacer los mismos dos modelos y usar button's diferentes)
- 3.- Capturar los cinco datos mas importantes de un Empleado, incluyendo el sueldo diario y los días trabajados desplegarle su cheque semanal solo si ganó mas de \$500.00 en la semana, en caso contrario desplegarle un bono de despensa semanal de \$150.00.(primer modelo button's diferentes)
- 4.- Capturar los datos mas importantes de un estudiante incluyendo tres calificaciones construir una boleta de calificaciones en una pagina de respuesta bien bonita si el estudiante es de la carrera de medicina, en caso contrario construir una pagina mas bonita todavia que despliega un oficio citando a los padres del estudiante a una platica amistosa con los maestros de la escuela. (segundo modelo)
- 5.- Capturar los datos mas importantes de un producto cualesquiera, incluyendo cantidad, precio, etc. desplegar una orden de compra, solo si el producto es de origen nacional, en caso contrario no hacer nada. (en el modelo que quieran)



WWW.PROGRAMACIONFACIL.COM**UNIDAD 2: INSTRUCCIONES DE CONTROL DE PROGRAMA****TEMA 7: CONDICIONES COMPUESTAS**

En muchas ocasiones es necesario presentar mas de una condición para su evaluación al computador.

Por ejemplo que el computador muestre la boleta de un alumno si este estudia la carrera de medicina y su promedio de calificaciones es mayor de 70.

Una condición compuesta se define como dos o mas condiciones simples unidas por los llamados operadores lógicos.

Los operadores lógicos que csharp reconoce son:

| OPERADOR | SIGNIFICADO |
|----------|-------------|
| && | Y LOGICO |
| | O LOGICO |
| ! | NEGACION |

Para que el computador evalúe como CIERTA una condición compuesta que contiene el operador lógico "&&", las dos condiciones simples deben ser ciertas.

Para que el computador evalúe como CIERTA una condición compuesta que contiene el operador lógico "||", basta con que una de las condiciones simples sea cierta.

La cantidad total de casos posibles cuando se unen dos o mas condiciones simples esta dada por la relación 2^n donde n = cantidad de condiciones, la primera mitad de ellos empieza en cierto y la segunda mitad en falso.

Ejemplo, si formamos una condición compuesta con dos condiciones simples y el operador lógico "y", la cantidad total de casos posibles serian $2^2 = 4$, y se puede construir la siguiente tabla de verdad.

Tabla de verdad con "y"

| IRA COND SIMPLE | 2DA COND SIMPLE | EVALUACION |
|-----------------|-----------------|------------|
| C | C | C |
| C | F | F |
| F | C | F |
| F | F | F |

La evaluación final, se obtiene usando la regla anteriormente descrita para una condición compuesta, que contiene el operador "Y".

Esta tabla significa lo siguiente;

1.- Cualquiera que sean la cantidad de datos procesados, siempre caerá en uno de estos cuatro casos generales.

La tabla de verdad para una condición compuesta con "Or" es la siguiente:

| IRA COND SIMPLE | 2DA COND SIMPLE | EVALUACION |
|-----------------|-----------------|------------|
| C | C | C |
| C | F | C |
| F | C | C |
| F | F | F |

Construir una tabla de verdad para una condición compuesta de tres o mas condiciones simples es también tarea sencilla, solo recordar que;

1.- La cantidad posible de casos es $2^3 = 8$ casos posibles, la mitad empiezan con Cierto y la otra mitad empiezan con Falso.

2.- Para evaluar esta condición triple primero se evalúan las dos primeras incluyendo su operador bajo las reglas ya descritas y luego se evalúa el resultado parcial contra la ultima condición y ultimo operador para obtener la evaluación final.

Ejemplo una condición compuesta de tres condiciones simples, donde el primer operador lógico es el "y" y el segundo operador lógico es el "o", daría la siguiente tabla de verdad.

| Ira cond | 2da cond | Eval 1a Y 2a | 3ra cond | Eval eval O 3ra |
|----------|----------|--------------|----------|-----------------|
| C | C | C | C | C |
| C | C | C | F | C |
| C | F | F | C | C |
| C | F | F | F | F |
| F | C | F | C | C |
| F | C | F | F | F |
| F | F | F | C | C |
| F | F | F | F | F |

PROBLEMAS SUGERIDOS

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- Construir un programa que capture un numero cualesquiera e informe si es o no es mayor de 50 y múltiplo de tres. (solo escribir el mensaje de respuesta de manera muy clara y esto resuelve el problema) (LOS DOS MODELOS ASPX Y CS)

2.- Construir un programa que indique si un numero es un par positivo.
(dos modelos)

3.- Capturar los datos de un producto incluyendo su cantidad en existencia, desplegar una orden de compra si la cantidad en existencia del producto es menor que el punto de reorden, o si el origen del producto es nacional. (ASPX)

4.- Construir un programa que capture los datos de un empleado, desplegar en una pagina su cheque semanal si gana mas de \$500.00 y si esta en el departamento de producción, en caso contrario desplegarle en otra pagina un bono de despensa del 25% de su sueldo semanal. (CS)



UNIDAD 2: INSTRUCCIONES DE CONTROL DE PROGRAMA

TEMA 8: INSTRUCCIÓN SWITCH

También existen ocasiones o programas donde se exige evaluar muchas condiciones a la vez, en estos casos o se usa una condición compuesta muy grande o se debe intentar convertir el problema a uno que se pueda resolver usando la instrucción SWITCH.

Esta instrucción es una instrucción de decisión múltiple donde el compilador prueba o busca el valor contenido en una variable ENTERA, CHARACTER, STRING contra una lista de constantes apropiadas, cuando el computador encuentra el valor de igualdad entre variable y constante entonces ejecuta el grupo de instrucciones asociados a dicha constante, si no encuentra el valor de igualdad entre variable y constante, entonces ejecuta un grupo de instrucciones asociados a un default, aunque este ultimo es opcional.

El formato de esta instrucción es el siguiente;

capturar o asignar variable de condición;

```
switch(var OPCION)
```

```
{
```

```
    case const1: instrucción(es);
```

```
    break;
```

```
    case const2: instrucción(es);
```

```
break;

case const3: instrucción(es);

break; .....

default: instrucción(es);break;

};
```

prog6.aspx

<HTML>

<FORM RUNAT=SERVER>

DAME UNA LETRA<ASP:TEXTBOX ID=LETRA RUNAT=SERVER/>

ANIMALITO<ASP:TEXTBOX ID=ANIMALITO BACKCOLOR=AZURE RUNAT=SERVER/>

<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/>

</FORM>

SALUDOS Y DESPEDIDA

</HTML>

<SCRIPT LANGUAGE=C# RUNAT=SERVER>

void EVENTO1 (Object sender, EventArgs e)

{

switch(LETRA.Text)

{ case "a":


```
ANIMALITO.Text="aguila";break;

case "b":case "B":

ANIMALITO.Text="baca";break;

case "c":

ANIMALITO.Text="caballo"; int alfa=5; break;

default:

ANIMALITO.Text="no hay";break;

}}

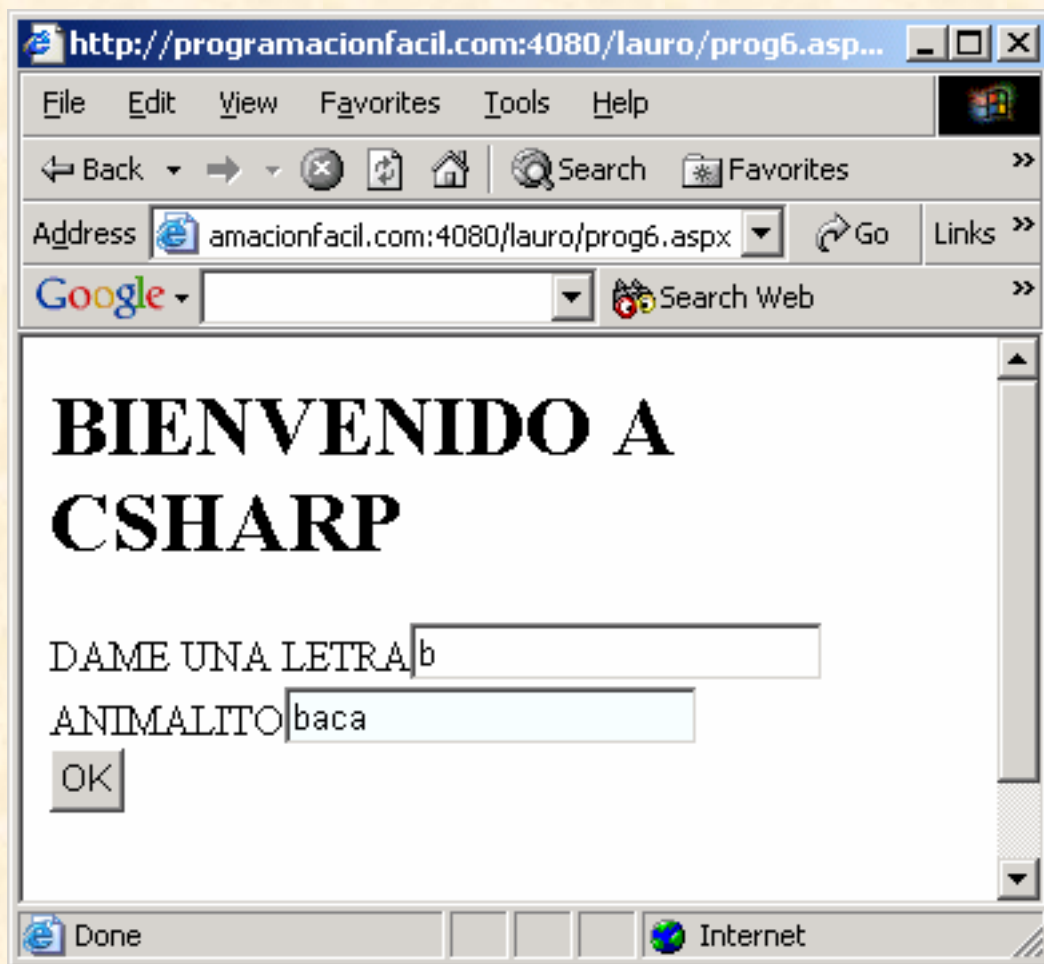
</script>
```

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Es el primer modelo aspx, el otro modelo CS les toca a ustedes y la unica nota digna de tomar en cuenta es que se pueden usar mas de dos instrucciones en cada case.

Observar el caso "b", observar como se pueden usar mas de dos case con un solo break, sorry por lo de BACA pero el unico animalito que me acorde fue el BURRO y luego mis alumnos se sienten aludidos y ofendidos.

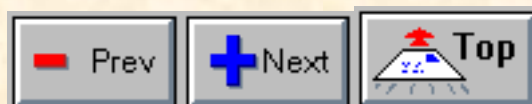
Corrida:



PROBLEMAS SUGERIDOS

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

- 1.- Construir un programa que capture un deporte cualesquiera y despliegue dos implementos deportivos apropiados. (aspx y cs)
- 2.- Evaluar cualquier función vista para cuando $x = 3, -4, 5^2$ (aspx y cs)



UNIDAD 2: CONTROL DE PROGRAMA

TEMA 9: Controles ListBox y DropDownList

Existen muchas ocasiones en donde el usuario del programa tiene que proporcionar datos que provienen de un conjunto finito y muy pequeño de posibles respuestas esto significa que cada vez que se ejecute el programa el usuario estará proporcionando las mismas respuestas.

Ejemplo de esta clase de datos, son por ejemplos Municipio en BC las posibles respuestas solo son (Tecate, Tijuana, Mexicali, Ensenada, Rosarito), otro ejemplo es Sexo (Hombre, Mujer), etc.

Para situaciones como esta, existen componentes webcontrols que permiten programar por adelantado las posibles respuestas y el usuario solo debe seleccionar la respuesta apropiada en lugar de tener que escribirla.

Estos controles nos permiten definir en primera instancia un conjunto de datos o valores o respuestas asociados a una caja de edición cualesquiera, así ahora el usuario tendrá la oportunidad de seleccionar un dato del conjunto de datos o respuestas ya predefinido.

Estos componentes DEBERAN CONSTRUIRSE EN dos partes una parte de encabezado para poner el nombre del grupo de respuestas(por ejemplo municipios, sexo, etc.)

La segunda parte es la lista de opciones o respuestas que se debe cargar al tiempo de ejecución de la forma aspx como lo muestra el siguiente programa:

Prog7.aspx

<HTML>

<FORM RUNAT=SERVER>

```

SEXO.....:<ASP:LISTBOX ID=SEXO ROWS=2 RUNAT=SERVER>

<ASP:LISTITEM TEXT=MASCULINO RUNAT=SERVER/>

<ASP:LISTITEM TEXT=FEMENINO RUNAT=SERVER/></ASP:LISTBOX> <BR>

MUNICIPIO:<ASP:DROPDOWNLIST ID=MUNICIPIO RUNAT=SERVER>

<ASP:LISTITEM TEXT=ENSENADA RUNAT=SERVER/>

<ASP:LISTITEM TEXT=MEXICALI RUNAT=SERVER/>

<ASP:LISTITEM TEXT=ROSARITO RUNAT=SERVER/>

<ASP:LISTITEM TEXT=TECATE RUNAT=SERVER/>

<ASP:LISTITEM TEXT=TIJUANA RUNAT=SERVER/></ASP:DROPDOWNLIST><BR>

SEXO.....:<ASP:LABEL ID=SEX RUNAT=SERVER/><BR>

MUNICIPIO...:<ASP:LABEL ID=MUNI RUNAT=SERVER/><BR>

<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/>

</FORM></HTML>

<SCRIPT LANGUAGE=C# RUNAT=SERVER>

void EVENTO1 (Object sender, EventArgs e)

{

// PROPIEDAD SELECTEDITEM queda cargado con el dato seleccionado y con TEXT SE ESTA

// MANDANDO Al TEXT de label

// OBSERVAR QUE SE ESTA USANDO UN COMPONENTE TEXT PARA DESPLEGAR MENSAJES

SEX.Text = SEXO.SelectedItem.Text;

MUNI.Text = MUNICIPIO.SelectedItem.Text;

}

```


</script>

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Observar que tanto en listbox como en dropdownlist se cargan sus elementos con LISTITEM.

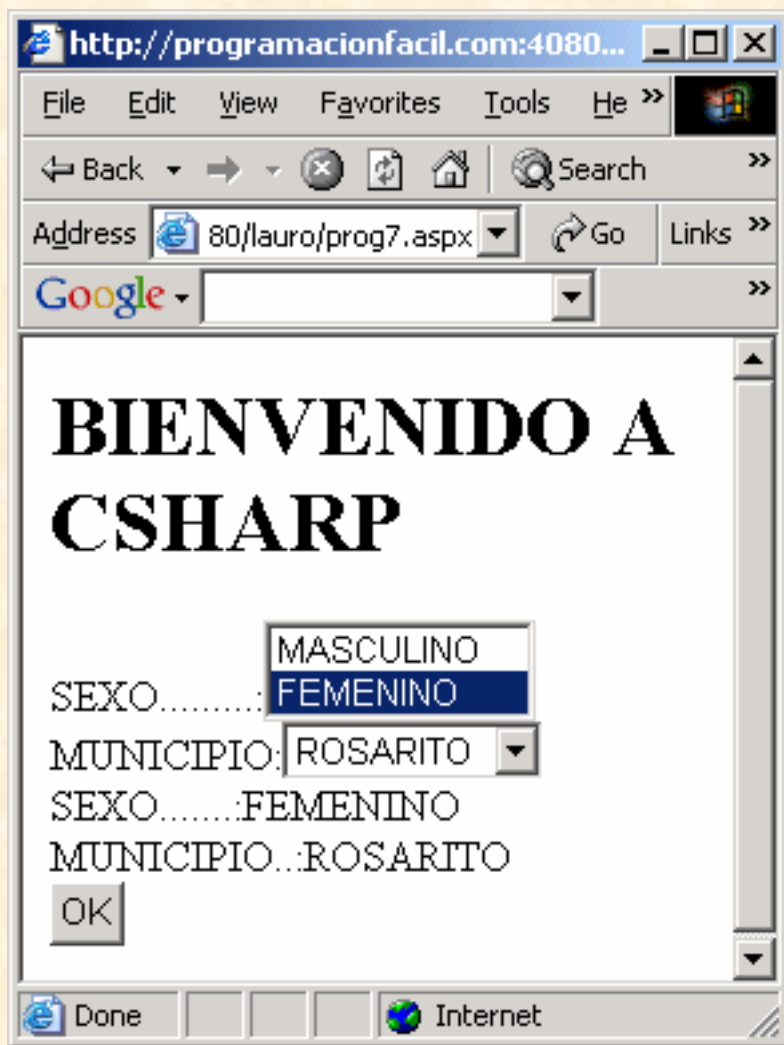
Ya en código se usa la propiedad SelectedItem, que esta apuntando o cargada con el valor o datos seleccionado por el usuario.

La diferencia en pantalla o ejecución entre ambos controles, se ve en la corrida, que esta unos parrafos mas abajo.

Recordar que estos controles tienen muchas propiedades muy utiles y que se seguiran usando a lo largo del curso.

Solo grabarlo como programa7.aspx y subirlo a tu sitio y ejecutarlo de manera normal <http://programacionfacil.com:4080/tusitio/prog7.aspx>

Corrida prog7.aspx



Problemas sugeridos:

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- 5 problemas de los ya vistos y deberan usar en unos listboxs y en otros dropdownlist's, tambien deberan construir (2 en aspx y 3 en cs)



WWW.PROGRAMACIONFACIL.COM

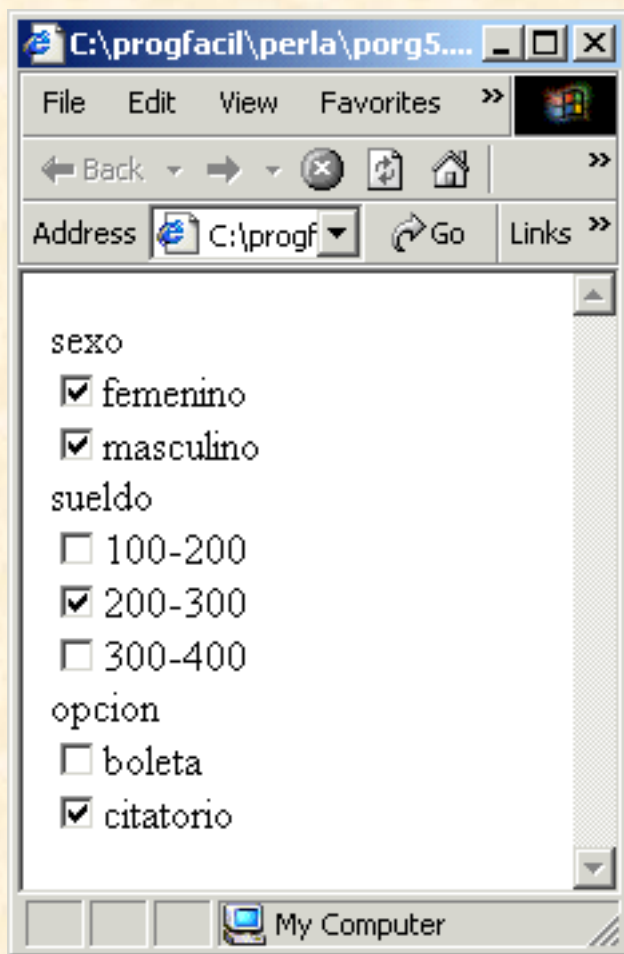
UNIDAD 2: CONTROL DE PROGRAMA

TEMA 10: CHECKBOX Y CHECKBOXLIST

Estos componentes CheckBox y CheckBoxList permiten seleccionar una opción al usuario del programa o tomar una decisión directamente en pantalla.

La diferencia entre ellos aparte de como se programa el componente, es que checkboxlist permite agrupar mejor sus elementos internos tal como se muestra en las corridas:

Ejemplos de uso:



Observar que dos o mas checkboxes pueden estar seleccionados a la vez.

CHECKBOX:

Codigo prog9.aspx:(recordar que ustedes hacen los prog.cs)

```
<HTML>
```

```
<FORM RUNAT=SERVER>
```

```
SEXO:<BR>
```

```
<ASP:CHECKBOX TEXT=MASCULINO ID=MASCULINO RUNAT=SERVER />
```

```
<ASP:CHECKBOX TEXT=FEMENINO ID=FEMENINO RUNAT=SERVER /><BR>
```

```
<ASP:CHECKBOX TEXT=0-10 ID=DIEZ RUNAT=SERVER />
```

```
<ASP:CHECKBOX TEXT=10-20 ID=VEINTE RUNAT=SERVER /><BR>
```



```
<ASP:BUTTON ONCLICK=EVENTO1 TEXT=OK RUNAT=SERVER /><BR>

<ASP:LABEL ID=SEXO RUNAT=SERVER /><BR>

<ASP:LABEL ID=EDAD RUNAT=SERVER /><BR>

</FORM></HTML>

<SCRIPT LANGUAGE=C# RUNAT=SERVER>

void EVENTO1(Object sender, EventArgs e)

{

if(MASCULINO.Checked) SEXO.Text="MASCULINO";

if(FEMENINO.Checked) SEXO.Text="FEMENINO";

if(DIEZ.Checked) EDAD.Text="DE CERO A DIEZ";

if(VEINTE.Checked) EDAD.Text="DE DIEZ A VEINTE";

}

</script>
```

1.- Grabarlo y subirlo como prog9.aspx a tusitio en programacionfacil.com

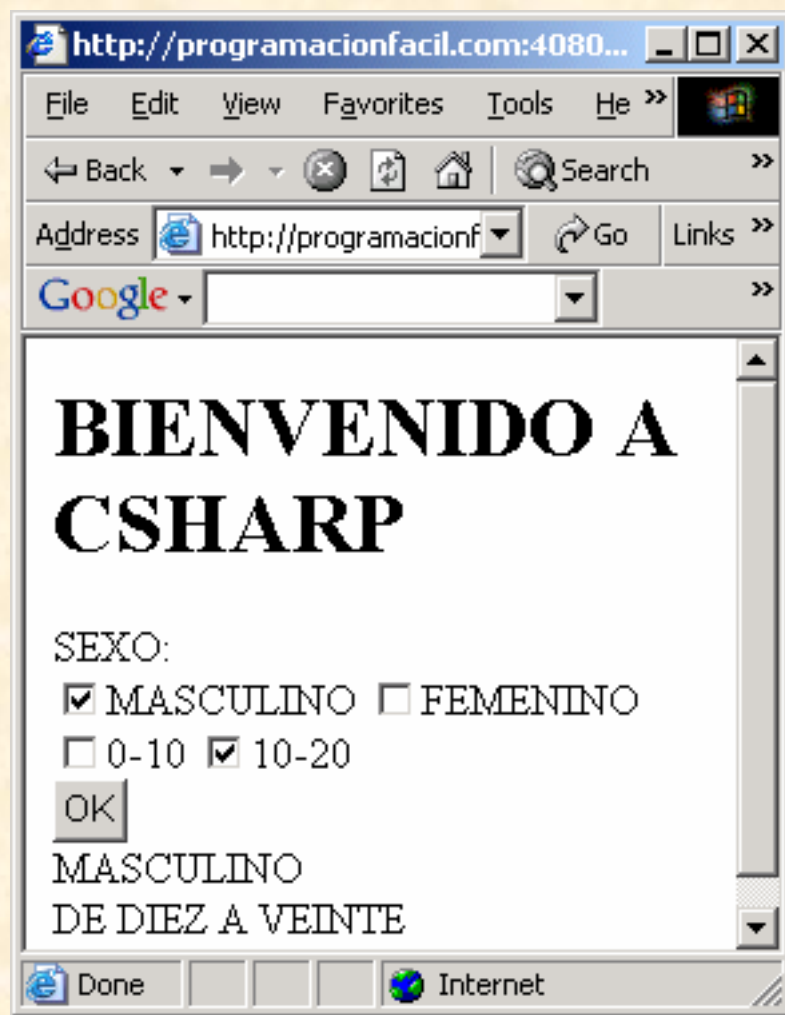
NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

2.- La propiedad ID debera ser diferente en cada checkbox usado tambien se puede agregar una propiedad checked=true para que aparezca ya palomeado o seleccionado el control.

3.- Cuando se activa prog9x.asp, esta forma manda el par NAME=ON solo

de los checkbox que fueron seleccionados.

Corrida prog9.aspx:



Para programar este componente:

Solo recordar usar la propiedad checked en codigo y un if por cada checkbox.

CHECKBOXLIST:

Este control nos permite mejorar la apariencia de la salida del checkbox, especialmente si usamos propiedades REPEATCOLUMNS y REPEATDIRECTIONS.

Prog10.aspx

```
<HTML>

<FORM RUNAT=SERVER>

SEXO:<BR>

<ASP:CHECKBOXLIST ID=SEXO RUNAT=SERVER>

<ASP:LISTITEM TEXT=MASCULINO RUNAT=SERVER />

<ASP:LISTITEM TEXT=FEMENINO RUNAT=SERVER />

<ASP:LISTITEM TEXT=NEUTRO RUNAT=SERVER />

</ASP:CHECKBOXLIST>

<ASP:BUTTON ONCLICK=EVENTO1 TEXT=OK RUNAT=SERVER /><BR>

<ASP:LABEL ID=SEX RUNAT=SERVER /><BR>

</FORM></HTML>

<SCRIPT LANGUAGE=C# RUNAT=SERVER>

void EVENTO1 (Object sender, EventArgs e)

{

// como es un control similar a listbox, tambien usa propiedad selecteditem

SEX.Text = SEXO.SelectedItem.Text;

}

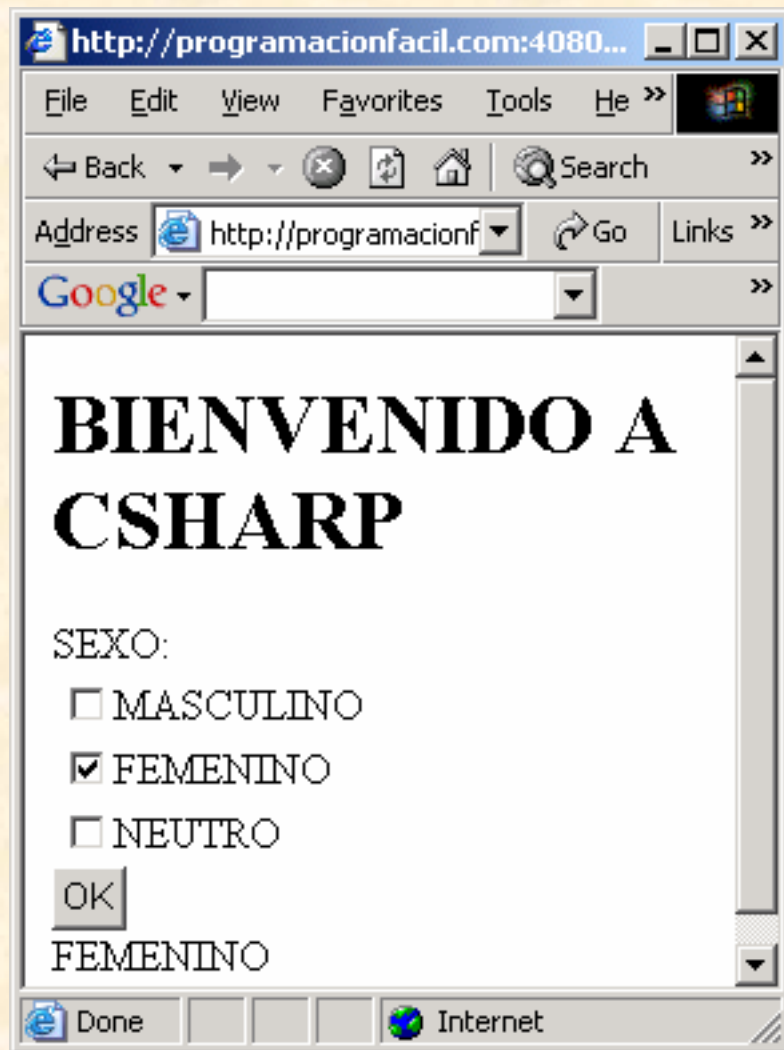
</script>
```

Solo agregar un ID al control y un listitem por cada elemento, para programarlo solo usar la propiedad selecteditem.

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una

cuenta e instrucciones de uso apropiadas.

Corrida:



PROBLEMAS SUGERIDOS

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- Evaluar la función $y = 3x^2 - 4x + 2$ para $x = 2, -5, 8$ (usar un CheckBox por cada valor de x , y programar cada if de cada CheckBox con la operación correspondiente y el despliegue del resultado)(dos modelos aspx y cs.)

2.- Construir un pagina.aspx con los datos de un automóvil y abajo construir un plan de financiamiento a dos años o muestra un plan de financiamiento a tres años. (son dos checkbox en la pagina.aspx mas un monton de botones de texto o labels, para pasar los datos al aspx y un botón de ok)(checkbox).

3.- Construir un prog.cs que evalúe una función cualquiera y que use además el checkboxlist.



[WWW.PROGRAMACIONFACIL.COM](http://www.programacionfacil.com)

UNIDAD 2: INSTRUCCIONES DE CONTROL DE PROGRAMA

TEMA 11: COMPONENTE RadioButton y RadioButtonList

Se utiliza para presentar al usuario un conjunto de opciones **mutuamente excluyentes entre si**, es decir, si el usuario selecciona un componente radio todos los demás componentes radioButton en la forma se desmarcan o deseleccionan solos, es por esta razón que decimos que radiobotones son mutuamente excluyentes.

RADIOBUTTON:

Codigo prog11.aspx

```
<HTML>

<FORM RUNAT=SERVER>

SEXO:<BR>

<ASP:RADIOBUTTON TEXT=MASCULINO ID=MASCULINO GROUPNAME=GRUPO1 RUNAT=SERVER />

<ASP:RADIOBUTTON TEXT=FEMENINO ID=FEMENINO GROUPNAME=GRUPO1 RUNAT=SERVER /><BR>

<ASP:RADIOBUTTON TEXT=0-10 ID=DIEZ GROUPNAME=GRUPO2 RUNAT=SERVER />

<ASP:RADIOBUTTON TEXT=10-20 ID=VEINTE GROUPNAME=GRUPO2 RUNAT=SERVER /><BR>

<ASP:BUTTON ONCLICK=EVENTO1 TEXT=OK RUNAT=SERVER /><BR>

<ASP:LABEL ID=SEXO RUNAT=SERVER /><BR>

<ASP:LABEL ID=EDAD RUNAT=SERVER /><BR>
```

```
</FORM></HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
if(MASCULINO.Checked) SEXO.Text="MASCULINO";
```

```
if(FEMENINO.Checked) SEXO.Text="FEMENINO";
```

```
if(DIEZ.Checked) EDAD.Text="DE CERO A DIEZ";
```

```
if(VEINTE.Checked) EDAD.Text="DE DIEZ A VEINTE";
```

```
}
```

```
</script>
```

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

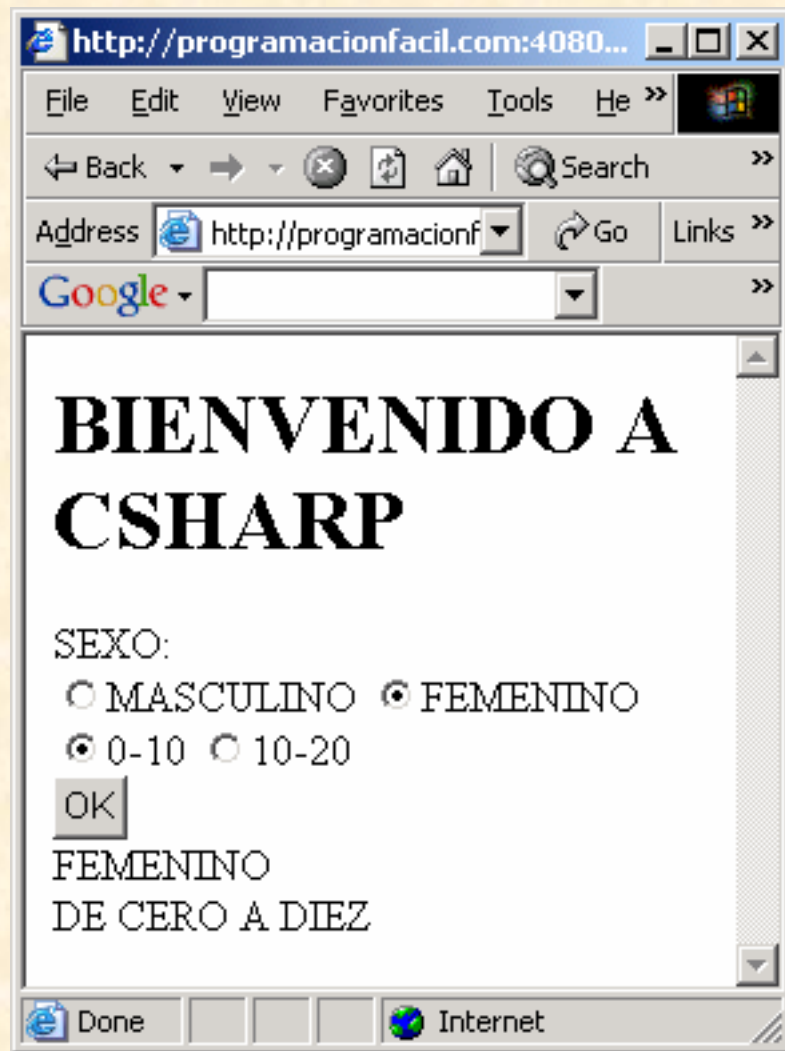
1.- Observar que tenemos dos grupos de radiobotones uno con GRUPNAME=GRUPO1 y otro con GROUPNAME=GRUPO2 sin embargo cada radiobuton tiene su propio valor o ID.

2.- La razón principal para esta situación es que los radiobotones son mutuamente excluyentes entre si Y QUE SOLO UNO PUEDE ESTAR ENCENDIDO A LA VEZ por eso los agrupamos con la propiedad GROUPNAME para que html los pueda considerar como dos o mas grupos diferentes.

3.- Tambien pueden usar la propiedad checked=true para que aparezcan seleccionados al cargar el programa prog11.aspx

4.- Para programarlo usar la misma tecnica que se analizo con CHECKBOX es decir revisar la propiedad checked y un monton de if's (un if por cada radiobutton).

Corrida prog11.aspx



Como se observa checkbox son cajitas con una palomita y radiobutton son circulitos con un puntito negro.

Pero su diferencia mas importante es que radiobtuton no permite que esten seleccionados dos o mas de ellos a la vez (dentro del mismo grupo o groupname).

RADIOBUTTONLIST:

Prog12.aspx

<HTML>

```
<FORM RUNAT=SERVER>
```

```
SEXO:<BR>
```

```
<ASP:RADIOBUTTONLIST ID=SEXO RUNAT=SERVER>
```

```
<ASP:LISTITEM TEXT=MASCULINO RUNAT=SERVER />
```

```
<ASP:LISTITEM TEXT=FEMENINO RUNAT=SERVER />
```

```
<ASP:LISTITEM TEXT=NEUTRO RUNAT=SERVER />
```

```
</ASP:RADIOBUTTONLIST>
```

```
<ASP:BUTTON ONCLICK=EVENTO1 TEXT=OK RUNAT=SERVER /><BR>
```

```
<ASP:LABEL ID=SEX RUNAT=SERVER /><BR>
```

```
</FORM></HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
// como es un control similar a listbox, tambien puede usar prop //  
selecteditem
```

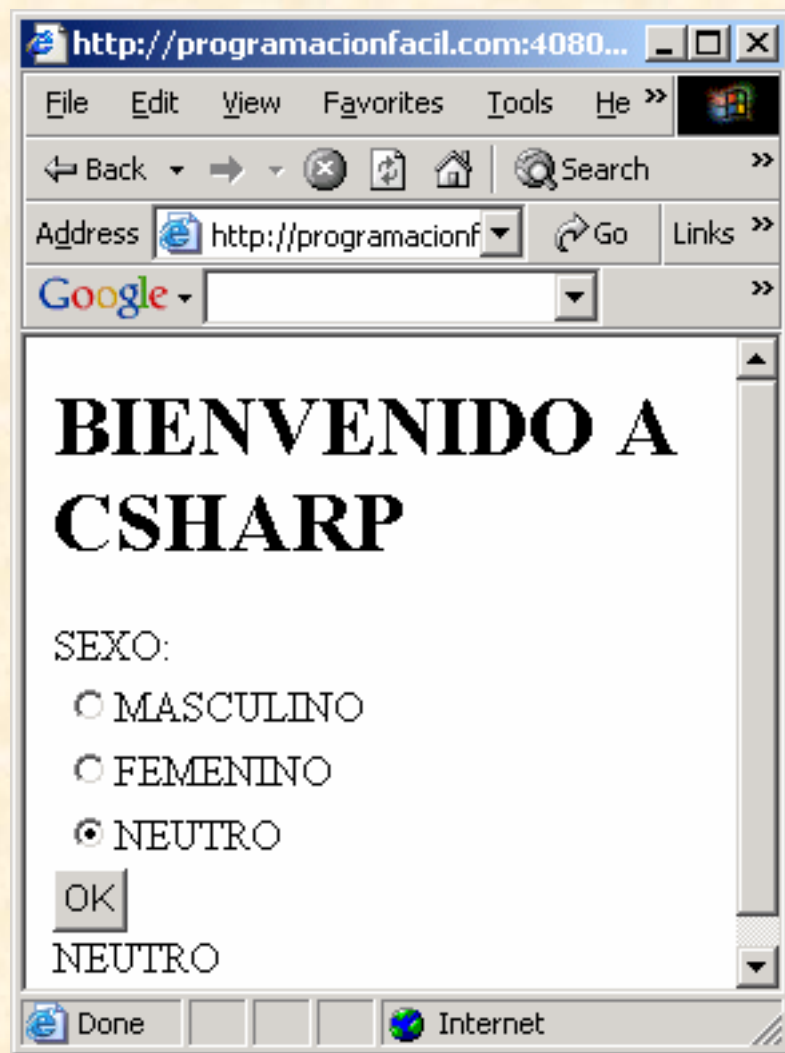
```
SEX.Text = SEXO.SelectedItem.Text;
```

```
}
```

```
</script>
```

Igual que checkboxlist es decir agregarle un ID al radiobuttonlist y un monton de listitem's y programarlo con la propiedad selecteditem que queda apuntando al radiobutton que selecciono el usuario.

Corrida prog12.aspx



NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

PROBLEMAS SUGERIDOS

- 1.- CONSTRUIR UN CUESTIONARIO DE 6 PREGUNTAS SOBRE LOS HÁBITOS DE ESTUDIO DE UN ESTUDIANTE Y PASAR SUS RESPUESTAS Abajo(radiobuton y.aspx).
- 2.- EVALUAR UNA FUNCION CUALESQUIERA para los valores de Y= 3, -5, 10 radiobuttonlist y cs.



UNIDAD 2: INSTRUCCIONES DE CONTROL DE PROGRAMA

TEMA 12: CICLO FOR

Instrucciones para ciclos resuelven el problema de repetir todo el programa o cierta parte del programa mas de una vez.

Este ciclo es uno de los mas usados para repetir una secuencia de instrucciones sobre todo cuando se conoce la cantidad exacta de veces que se quiere que se ejecute una instrucción simple o compuesta.

Su formato general es:

```
for (inicialización; condición; incremento)

{ instrucción(es); };
```

En su forma simple la inicialización es una instrucción de asignación que carga una variable de control de ciclo con un valor inicial.

La condición es una expresión relacional que evalúa la variable de control de ciclo contra un valor final o de parada que determina cuando debe acabar el ciclo.

El incremento define la manera en que la variable de control de ciclo debe cambiar cada vez que el computador repite un ciclo.

Se deben separar esos 3 argumentos con punto y coma (;)

EJEMPLO

Codigo prog13.aspx y recuerden subirlo y pedirlo con:

<http://programacionfacil.com:4080/tusitio/prog13.aspx>.

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

```
<HTML>
```

```
<FORM RUNAT=SERVER>
```

```
<ASP:LISTBOX ID=LISTA ROWS=10 RUNAT=SERVER></ASP:LISTBOX>
```

```
<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/><BR>
```

```
</FORM></HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
int reng;
```

```
LISTA.Items.Clear();
```

```
for(reng=1; reng<=10; reng++)
```

```
LISTA.Items.Add(reng.ToString() + " mama");
```

```
}
```

```
</SCRIPT>
```

nota:

Se esta usando un objeto listbox para procesar el conjunto de datos recordar que listbox, dropdownlist, comboboxlist, etc son objetos similares y por tanto se pueden usar para estos problemas.

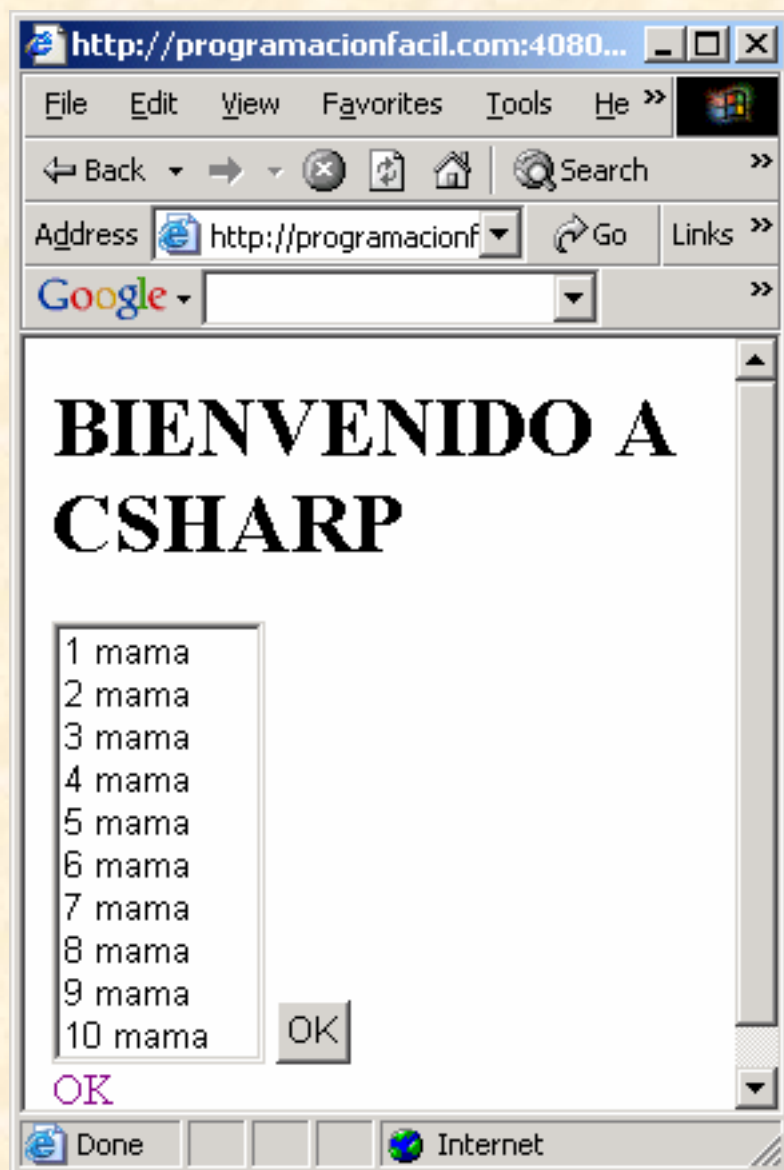
Se esta usando la propiedad add de la coleccion items del componente o control listbox(lista).

Observar que para encadenar strings en csharp se usa el signo +

Como dentro del listbox entran y salen puros datos strings la variable numerica reng de tipo entero se esta convirtiendo a string dentro del listbox.

Y el metodo items.clear() es porque cuando el usuario usa el click mas de una vez el control listbox los agrega abajo por eso en cuanto se activa el onclick lo primero que se realiza es limpiar el listbox.

corrida: prog13.aspx



Casos Particulares del ciclo for:

1.- El ciclo comienza en uno y se incrementa de uno en uno este es el caso mas general.

2.- Pero el valor inicial puede se diferente de uno, ejemplo;

```
for(x=5;x<=15;x=x+1){ etc.};
```

3.- Incluso el valor inicial puede ser negativo, ejemplo;

```
for (x = -3 ;x<= 8; x=x+1) { etc.};
```

4.- Los incrementos también pueden ser diferentes al de uno en uno, ej.;

```
for (x=1; x<= 20; x=x+3){ etc. };
```

5.- Incluso pueden ser decrementos, solo que en este caso, recordar;

5.1.-el valor inicial de la variable debe ser mayor que el valor final.

5.2.-cambiar el sentido de la condición.

ejemplo;

```
for (x= 50 ; x >= 10; x= x-4 ) { etcétera };
```

6.- Solo para los casos de incrementos y decrementos de una en una unidad substituir en el for;

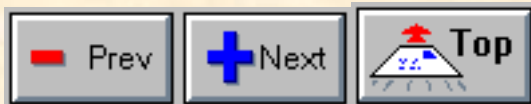
el `x = x + 1` por `x++`

el `x = x - 1` por `x--`

PROBLEMAS SUGERIDOS:

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

- 1.- CONSTRUIR UN PROGRAMA QUE DESPLIEGUE LOS NÚMEROS DEL 20 AL 30.
(aspx y cs)
- 2.- DESPLEGAR LOS ENTEROS ENTRE 50 Y 30 ACOMPAÑADOS DE SU POTENCIA CUADRADA Y RAÍZ CUBICA RESPECTIVA(rvisar tema de operadores aritmeticos).(aspx)
- 3.- DESPLEGAR LOS MÚLTIPLOS DE 5, ENTRE 10 Y 50, ACOMPAÑADOS DE SU FACTORIAL Y LOGARITMO RESPECTIVO(la misma nota de arriba).(cs)
- 4.- DESPLEGAR LA TABLA DE MULTIPLICAR QUE EL USUARIO INDIQUE (aspx)
- 5.- EVALUAR LA FUNCION $Y=5X^2 + 3X + 8$ CUANDO $X \rightarrow -3 \dots 10$ (RANGO DE -3 HASTA 10) (cs)



UNIDAD 2: CONTROL DE PROGRAMA

TEMA 13: CICLO WHILE

En este ciclo el cuerpo de instrucciones se ejecuta mientras una condición permanezca como verdadera en el momento en que la condición se convierte en falsa el ciclo termina.

Su formato general es :

cargar o inicializar variable de condición;

```
while(condición)
```

```
{
```

```
    grupo cierto de instrucciones;
```

```
    instrucción(es) para salir del ciclo;
```

```
};
```

progl4.aspx:

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

<HTML>

<FORM RUNAT=SERVER>

```
<ASP:LISTBOX ID=LISTA ROWS=10 RUNAT=SERVER></ASP:LISTBOX>
```

```
<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/><BR>
```

```
</FORM></HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
int reng=1;
```

```
LISTA.Items.Clear();
```

```
while(reng<=10)
```

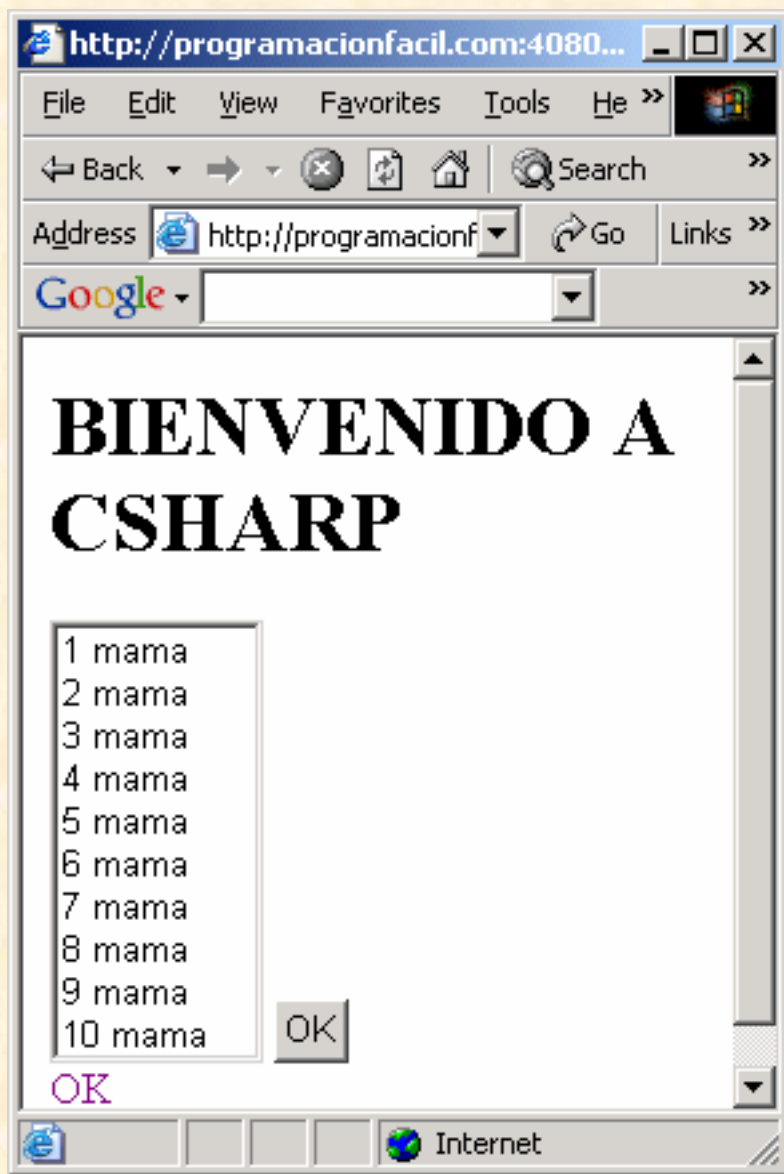
```
{LISTA.Items.Add(reng.ToString()+" mama");
```

```
reng++;};
```

```
}
```

```
</SCRIPT>
```

corrida prog14.aspx



While puede llevar dos condiciones en este caso inicializar 2 variables de condición y cuidar que existan 2 de rompimiento o terminación de ciclo.

El grupo cierto de instrucciones puede ser una sola instrucción o todo un grupo de instrucciones.

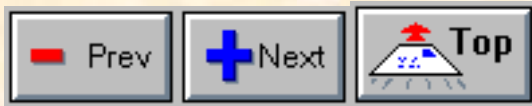
La condición puede ser simple o compuesta.

A este ciclo también se le conoce también como ciclo de condición de entrada o prueba por arriba porque este ciclo evalúa primero la condición y posteriormente ejecuta las instrucciones.

PROBLEMAS SUGERIDOS

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

- 1.- DESPLEGAR ENTEROS ENTRE 50 Y 80(aspx, cs)
- 2.- DESPLEGAR MULTIPLOS DE 4 ENTRE 60 Y 20 ACOMPAÑADOS DE SU LOGARITMOS DE BASE 10 Y BASE e RESPECTIVOS (revisar tema operadores atitmeticos)(aspx, cs)
- 3.- CONSTRUIR LA TABLA DE DIVIDIR QUE EL USUARIO INDIQUE(aspx).
- 4.- Evaluar una funcion cualesquiera para el rango de valores de x de -3 a +5(cs)



UNIDAD 2: CONTROL DE PROGRAMA

TEMA 14: CICLO DO-WHILE

Su diferencia básica con el ciclo while es que la prueba de condición es hecha al finalizar el ciclo, es decir las instrucciones se ejecutan cuando menos una vez porque primero ejecuta las instrucciones y al final evalúa la condición;

También se le conoce por esta razón como ciclo de condición de salida.

Su formato general es :

```
cargar o inicializar variable de condición;

do {

    grupo cierto de instrucción(es);

    instrucción(es) de rompimiento de ciclo;

} while (condición);
```

progl5.aspx

<HTML>

<FORM RUNAT=SERVER>

<ASP:LISTBOX ID=LISTA ROWS=10 RUNAT=SERVER></ASP:LISTBOX>

<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/>


```
</FORM></HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
int reng=1;
```

```
LISTA.Items.Clear();
```

```
do
```

```
{LISTA.Items.Add(reng.ToString()+" mama");
```

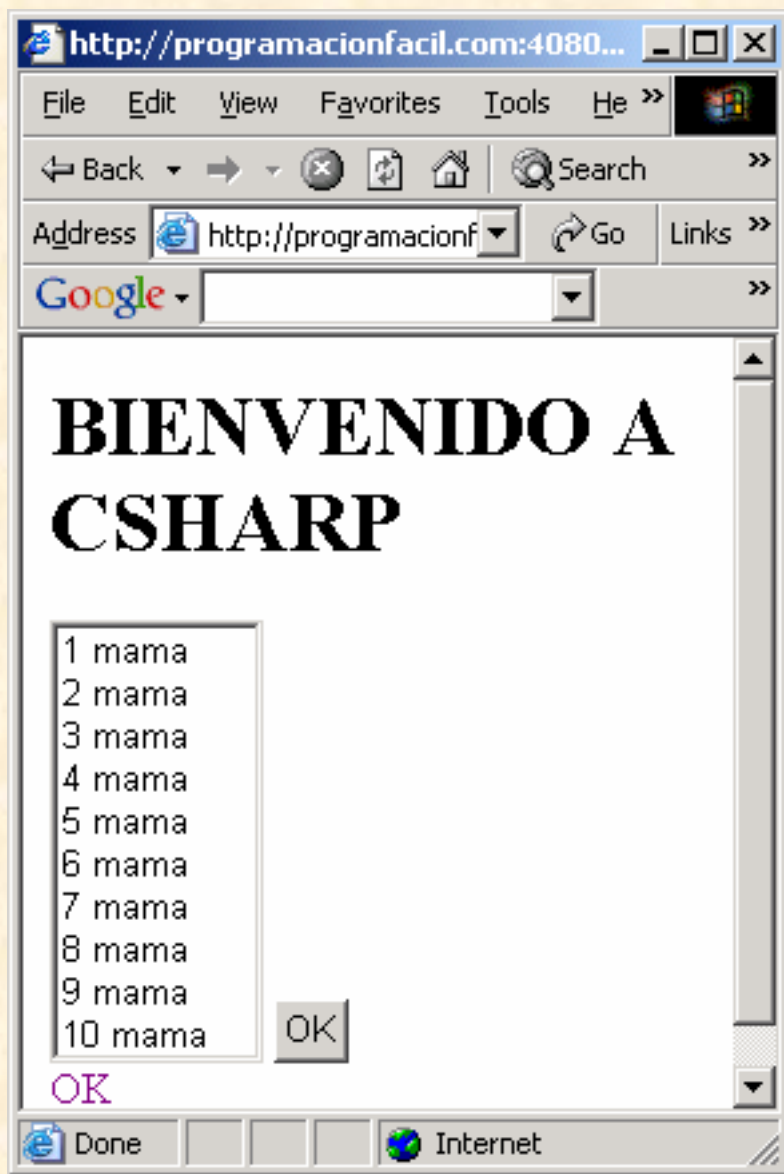
```
reng++; } while(reng<=10);
```

```
}
```

```
</SCRIPT>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

corrida:



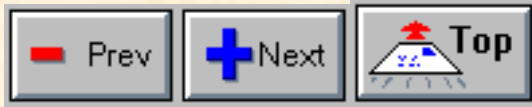
Otra diferencia básica con el ciclo while es que, aunque la condición sea falsa desde un principio el cuerpo de instrucciones se ejecutara por lo menos una vez.

PROBLEMAS SUGERIDOS

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- tres del for

2.- tres del while

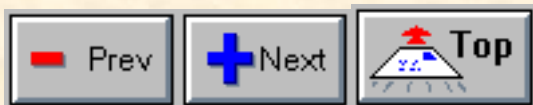


UNIDAD II INSTRUCCIONES DE CONTROL DE PROGRAMA

15.- CONCLUSIONES ACERCA DE CICLOS

El problema de dado un problema cualesquiera cual ciclo se debe usar se resuelve con:

1. Si se conoce la cantidad exacta de veces que se quiere que se ejecute el ciclo o si el programa de alguna manera puede calcularla usar for.
 2. Si se desconoce la cantidad de veces a repetir el ciclo o se quiere mayor control sobre la salida o terminación del mismo entonces usar while.
 3. Si se quiere que al menos una vez se ejecute el ciclo entonces usar do while.
-



UNIDAD 3: ARREGLOS

TEMA 1: INTRODUCCION

Uno de los problemas mas comunes en los diversos sistemas de información es el tratamiento o procesamiento de una gran volumen de datos o de información.

Las variables usadas hasta ahora reciben propiamente el nombre de variables escalares, porque solo permiten almacenar o procesar un dato a la vez.

Por ejemplo si se quiere almacenar nombre y edad de 15 personas con el método tradicional se ocuparan 30 variables y solo es nombre y edad de 15 personas, agreguen mas datos y mas personas y ya es tiempo de empezar a analizar otro tipo de variables.

Es decir, en problemas que exigen manejar mucha información o datos a la vez, variables escalares no son suficientes ya que su principal problema es que solo permiten almacenar y procesar un dato a la vez.

Se ocupan entonces variables que sean capaces de almacenar y manipular conjuntos de datos a la vez.

Variables de tipo **arreglo** si permiten almacenar y procesar conjuntos de datos del mismo tipo a la vez.

Cada dato dentro del arreglo se le conoce como elemento del arreglo y se simboliza y procesa (captura, operación, despliegue) usando el nombre del arreglo respectivo y un subíndice indicando la posición relativa del elemento con respecto a los demás elementos del arreglo, **solo recordar que en csharp la primera posición, elemento o renglón es el 0 (cero), ej.**

NOMBRES

Juan --> nombres(0)

Pedro -> nombres(1)

Rosa --> nombres(2)

Jose --> nombres(3)

Sin embargo sus problemas son similares a los de variables normales es decir hay que declararlos, capturarlos, hacer operaciones con ellos, desplegarlos, compararlos, etc.



UNIDAD 3: ARREGLOS

TEMA 2: ARREGLOS

En programación tradicional siempre se manejan dos tipos de arreglos los arreglos tipo listas, vectores o unidimensionales y los arreglos tipo tablas, cuadros, concentrados, matrices o bidimensionales en ambos casos son variables que permiten almacenar un conjunto de datos del mismo tipo a la vez, su diferencia es en la cantidad de columnas que cada uno de estos tipos contiene, como en los siguientes ejemplos:

a) LISTAS

EDAD

18

34

22

15

B) TABLAS

CIA ACME

ING MENS VTAS

(MILES DE \$)

.....ENE FEB MAR ABR MAY

| | | | | |
|-------|----|-----|-----|-----|
| SUC A | 10 | 20 | 30 | 40 |
| SUC B | 50 | 60 | 70 | 80 |
| SUC D | 90 | 100 | 110 | 120 |

Como se observa la diferencia principal entre un arreglo tipo lista y un arreglo tipo tabla son las cantidades de columnas que contienen.

NOTA IMPORTANTE.- LOS CONCEPTOS MANEJADOS AQUI ESTAN ENFOCADOS A LOS SISTEMAS DE INFORMACION CONTABLE-ADMINISTRATIVOS.

EN ALGEBRA MATRICIAL, SI SON IMPORTANTES LOS CONCEPTOS DE VECTORES Y MATRICES, PERO LAS OPERACIONES Y METODOS SON PRECISAMENTE LOS DEL ALGEBRA MATRICIAL.



UNIDAD 3: ARREGLOS

TEMA 3: ARREGLO TIPO LISTA

Un arreglo tipo lista se define como una variable que permite almacenar un conjunto de datos del mismo tipo organizados en una sola columna y uno o mas renglones.

También reciben el nombre de vectores en álgebra o arreglos unidimensionales en programación.

Los procesos normales con una lista o con sus elementos, incluyen declarar toda la lista, capturar sus elementos, desplegarlos, realizar operaciones con ellos, desplegarlos, etc.

Para declarar una lista se usa el siguiente formato;

```
Tipodato[] nomlista= new tipodato[cant de elementos o renglones];
```

Como se observa por el formato y como ya se ha indicado anteriormente en csharp no existen tipos de datos tradicionales, en su lugar csharp usa objetos derivados de las clases numericas apropiadas, por lo que no debe sorprender que realmente se esta crando un objeto arreglo derivado de la clase de los enteros.

Recordar tambien, que como objeto arreglo, tambien puede usar una serie de metodos pertenecientes a la clase numerica de la cual heredo.

Ejemplos:

```
public static int[] edad= new int[12];
```



```
public static float[] sueldos= new float[10];

public static string[] municipios= new strings[5];
```

Lo de public static, es porque en el programa se va a estar compartiendo el arreglo entre los metodos asociados a dos o mas botones de ordenes (button, buttonlink, imagebutton) el objeto arreglo debera ser de tipo publico o global (este tema se analiza mas ampliamente en la siguiente unidad).

notas:

Recordar que la primera posición o renglón en una lista es la posición o renglón 0 (cero).

El dato capturado, proviene de momento de un componente escalar webcontrol(textbox) y o se usan tantos de estos controles como elementos tenga el arreglo o mas facil aún se usa uno y se va agregando a nuestro arreglo como lo muestra el programa ejemplo.

Progl6.aspx

<HTML>

<FORM RUNAT=SERVER>

DAME EDAD<ASP:TEXTBOX ID=EDAD RUNAT=SERVER/>

<ASP:BUTTON TEXT=CARGAR ONCLICK=CARGAR RUNAT=SERVER/>

<ASP:LISTBOX ID=LISTAORIGINAL ROWS=5 RUNAT=SERVER></ASP:LISTBOX>

<ASP:LISTBOX ID=LISTACOPIA ROWS=5 RUNAT=SERVER></ASP:LISTBOX>

<ASP:BUTTON TEXT=PROCESAR ONCLICK=PROCESO RUNAT=SERVER/>

</FORM></HTML>

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>

public static int[] edad= new int[5];

public static string[] ciudad=new string[10];

public static int reng=0;

void CARGAR (Object sender, EventArgs e)

{

if(reng<=4){

edad[reng]=System.Int32.Parse(EDAD.Text);

reng++;

EDAD.Text=" " ; };

if(reng==5){EDAD.Text="YA SON CINCO";};

}

void PROCESO (Object sender, EventArgs e)

{

// LIMPIANDO LISTAS

LISTAORIGINAL.Items.Clear();

LISTACOPIA.Items.Clear();

//CARGANDO LISTA EDAD CAPTURADA

for (reng=0; reng<=4; reng++)

{ LISTAORIGINAL.Items.Add(edad[reng].ToString()); };

//CALCULANDO Y DESPLEGANDO
```

```
for (reng=0; reng<=4; reng++)

{ edad[reng]=edad[reng]*12; };

//usando ciclo foreach para desplegar

foreach(int r in edad)

{LISTACOPIA.Items.Add(r.ToString() );};

//dejando listo el arreglo para nueva corrida

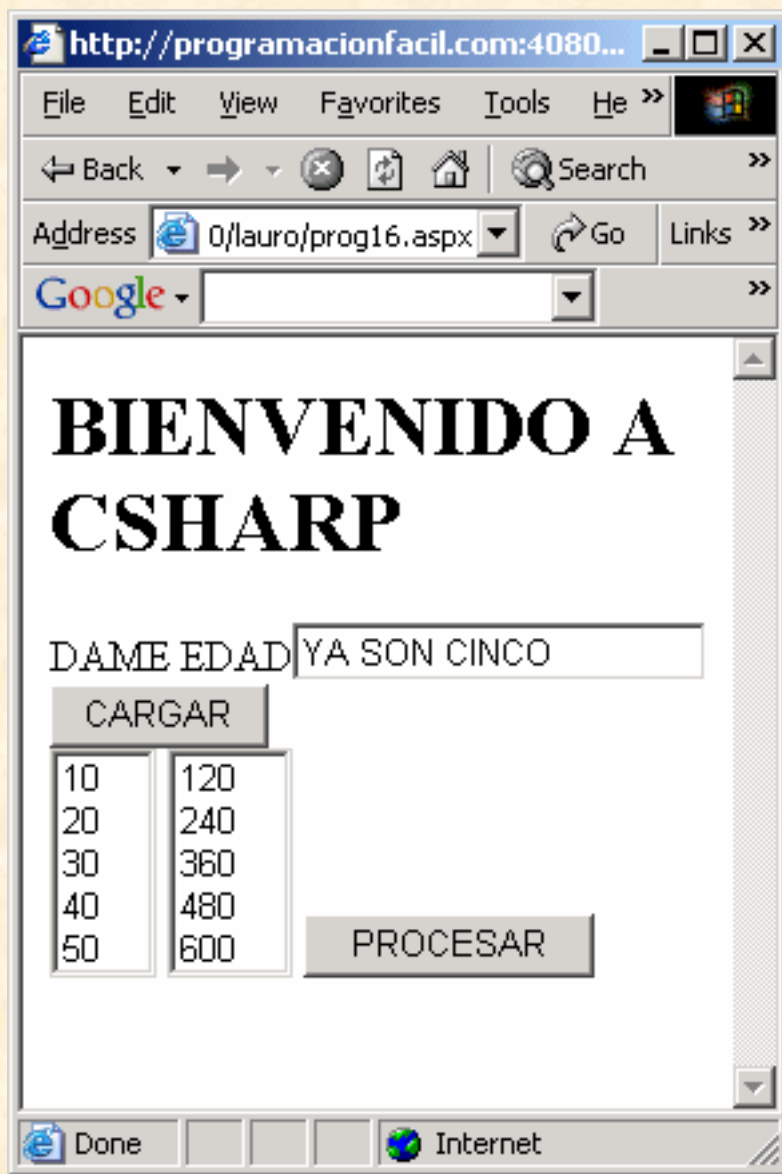
reng=0;

}

</SCRIPT>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

corrida prog16.aspx



notas:

Observar que en el programa el arreglo edad y la variable renglon se declararon de tipo publico y estatico porque los dos metodos el de captura y el de operacion-despliegue, las estan compartiendo.

Para el caso de operaciones y comparaciones con todos los elementos de la lista a la vez se deberá usar un ciclo for con una variable entera llamada renglón, misma que también se usa como índice de la lista.

Recordar que todos los datos internos de la lista estarán almacenados en la memoria ram del computador, para despliegues se usa un

componente visual que permite manipular un conjunto de datos a la vez, el ListBox con sus metodos apropiados pero se tiene que usar un ciclo for() para ir añadiendo o agregando elemento por elemento como se observa en el problema ejemplo que se ha venido desarrollando, en este caso se quiere desplegar las cinco edades convertidas a meses.

Se estan usando metodos apropiados de conversi3n de enteros a strings y viceversa.

Casi al final se usa un ciclo foreach para desplegar el arreglo edad, como se indico en la unidad anterior este ciclo foreach se especializa en la manipulaci3n de arreglos y colecciones (estas se veran en capitulos posteriores), el formato de foreach es:

```
foreach( tipodato varcontrol in arreglo) instruccion(es);
```

Observar tambien que en foreach quien se procesa es la variable de control (r.ToString()) no el arreglo, no se aconseja usar foreach ni para cargar arreglos, ni para actualizarlos, solo para navegar dentro de ellos.

La ultima instrucci3n y muy importante es poner en cero las variables de control de ciclos o indice de arreglos, esto es porque el servidor mantiene el programa ejecutandose continuamente en memoria y si se vuelve a pedir la ejecuci3n del programa, en cuento se intente capturar un nuevo dato va a marcar el error arreglofuera del limite o arrayofbound, estan avisados.

Para inicializar una lista se debe usar el siguiente formato:

```
tipodato[] nomlista={lista de valores};
```

ej;

```
int[] edad={15,16,17,18};
```

```
float[] sueldo={40.85, 65.30, 33.33};
```

```
string[] ciudad={"tecate", "tijuana", "mexicali", "rosarito",  
"ensenada"};
```


PROBLEMAS SUGERIDOS

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- Capturar y desplegar 5 precios de productos cualesquiera usando dos panel, uno para capturar y uno para desplegar(2 aspx uno capturado y otro inicializado).

2.- Capturar 4 sueldos en un panel desplegarlos aumentados en un 25% en otro panel (2 cs uno capturado y otro inicializado).

3.- Capturar los datos de 5 productos comprados en una tienda, incluyendo nombre, precio y cantidad en sus 3 listas respectivas, después calcular una cuarta lista con el gasto total por cada producto desplegarlo todo en un segundo panel e incluir también el gran total(aspx).

4.- Capturar en una lista solamente 6 números múltiplos de 5, se debe de estar capture y capture números hasta que se completen los 6 múltiplos de 5(cs)



UNIDAD 3: ARREGLOS

TEMA 4: ARREGLOS TIPO TABLA

Un arreglo tipo tabla se define como un conjunto de datos del mismo tipo organizados en dos o mas columnas y uno o mas renglones.

Para procesar (recordar solo operaciones y comparaciones) internamente todos los elementos de la tabla se ocupan dos ciclos for (), uno externo para controlar renglón y uno interno para controlar columna.

Los elementos de la tabla se deberan simbolizar con el nombre de la tabla y 2 subindices, el primer subindice referencia al renglon y el siguiente subindice referencia la columna los dos dentro del mismo corchete.

La declaración de una tabla sera de acuerdo al siguiente formato:

```
Public static tipodato[,] nomtabla=new tipodato[cant reng, cantcol];
```

```
Ej: public static float[,] sueldos=new float[5,8];
```

Para capturar sus elementos, usaremos un textbox y un boton de captura, solo tener cuidado o mucho control sobre los indices ren y col como lo muestra el ejemplo.

Para efectuar otros procesos tales como operaciones, despliegue con todos los elementos de la tabla se deberan usar 2 ciclos, un for externo para controlar renglon y un for interno para controlar columna.

Progl7.aspx

```
<HTML>

<FORM RUNAT=SERVER>

Ren<ASP:TEXTBOX ID=REN SIZE=2 VALUE=0 RUNAT=SERVER/>

Col<ASP:TEXTBOX ID=COL SIZE=2 VALUE=0 RUNAT=SERVER/><BR>

DAME CALIF<ASP:TEXTBOX ID=CALIF SIZE=10 RUNAT=SERVER/>

<ASP:BUTTON TEXT=CARGAR ONCLICK=CARGAR RUNAT=SERVER/><BR>

<ASP:LISTBOX ID=L1 ROWS=5 RUNAT=SERVER></ASP:LISTBOX>

<ASP:LISTBOX ID=L2 ROWS=5 RUNAT=SERVER></ASP:LISTBOX>

<ASP:LISTBOX ID=L3 ROWS=5 RUNAT=SERVER></ASP:LISTBOX>

<ASP:BUTTON TEXT=PROCESAR ONCLICK=PROCESO RUNAT=SERVER/><BR>

</FORM></HTML>

<SCRIPT LANGUAGE=C# RUNAT=SERVER>

static int[,] calif= new int[2,3];

void CARGAR (Object sender, EventArgs e)

{

int reng=System.Int32.Parse(REN.Text);

int col=System.Int32.Parse(COL.Text);

calif[reng,col]=System.Int32.Parse(CALIF.Text);

col++;

CALIF.Text="  ";
```

```
if (col==3){reng++; col=0;};

if (reng==2){CALIF.Text="TABLA LLENA";};

REN.Text=reng.ToString();

COL.Text=col.ToString();

}

void PROCESO (Object sender, EventArgs e)

{

// procesando y regalando 10 puntos a la calificacion

for(int reng=0; reng <= 1; reng++)

for(int col=0; col <=2; col++)

{calif[reng,col]=calif[reng,col] +10;};

// desplegando

for(int reng=0; reng<=1; reng++)

{ L1.Items.Add((calif[reng,0]).ToString());

L2.Items.Add((calif[reng,1]).ToString());

L3.Items.Add((calif[reng,2]).ToString());

}; }

</SCRIPT>
```

Notas:

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Observar el formato de declaración y como se controlan los indices de captura reng, col.

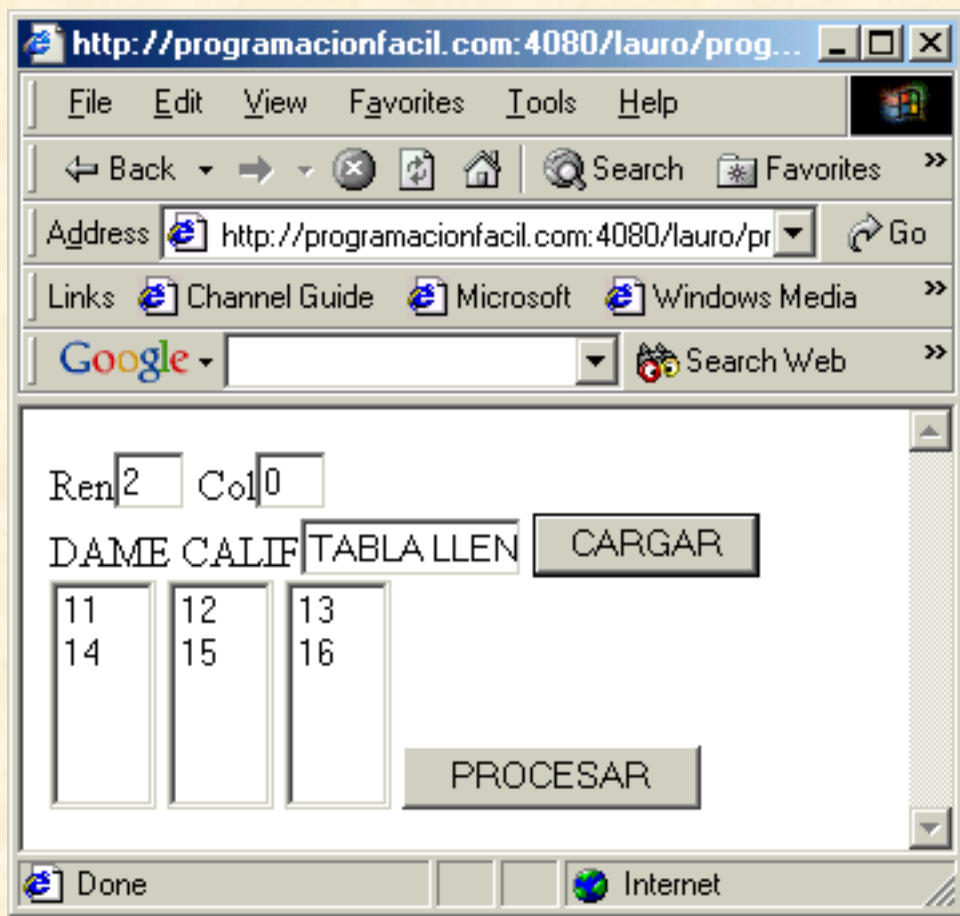
Para procesar los elementos se usan dos ciclos for y el formato tabla [reng,col].

Recordar que un programa en NET es un conjunto de objetos de diferente fuente interactuando entre si, en este problema se uso el objeto LISTBOX para preesentar el resultado , mas adelante se usara un objeto mas apropiado.

Observar que no se ocupo al final poner en cero las variables de control (reng, col) porque:

- a) Se estan al principio cargando de las cajas de texto ren,col
- b) Estas cajas de texto si se pusieron en 0(cero) en una de las intrucciones if.

Corrida prog17.aspx



Para inicializar tablas, se usa el siguiente formato:

```
tipodato[,] nomtabla= { {val reng 0}, {val reng 1}, {val reng n} };
```

ejemplo una matriz de 3 x 4 calificaciones:

```
int[,] calif= { { 10,20,30,40}, { 50,60,70,80}, {90,10,20,30} };
```

PROBLEMAS SUGERIDOS

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- CONSTRUIR UN CUADRO QUE CONTENGA LOS COSTOS FIJOS DE CUATRO PRODUCTOS CUALESQUIERA, QUE SE PRODUCEN EN TRES PLANTAS DIFERENTES DE UNA EMPRESA MAQUILADORA(2 aspx uno capturado y otro inicializado).

2.- CONSTRUIR UN CUADRO QUE CONTENGA LOS INGRESOS MENSUALES POR

VENTAS DURANTE LOS TRES PRIMEROS MESES DEL AÑO DE CUATRO SUCURSALES DE UNA CADENA DE AUTO REFACCIONES, AGREGAR AL FINAL UNA LISTA QUE MUESTRE LOS INGRESOS MENSUALES TOTALES POR MESES Y UNA SEGUNDA LISTA QUE MUESTRE LOS INGRESOS MENSUALES TOTALES POR SUCURSAL(2 cs uno capturado y otro inicializado).

3.-CONSTRUIR UN CUADRO QUE CONTENGA LAS COMISIONES GANADAS POR TRES VENDEDORES, DE LOS 5 TIPOS DE LINEA BLANCA DE CONOCIDA MUEBLERIA, ADEMAS LISTAS DE COMISIONES TOTALES Y PROMEDIOS GANADAS POR LOS VENDEDORES, ASI COMO LISTAS DE COMISIONES TOTALES Y PROMEDIOS POR TIPO DE LINEA BLANCA (aspx y cs).

ANALIZAR ESTE CODIGO:

' PARA TOTALES Y PROMEDIOS POR RENGLON

FOR R = 0 TO 3

FOR C = 0 TO 2

TOTRENG(R) = TOTRENG(R) + TABLA(R,C)

NEXT C

PROMRENG(R) = TOTRENG(R)/3

NEXT R

'PARA TOTALES Y PROMEDIOS POR COLUMNA

FOR C = 0 TO 2

FOR R = 0 TO 3

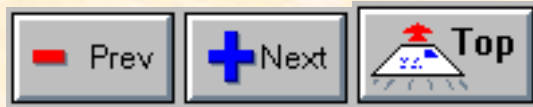
TOTCOL(C)=TOTCOL(C) + TABLA(R,C)

NEXT R

PROMCOL(C) = TOTCOL(C) / 4

NEXT C

SUGERENCIA: CONSTRUIR PRIMERO LOS CUADROS EN PAPEL.



UNIDAD 3: ARREGLOS

TEMA 5: ARREGLOS IRREGULARES (JAGGED)

Hasta ahora listas y tablas son conocidos como arreglos o matrices rectangulares, sin embargo la tecnologia microsoft.net ofrece un nuevo tipo de arreglos, denominados arreglos irregulares (jagged).

Este tipo de arreglo es un arreglo de arreglos, pero cada arreglo puede ser de diferente tamaño, es decir a veces en problemas especiales se puede ocupar una matriz, cuadro, tabla o concentrado donde el tamaño de cada renglón sea DIFERENTE.

Para declarar un arreglo irregular se usara el siguiente formato:

```
Tipodato[][] nomarreglo=new tipodato[cant reng][];
```

```
Ej; int[][] tabla=new int[3][];
```

Observar los cambios en la declaración en cuanto un arreglo de tipo tabla normal, es decir ahora son dos corchetes y no llevan la coma dentro de ellos.

Para procesar los elementos, de un arreglo irregular, se debera ahora manejar dos corchetes, no uno con dos indices como se realizó en tablas rectangulares.

La pobre gente de c++ usan este metodo para procesar tablas, menuda sorpresa les espera con csharp.

Observar en la declaración que ya se determino la cantidad de renglones que tendra el arreglo irregular, pero no se ha determinado el tamaño de cada uno de los renglones, para realizar esto se debera incluir en la declaración del arreglo irregular, una definicion de

cada renglon usando el siguiente formato;

```
nomarreglo[reng]=new tipodato[cant de columnas];
```

ejemplo completo de declaración;

```
int[][][] tabla=new int[3][];
```

```
tabla[0]= new int[5]; tabla[1]= new int[2]; tabla[2]= new int[3];
```

Para este caso se tiene un arreglo de tres renglones, pero el primer renglon tendra 5 columnas o elementos, el segundo renglon tendra 2 columnas o elementos y el tercer renglon tendra 3 columnas o elementos, en total el arreglo irregular contendra 5+2+3=10 elementos.

Para realizar procesos se deberan usar tantos ciclos for(columnas) como renglones tenga el arreglo y se puede manejar como constante el renglon en cada ciclo, como lo muestra el ejemplo;

progl8.aspx

```
<HTML>
```

```
<FORM RUNAT=SERVER>
```

```
<ASP:BUTTON TEXT=PROCESAR ONCLICK=PROCESO RUNAT=SERVER/><BR>
```

```
</FORM></HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void PROCESO (Object sender, EventArgs e)
```

```
{
```

```
// creando el arreglo iregular de tres renglones
```

```
int[][][] tabla=new int[3][];
```

```
// creando y cargando los tres renglones pero de diferente tamano
```



```
tabla[0]= new int[4]; tabla[1]=new int[2]; tabla[2]=new int[5];

//inicializando renglones

// tambien puede ser uno a uno pero son 4 + 2 + 5=11 datos

for(int col=0; col<=3; col++) tabla[0][col]=10;

for(int col=0; col<=1; col++) tabla[1][col]=20;

for(int col=0; col<=4; col++) tabla[2][col]=30;

// para procesar cada renglon usar el metodo de arriba ejemplo

// sumar 2 al primer renglon, 3 al segundo renglon y 4 al tercer renglon

for(int col=0; col<=3; col++) tabla[0][col]=tabla[0][col]+2;

for(int col=0; col<=1; col++) tabla[1][col]=tabla[1][col]+3;

for(int col=0; col<=4; col++) tabla[2][col]=tabla[2][col]+4;

// para desplegar usar el mismo metodo

for(int col=0; col<=3; col++)

Response.Write(tabla[0][col]+"..");

Response.Write("<br>");

for(int col=0; col<=1; col++)

Response.Write(tabla[1][col]+"..");

Response.Write("<br>");

for(int col=0; col<=4; col++)

Response.Write(tabla[2][col]+"..");

Response.Write("<br>Observan porque es irregular");
```

```
}
```

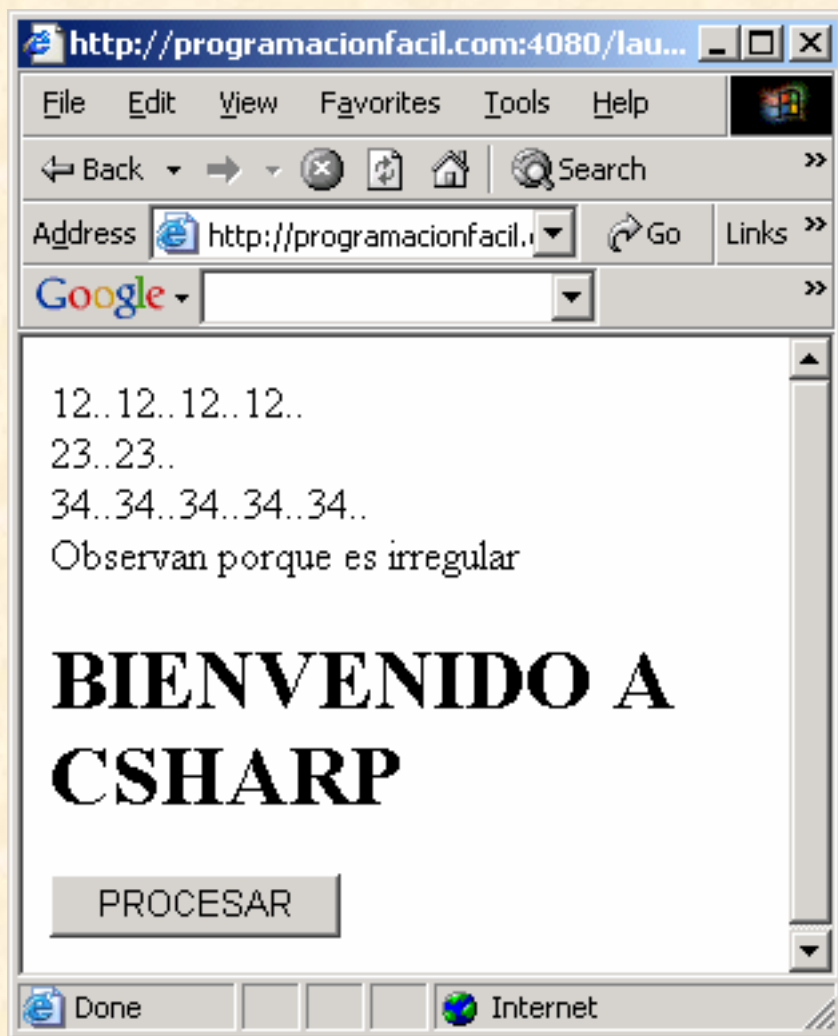
```
</SCRIPT>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Observar como se manipulan ahora los elementos con el formato tabla [reng][col].

Observar tambien que se esta usando el Objeto RESPONSE de la libreria ASP vieja, esta libreria antigua tiene siete objetos que en algunas ocasiones nos permiten resolver algunos problemas, al final de este capitulo se incluye el apendice con los objetos de esta libreria.

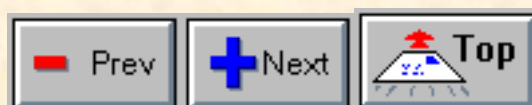
Corrida prog18.aspx



problemas sugeridos:

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- construir un par de tablas irregulares, con algunos procesos, una de ellas en aspx y otra en cs



WWW.PROGRAMACIONFACIL.COM

UNIDAD 3: ARREGLOS

TEMA 6: LISTBOX

ListBox uno de los nuevos WebControls, es un componente DINAMICO(es decir no tiene tamaño definido) que permite procesar visualmente un conjunto de elementos de tipo string.

La propiedad Rows que se usa al crearlo, es solo para indicarle cuantos renglones desplegara en pantalla, es decir si se usa rows=5, en listbox se podra capturar todos los elementos o datos que se quiera, pero solo desplegara los ultimos cinco elementos.

Sin embargo existen ciertas propiedades del listbox que permiten conocer cuantos elementos estan cargados en el listbox.

Otra importante aspecto a recordar cuando se procese o programe, es que el primer indice de la lista, es el indice numero 0(cero).

Este componente, contiene muchas propiedades y métodos que facilitan el trabajo con datos la mas importante es su propiedad ITEMS, que a su vez tiene:

PROPIEDAD ACCIÓN O SIGNIFICADO

Items.Add(dato): Inserta un elemento al final del listbox.

Items.Clear(): Elimina todos los elementos de la lista.

Items.Count(): Regresa la cantidad de elementos en lista.

Items.Sorted=true; Ordena los elementos de la lista usada solo al

tiempo de diseño.

Items.Contains(dato): Regresa true o false, si el dato se encuentra o no se encuentra en la lista.

Items.IndexOf(dato): Regresa el indice del objeto dentro del listbox.

Items.Insert(indice,dato): Inserta el dato en la posición indicada.

Items.Remove(dato): Elimina el dato de el listbox.

Items.RemoveAt(indice): Elimina el dato que esta en la posición indicada.

Items[indice].Text: get or set el dato en la posición indicada (ver primera nota abajo).

Notas:

Como ya se indico anteriormente GET y SET son propiedades asociadas a todos los objetos o controles y sus propiedades de microsoft.net, por ejemplo para un textbox, si en un programa se dice `alfa=text5.text;` se esta usando get, si se dice `text5.text=500;` se esta usando set.

Otro ejemplo `alfa=listbox2.Items[2].Text;` se esta usando (get)

`listbox2.Items[4].Text="mama";` se esta usando (set).

Esto de get-set se puede usar para cualquier propiedad, por ejemplo `alfa=listbox8.background;` se esta usando get, pero si se codifica `listbox8.background=amarillo1` se esta usando set, como se observa es importante entender y aplicar este GET-SET en todos los programas.

Capturas: Solo se ocupara un Text, el evento click del button, y el método Add del ListBox.

Proceso: Se ocupara un ciclo for y los métodos count y text de `ListBox.Items[indice].`

Despliegues: No se ocupa porque todos los cambios son visibles.

Pero si se quiere pasar de un ListBox a otro ListBox, entonces ciclo for, count, etc.

ejemplo prog19.aspx

```
<HTML>
```

```
<FORM RUNAT=SERVER>
```

```
EDAD<ASP:TEXTBOX ID=DATO RUNAT=SERVER/>
```

```
<ASP:BUTTON TEXT=CAPTURAR ONCLICK=INSERTAR RUNAT=SERVER/><BR>
```

```
<ASP:LISTBOX ID=LISTA ROWS=5 RUNAT=SERVER></ASP:LISTBOX>
```

```
<ASP:BUTTON TEXT=PROCESAR ONCLICK=PROCESAR RUNAT=SERVER/><BR>
```

```
</FORM></HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void INSERTAR (Object sender, EventArgs e)
```

```
{
```

```
LISTA.Items.Add(DATO.Text);
```

```
DATO.Text=" ";
```

```
}
```

```
void PROCESAR (Object sender, EventArgs e)
```

```
{
```

```
int reng, cant, meses;
```

```
cant=LISTA.Items.Count;

for (reng=0; reng<=cant-1; reng++)

{

meses=Int32.Parse(LISTA.Items[reng].Text);

meses=meses*12;

LISTA.Items[reng].Text=meses.ToString();};

}
```

</SCRIPT>

Recordar que el primer índice en un ListBox es el cero por eso el ciclo va desde el cero hasta la cantidad de elementos menos uno.

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Corrida:



COMO ULTIMA NOTA IMPORTANTE ES QUE EXISTEN OTROS DOS CONTROLES QUE PUEDEN COMPORTARSE Y USAR LOS MISMOS METODOS ASOCIADOS A LISTBOX, ESTOS CONTROLES SON EL COMBOBOX Y DATALIST(EN PROXIMOS CURSOS SE INCLUIRAN COMO DOS TEMAS EXTRAS)

PROBLEMAS SUGERIDOS

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- CAPTURAR EN UNA LISTA LOS SUELDOS DE 6 EMPLEADOS Y DESPLEGARLOS EN UNA SEGUNDA LISTA AUMENTADOS EN UN 30% (aspx)

2.- CAPTURAR EN UNA LISTA LOS PESOS EN KILOGRAMOS DE 6 PERSONAS DESPLEGARLOS EN UNA SEGUNDA LISTA CONVERTIDOS A LIBRAS Y ADEMAS SOLO LOS MAYORES DE 100 LIBRAS. (cs)

3.- CAPTURAR EN SUS 4 LISTAS RESPECTIVAS MATRICULA, NOMBRE Y DOS CALIFICACIONES DE 5 ALUMNOS, DESPUÉS CALCULAR UNA LISTA DE PROMEDIOS DE CALIFICACIONES. (aspx)

4.-CAPTURAR EN SUS LISTAS RESPECTIVAS NUMEMPLEADO, NOMEMPLEADO, DÍAS TRABAJADOS Y SUELDO DIARIO DE 5 EMPLEADOS, DESPLEGAR EN OTRA PANTALLA O PANEL LA NOMINA PERO SOLO DE AQUELLOS EMPLEADOS QUE GANAN MAS DE \$300.00 A LA SEMANA. (cs)



UNIDAD 3: ARREGLOS**TEMA 7: TABLAS VISUALES (TABLE)**

Tablas visuales nos permiten presentary manipular información al usuario en forma tabular, pero tambien se pueden utilizar para formatear diferentes tipos de informacion en una pagina web (observar que en varias partes de este libro sobre todo en cuadros se han estado usando tablas para formatear la información)

Uno de los elementos importantes a entender con respecto a componentes o tablas visuales en el modelo de programación que se ha estado analizando, es que existen los siguientes tipos de tablas visuales:

El objeto TABLE de HTML que es el control mas basico de este tipo.

El WAPPER de este objeto es decir HTMLTABLE con una nueva serie de atributos y metodos.

El WEBCONTROL TABLE que permite procesar y manipular un objeto de tipo tabla en forma natural, con propiedades y metodos muy comunes a todas las propiedades y metodos de los diversos WEBCONTROLS que se han venido estudiando a lo largo de este libro.

DATAGRID WEBSERVERCONTROL es un excelente objeto que permiten desplegar y editar tablas, con la especificacion de que un datagrid debera estar enlazado(databound) fisicamente a una fuente de datos(datasource) del mismo tipo, por ejemplo una variable de tipo arreglo como las que se analizo en temas pasados o por ejemplo una tabla de ACCESS.

Para proposito de este tema se analizara el control TABLE DE WEBCONTROL, en razon de que este curso se ha centrado en WEBCONTROLS y el componente, objeto o control DATAGRID se analizara y usara mas adelante en la quinta unidad.

Este componente es de los mas importantes para el procesamiento de muchos datos permite concentrar, procesar y mostrar gran cantidad de información a la del vista del usuario.

Este componente presenta, manipula y procesa conjuntos de datos de tipo strings en forma tabular, es decir en forma de tablas, matrices, cuadros concentrados, ejemplo;

CIA ACME**INGRESOS POR VENTAS MENSUALES****MILLONES DE PESOS**

| | ENE | FEB | MAR | ABR |
|-------|-----|-----|-----|-----|
| Suc A | 1 | 2 | 3 | 4 |
| Suc B | 5 | 6 | 7 | 8 |

Recordar que son los datos numéricos internos quienes se procesan (es decir, se capturan, se realizan operaciones con ellos, se despliegan, etc.) es la información externa quien le da sentido.

Es importante tambien entender que un objeto **TABLE** es una colección o conjunto de objetos de tipo **TABLEROW**(y por ser objetos cada renglon tambien debera ser creado usando el operador **new**) ademas de que cada celda en cada uno de los renglones tambien sonun objeto de tipo **TABLECELL**(es decir cada celda tambien debera ser creada usando **new**).

Algunas de sus propiedades y métodos mas interesantes que se usan en el programa ejemplo son:

CellPadding.- Se usa para definir el tamaño del marco o separador de la celdas

GridLine.- Se usa para indicar si el marco debera ser solo vertical, horizontal o ambos.

Text.- Propiedad de la celda(**TABLECELL**) que se usa para manipular y procesar el dato.

Add.- Otra propiedad de **CELL** y tambien de **ROW** que se puede usar para cargar un dato en una celda o una celda en un renglon.

Como se observa de las dos propiedades anteriores **CELDA** es el elemento mas importante de una **TABLA**, **CELDA** tiene muchas propiedades y metodos y en el caso particular de **TEXT** y **ADD** recordar que **TEXT** tiene las características de **get-set** que ya se ha analizado en temas anteriores y por tanto sera mas util o usado que el metodo **ADD**.

Otro aspecto importante a recordar es que **TABLE** no permite edición directa por parte del usuario de sus celdas por ese motivo se usara un componente externo **TextBox** para capturas, así como el evento **click** de un **button** apropiado.

Para procesar **todos** los elementos de la tabla solo recordar que se deben usar dos ciclos **for uno externo para controlar renglones y uno interno para controlar columna**.

Si solo se quiere procesar un solo renglón o columna, entonces solo se ocupara el ciclo **contrario** y el renglón o columna original se darán como constantes.

Ejemplo prog20.aspx

```
<% // creando una pagina %>
```

```
<%@ PAGE LANGUAGE=C# %>
```

```
<% // creando y definiendo el objeto tabla %>
```

```
<FORM RUNAT=SERVER>
```

```
<ASP:TABLE ID=TABLA RUNAT=SERVER CELLPADDING=10 GRIDLINES=BOTH /></ASP:TABLE>
```

```
Ren<ASP:TEXTBOX ID=REN SIZE=2 VALUE=0 RUNAT=SERVER/>

Col<ASP:TEXTBOX ID=COL SIZE=2 VALUE=0 RUNAT=SERVER/><BR>

DAME DATO<ASP:TEXTBOX ID=DATO SIZE=10 RUNAT=SERVER/>

<ASP:BUTTON TEXT=CARGAR ONCLICK=CARGAR RUNAT=SERVER/><BR>

<ASP:BUTTON TEXT=PROCESAR ONCLICK=PROCESO RUNAT=SERVER/><BR></FORM>

<SCRIPT RUNAT=SERVER>

//CREANDO E INICIALIZANDO TABLA AL TIEMPO DE EJECUCION

void Page_Load(Object sender, EventArgs e)

{

// GENERANDO E INICIALIZANDO RENGLONES Y COLUMNAS

// creando los renglones de la tabla

for (int cantreng=1; cantreng <= 3; cantreng++)

{ TableRow renglon = new TableRow();

// creando las celdas o columnas del renglon

for (int cantcol=1; cantcol <= 4; cantcol++)

{TableCell celda = new TableCell();

// cargando celda con un dato cualquiera para inicializar

celda.Text="dato";

// cargando la celda al renglon

renglon.Cells.Add(celda);

}

// cargando el renglon a la tabla

TABLA.Rows.Add(renglon);

}

}

void CARGAR (Object sender, EventArgs e)

{

int reng=System.Int32.Parse(REN.Text);

int col=System.Int32.Parse(COL.Text);
```

```

TABLA.Rows[reng].Cells[col].Text =DATO.Text;

col++;

DATO.Text=" ";

if (col==4){reng++; col=0;};

if (reng==3){reng=0;col=0;DATO.Text="TABLA LLENA";};

REN.Text=reng.ToString();

COL.Text=col.ToString();

}

void PROCESO (Object sender, EventArgs e)

{

int temp;

// procesando y sumandole 10 puntos al dato y desplegando

for(int reng=0; reng <= 2; reng++)

for(int col=0; col <= 3; col++)

{

temp= System.Int32.Parse(TABLA.Rows[reng].Cells[col].Text);

temp=temp+10;

TABLA.Rows[reng].Cells[col].Text=temp.ToString(); };

}

</SCRIPT>

```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Se empieza creando un objeto de tipo pagina (PAGE) de html la razon de esto es doble, primero porque se necesita crear y tambien inicializar un objeto table y se aprovecha el metodo ONLOAD DE PAGE para realizar este proceso, es decir en cuanto se carga la pagina se inicializa la tabla.

En cuanto a la creación e inicialización de la tabla recordar que TABLEROW y TABLECELL son objetos y por tanto se usó el operador new.

Recordar tambien que cuando se crean o inicializan tablas se usan CANTRENG y CANTCOL, sim embargo cuando se va a acceder o manipular las celdas mas adelante en el programa se debera usar notación [0][0].

Observar que Celda usa el metodo set en CELDA.TEXT=DATO;

Para cargar la celda al renglón respectivo asi como para cargar el renglón a la tabla(otro objeto tambien) se usa el metodo ADD.

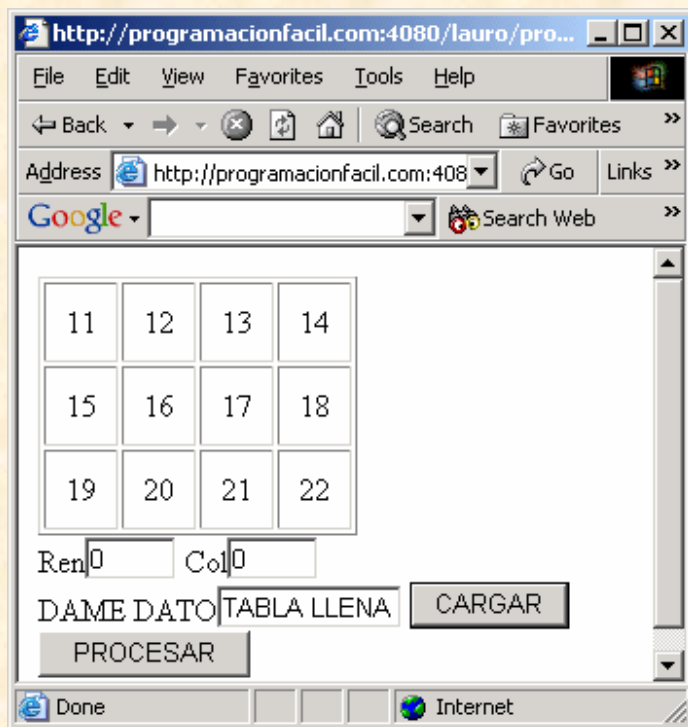
Ya dentro del metodo de CARGAR observar que para procesar una celda determinada, se debera usar toda la ruta completa es decir TABLA.RENGLON.CELDA; en el caso de renglón y columna o celda respectiva usando su [indice] correspondiente.

Como se esta cargando TEXT de TextBox a TEXT de CELL, solo se igualarán las propiedades respectivas.

El resto de codigo en CARGAR ya se analizó en tema anterior solo es para validar no salir de los limites de la tabla, avisar cuando ya se lleno y dejarla lista para una nueva captura se usan los dos textboxes como indices de reng y col respectiva.

En el metodo PROCESO, se usan los ciclos normales de for(renglon) y for(columna), los indices empiezan en 0, y el metodo TEXT de CELL con todas las notas vistas en temas anteriores.

Corrida prog20.aspx



Un proceso muy común con tablas, cuadros y concentrados es agregarles listas de totales y promedios ya sea por columna o por renglón, o ambas por ejemplo:

CIA ACME

INGRESOS MENSUALES

(MILES DE PESOS)

| | ene | feb | mar | totsuc | promsuc |
|--------|-----|-----|-----|--------|---------|
| Suc a | 1 | 2 | 3 | 4 | 2 |
| Suc b | 4 | 5 | 6 | 15 | 5 |
| Suc c | 7 | 8 | 9 | 24 | 8 |
| Suc d | 10 | 11 | 12 | 33 | 11 |
| totmes | 22 | 26 | 30 | | |
| promes | 5.5 | 6.5 | 7.9 | | |

En este ejemplo aparte de la tabla se ocupan 4 listas dos para totales y dos para promedios.

El Codigo para este tipo de problemas ya se dio en el tema de arreglos normales tipo tabla.

PROBLEMAS SUGERIDOS

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

- 1.- Construir un concentrado que despliegue los costos fijos de tres diversos productos que se fabrican en cuatro sucursales de una empresa MAQUILADORA.
- 2.- Construir un concentrado que contenga los ingresos por ventas mensuales de los 4 primeros meses del año de tres sucursales de una cadena refaccionaría, agregar listas de ingresos totales por mes e ingresos promedios por sucursal.
- 3.- Construir un cuadro que contenga las calificaciones de 5 materias de cuatro alumnos cualesquiera, incluir promedios de calificaciones por materia y por alumno.



UNIDAD 4: PROCEDIMIENTOS Y FUNCIONES

TEMA 1: PROCEDIMIENTOS

Recordar que un objeto presenta tres aspectos, propiedades, metodos y eventos, en esta unidad se analizan algunos elementos que intervienen en la definicion de un metodo.

Estamos hablando de los llamados procedimientos y funciones, que quede claro que procedimientos y funciones son solo algunos aspectos (importantes) de la definicion de un metodo, pero que existen elementos tan o mas importantes que los analizados en esta unidad.

Un procedimiento es un grupo de instrucciones, variables, constantes, etc, que estan diseñados con un propósito particular y tiene su nombre propio.

Es decir un procedimiento es un modulo de un programa que realiza tareas especificas y que no puede regresar valores al programa principal u a otro procedimiento que lo este invocando.

Despues de escribir un procedimiento se usa su propio nombre como una sola instrucción o llamada al procedimiento.

Su formato es **`void NomProc(){instrucciones;};`**

Un programa puede tener tantos procedimientos como se deseen, para hacer una llamada o invocación al procedimiento durante la ejecución de un programa solo se debera escribir el nombre del procedimiento y los parentesis en blanco.

Prog21.aspx

```
<HTML>

<FORM RUNAT=SERVER>

EDAD.....:<ASP:TEXTBOX ID=EDAD RUNAT=SERVER/><BR>

MESES...:<ASP:LABEL ID=MESES RUNAT=SERVER/><BR>

<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/>

</FORM><BR>

</HTML>

<SCRIPT LANGUAGE=C# RUNAT=SERVER>

void EVENTO1 (Object sender, EventArgs e)

{

// llamando a procedimiento

proc1();

}

void proc1(){

int edad = Int32.Parse(EDAD.Text);

edad=edad*12;

MESES.Text=edad.ToString();

}

</SCRIPT>
```

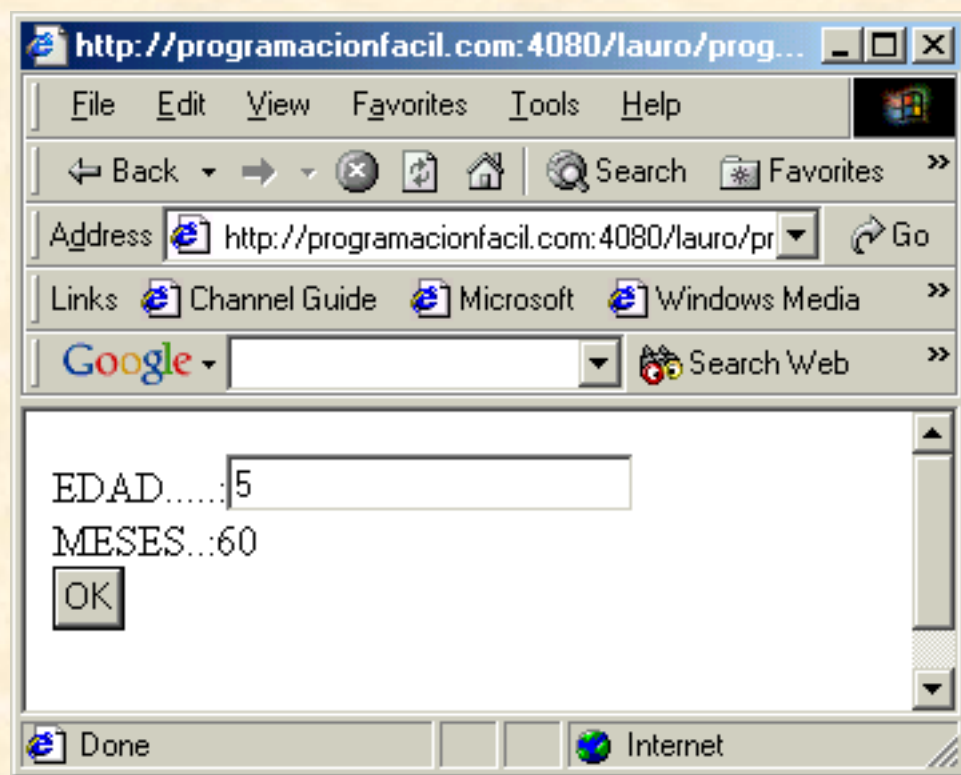
NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Es un script, pero es similar para un cs.

Observar que se puede crear el procedimiento o los procedimientos (aunque realmente es una función) arriba o abajo de la parte principal del programa.

Tambien pueden crearse en sus propio tags `<script> procn() </script>` para mayor claridad del programa.

Corrida prog21.aspx



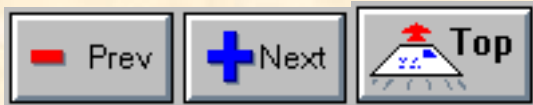
Como se observa un procedimiento puede ser un programa completo.

Construir los siguientes problemas usando procedimientos,

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder

abrirles una cuenta e instrucciones de uso apropiadas.

- a) Convertir \$800.00 Pesos a dolares.
- b) Calcular el Area de un triangulo
- c) Deplegar una Boleta de calificaciones.



UNIDAD 4: PROCEDIMIENTOS Y FUNCIONES

TEMA 2: PARAMETROS

Un parametro es una variable que puede pasar su valor a un procedimiento desde el principal o desde otro procedimiento.

Existen ocasiones en que es necesario mandar al procedimiento ciertos valores para que los use en algún proceso.

Estos valores que se pasan del cuerpo principal del programa o de un procedimiento a otros procedimientos se llaman parametros.

Entonces la declaración completa de un procedimiento es :

```
Void Nom_Proc(lista de parametros)
```

```
{ cuerpo de instrucciones;};
```

Donde lista de parametros es una o mas variables separadas por comas como lo muestra el programa ejemplo.

Prog22.aspx

<HTML>

<FORM RUNAT=SERVER>

EDAD.....:<ASP:TEXTBOX ID=EDAD RUNAT=SERVER/>

MESES...:<ASP:LABEL ID=MESES RUNAT=SERVER/>


```
<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/>
```

```
</FORM><BR>
```

```
</HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
int edad = Int32.Parse(EDAD.Text);
```

```
// llamando a procedimiento y pasando parametro edad
```

```
procl(edad, "juan");
```

```
}
```

```
void procl(int edad1, string nombre){
```

```
edad1=edad1*12;
```

```
MESES.Text=edad1.ToString();
```

```
Response.Write(nombre);
```

```
}
```

```
</SCRIPT>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

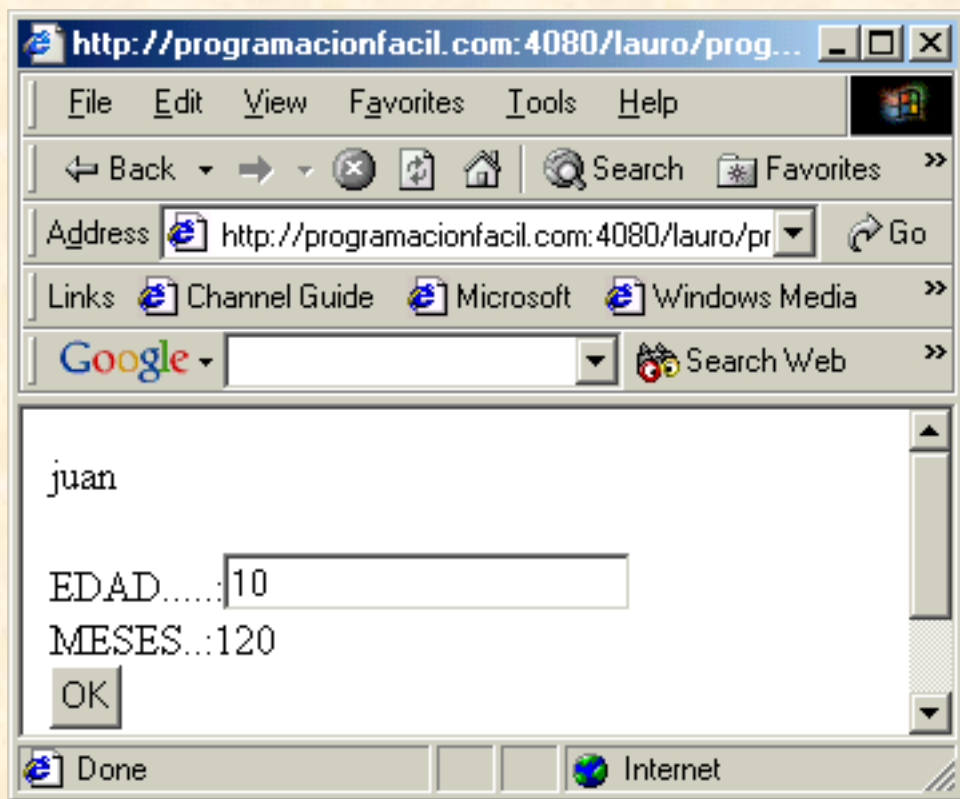
Observar que en el procedimiento los parametros crean dos variables

de manera local, es decir variables que solo pueden usar dentro del procedimiento, estas variables son quienes reciben los datos o valores.

REGLAS PARA EL USO DE PARAMETROS :

- 1.- Cuando se usan variables como parametros la variable que se manda debe ser declarada dentro del principal o del procedimiento de donde se esta enviando.
- 2.- La variable que se manda tiene un nombre, la que se recibe **puede** tener otro nombre o el mismo nombre por claridad de programa, pero recordar que internamente en la memoria del computador existiran dos variables diferentes.
- 3.- La cantidad de variables que se envian deben ser igual en cantidad, orden y tipo a las variables que reciben.
- 4.- La variable que se recibe tiene un ambito local dentro del procedimiento, es decir solo la puede usar ese procedimiento.
- 5.- Se puede mandar a un procedimiento un dato una variable(como lo muestran los ejemplos) o una expresión algebraica (no ecuación o formula) pero siempre se deberan recibir en una variable.

Corrida prog22.aspx



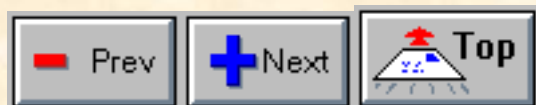
TAREAS

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Una forma activa un programa, recoger 3 calificaciones en el onclick, calcular promedio en procedimiento uno e imprimir nombre y promedio en un segundo procedimiento(aspx).

Construir una tabla de multiplicar que el usuario indique, captura y control de ciclo en el principal, calculo y despliegue en un procedimiento, usar response.write(cs)

Construir un procedimiento que reciba un numero entero y que mande llamar a un segundo procedimiento pasando el letrero "PAR O IMPAR" (aspx y cs).



UNIDAD 4: PROCEDIMIENTOS Y FUNCIONES

TEMA 3: VARIABLES LOCALES Y GLOBALES

El lugar donde sea declarada una variable afectara el uso que el programa quiera hacer de esa variable.

Las reglas basicas que determinan como una variable puede ser usada depende de 3 lugares donde se puede declarar una variable.

En primer lugar es dentro de cualquier función o procedimiento a estas se les llama variables locales y solo pueden ser usadas por instrucciones que esten dentro de esa función o procedimiento.

En segundo lugar es como parametro de una función donde despues de haber recibido el valor podra actuar como variable local en esa función o procedimiento.

En escencia una variable local solo es conocida por el código de esa función o procedimiento y es desconocida por otras funciones o procedimientos.

En tercer lugar es fuera de todas los procedimiento o funciones (que es el caso comun de casi todas las variables usadas hasta ahora en los ejemplos y programas hechos) a este tipo de variables se les llama variables globales y pueden ser usadas por cualquier función o procedimiento del programa.

En programación en serio no es acostumbrado usar muchas variables globales por varias razones, una de ellas es que variables globales estan vivas todo el tiempo de ejecución del programa y si una global solo la ocupa unos cuantos procedimientos no tiene caso que este viva para todo el resto, otra razón es que es peligroso tener variables globales porque todo el conjunto de procedimiento y funciones que

componen un programa tienen acceso o comparten su valor y se corre el riesgo de que inadvertidamente alguno de ellos modifique su valor.

Prog23.aspx

```
<HTML>
```

```
<FORM RUNAT=SERVER>
```

```
<ASP:LABEL ID=RESULTADO RUNAT=SERVER/><BR>
```

```
<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/>
```

```
</FORM><BR>
```

```
</HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
// variable global
```

```
int varuno=50;
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
// variable local
```

```
int vardos=20;
```

```
// llamando a procedimiento y pasando como parametro vardos
```

```
procl(vardos);
```

```
}
```

```
void procl(int vartres){
```

```
// el parametro vartres es tambien local

// observar que procl puede usar varuno porque es local

varuno= varuno + vartres;

RESULTADO.Text=varuno.ToString();

}

</SCRIPT>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

corrida prog23.aspx:



PROBLEMAS SUGERIDOS

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN

PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

- 1.- BOLETA DE CALIFICACIONES Y SOLO USAR DOS VARIABLES GLOBALES
- 2.- UNA TABLA DE MULTIPLICAR Y SOLO USAR UNA VARIABLE GLOBAL



UNIDAD 4: PROCEDIMIENTOS Y FUNCIONES

TEMA 4: FUNCIONES

Una funcion es un modulo de un programa separado del cuerpo principal, que realiza una tarea especifica **y que puede regresar** un valor a la parte principal del programa u otra funcion o procedimiento que la invoque.

La forma general de una funcion es:

```
Tipodato Nomfun(parametros)

{

cuerpo de instrucciones;

return [dato,var,expresion];

}
```

Donde tipo dato especifica el tipo de dato que regresara la función.

La instrucción RETURN es quien regresa un y solo un dato a la parte del programa que la este llamando o invocando, sin embargo es de considerar que RETURN puede regresar un dato, una variable o una expresión algebraica(no ecuación o formula) como lo muestran los siguientes ejemplos;

a) return 3.1416;

b) return area;

c) return x+15/2;

La lista de parametros formales es una lista de variables separadas por comas (,) que almacenaran los valores que reciba la funcion estas variables actuan como locales dentro del cuerpo de la funcion.

Aunque no se ocupen parametros los paréntesis son requeridos.

INSTRUCCION RETURN

Dentro del cuerpo de la función deber haber una instrucción RETURN cuando menos para regresar el valor esta instrucción permite regresar datos.

Recordar ademas que cuando se llame una función debera haber una variable que reciba el valor que regresara la función, es decir generalmente se llama una función mediante una sentencia de asignacion, por ejemplo resultado = funcion(5, 3.1416);

Prog24.aspx

```
<HTML>
```

```
<FORM RUNAT=SERVER>
```

```
<ASP:LABEL ID=RESULTADO RUNAT=SERVER/><BR>
```

```
<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/>
```

```
</FORM><BR>
```

```
</HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)
```

```
{
```

```
// llamando a funcion observar que puede ser por igualdad
```



```
double resultado = sumar(10, 3.1416);  
  
RESULTADO.Text=resultado.ToString();  
  
}
```

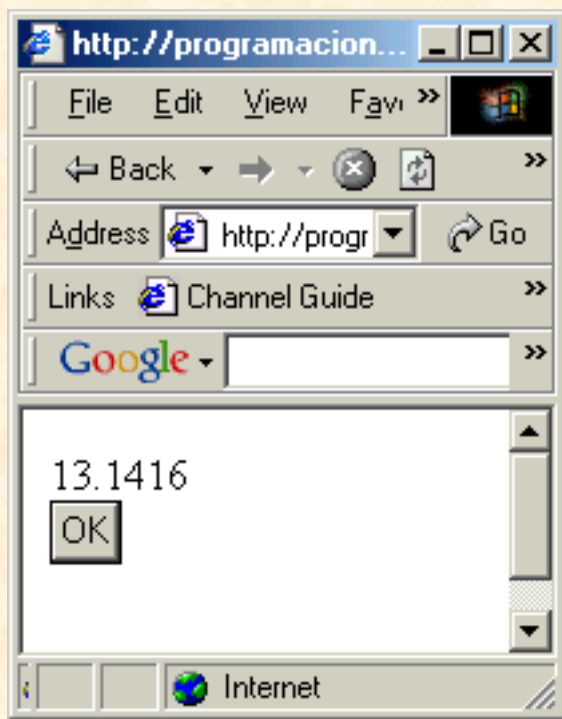
```
double sumar(int alfa, double beta)  
  
{  
  
return alfa + beta;  
  
}
```

```
</SCRIPT>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Usar solo ints y doubles como parametros.

Corrida prog24.aspx



Es permitido poner mas de un return en el cuerpo de instrucciones sobre todo en condiciones pero solo un return se ejecutara ejemplo;

```
if (suma >= 10)
```

```
{ return 10; }
```

```
else
```

```
{ return 20; }
```

EXISTEN 3 CLASES USUALES DE FUNCIONES.

Las primeras son de tipo computacional que son diseñadas para realizar operaciones con los argumentos y regresan un valor basado en el resultado de esa operación.

Las segundas funciones son aquellas que manipulan información y regresan un valor que indican la terminacion o la falla de esa manipulacion.

Las terceras son aquellas que no regresan ningun valor, es decir son estrictamenta procedurales.

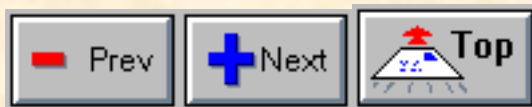
Esto quiere decir que en general toda operación o calculo en un programa debiera convertirse a una o muchas funciones, y el resto deberan ser procedimientos.

TAREAS

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Capturar 3 calificaciones y nombre en un procedimiento, calcular promedio en una funcion, desplegar en otro procedimiento.(aspx)

Crear una tabla de multiplicar, captura y control de ciclo en el principal, operaciones en una funcion, despliegue en el principal.
(cs)



UNIDAD 4: PROCEDIMIENTOS Y FUNCIONES

TEMA 5: ARREGLOS COMO PARAMETROS

Para pasar un arreglo completo como parametro a un procedimiento a una función solo se manda el nombre del arreglo sin corchetes e indices, en el procedimiento o función que recibe solo se declara un arreglo del mismo tipo y se puede usar el mismo o diferente nombre del arreglo sin corchetes e indices.

Sim embargo es conveniente aclarar, que a diferencia de variables escalares normales, csharp no genera una nueva variable en memoria ni tampoco copia los datos al arreglo que recibe, en su lugar csharp sigue usando los datos que estan en el arreglo original, es por esta razón que cambios que se le hagan a los datos del arreglo que recibe realmente se esta haciendo al arreglo original como lo muestra el siguiente ejemplo:

Prog25.aspx

```
<HTML>
```

```
<FORM RUNAT=SERVER>
```

```
<ASP:BUTTON TEXT=OK ONCLICK=EVENTO1 RUNAT=SERVER/>
```

```
</FORM><BR>
```

```
</HTML>
```

```
<SCRIPT LANGUAGE=C# RUNAT=SERVER>
```

```
void EVENTO1 (Object sender, EventArgs e)

{

// creando una lista local

int[] lista= new int[5];

// cargando la lista local con 10,11,12,13,14

for ( int x=0; x<=4; x++) {lista[x]=x+10;};

// pasandola a procedimiento

procl(lista);

//desplegando lista original

for(int x=0; x<=4; x++){Response.Write(lista[x]+"<BR>");};

}


void procl(int[] vector) {

// sumandole 15

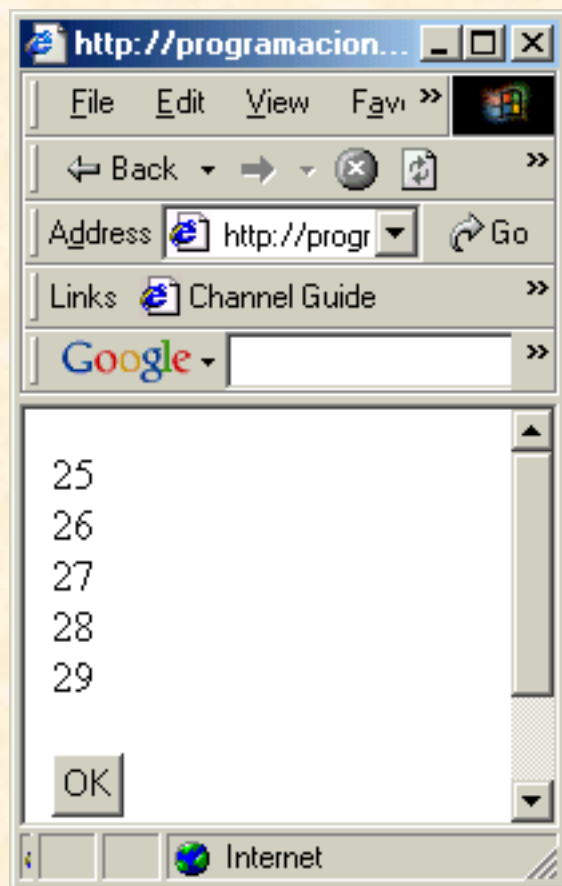
for (int x=0; x<=4; x++){ vector[x]=vector[x]+15;};

}

</SCRIPT>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL. COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

corrida prog25.aspx



Es de recordar que los cambios que le hagan al arreglo dentro de la función se reflejaran en el arreglo original, es por esto que si se quiere modificar un arreglo en una función no hay necesidad de regresar ningún valor.

TAREA

NOTA: A TODAS LAS PERSONAS QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Inicializar 10 edades en el principal mandar la lista a un procedimiento que la convierte a meses, desplegar en principal.

TAREA

Capturar un arreglo de 7 ciudades en un primer procedimiento, sortear en un segundo y desplegar en un tercero, la lista original y la lista

ordenada.



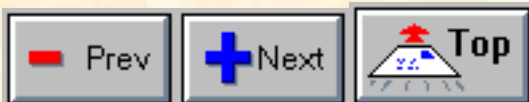
UNIDAD 5: INT A LAS BASES DE DATOS

TEMA 1: INTRODUCCIÓN

En este capitulo se analizan en general dos problemas:

- a) Variables que permitan almacenar conjuntos de datos como los arreglos pero con distintos tipos de datos este primer problema se resolvía en la antigüedad usando las llamadas variables registro.
- b) Permanencia de los datos hasta ahora todos los datos capturados, calculados, creados, etc. al terminar o cerrarse el programa se pierden y es necesario volver a capturarlos en la siguiente ejecución o corrida del programa.

Tradicionalmente en programación antigua este segundo problema se resolvía usando el concepto de archivos que son medios permanentes de almacenamiento de datos en los dispositivos o periféricos apropiados generalmente disco, cinta magnética, etc.



WWW.PROGRAMACIONFACIL.COM

UNIDAD 5: INT A LAS BASES DE DATOS

TEMA 2: MODELOS DE ALMACENAMIENTO DE DATOS

En general existen dos modelos de almacenamiento de datos en los sistemas de información.

a) El modelo tradicional de archivos que se construye con los siguientes elementos:

1.- Variables Registros que como ya se indico son variables que permiten almacenar conjuntos de datos de diverso tipo.

También se pueden definir como representaciones simbólicas y programáticas de entidades lógicas de información ejemplos de variables registros son alumnos, empleados, clientes, proveedores, productos, autos, etc.

Estas variables registros también ocupan programas o rutinas de programas para procesarlas por ejemplo un procedimiento, modulo o subrutina se encargara de capturar los datos que contendrá la variable registro otro procedimiento para corregir los datos que ya contiene, otro procedimiento para desplegarlos en pantalla ya cuando ha sido capturada y así sucesivamente.

2.-Archivos, que en principio pueden entenderse como una especie de almacenes o bodegas para almacenamiento de datos en forma permanente en disco es decir, un archivo de empleados en disco contiene todos los datos de todos los empleados de una empresa.

Igualmente los archivos ocupan su propios programas o subrutinas o procedimientos especializados por ejemplo, procedimientos para crear los archivos, para almacenar o dar de altas los registros en el archivo, procedimientos para buscar un registro determinado,

procedimiento para dar de baja un registro, etc.

3.- Una aplicación que es un programa que se encarga de coordinar todos los programas descritos y presentar a usuarios de manera clara, fácil, accesible y entendible.

Salta a la vista que construir un sistema de información por ejemplo para una tienda de vídeo o para un refaccionaria, etcetera, involucra un gran cantidad de trabajo de programación puesto que hay que programar muchas variables registros, muchos archivos en disco y construir una o muchas aplicaciones.

Este modelo se usa todavía en la actualidad pero es obvio que mejores maneras, mas rápidas, seguras y eficientes existen en la actualidad para resolver estos problemas, y esto nos lleva al segundo modelo de datos.

b) Modelo de Bases de Datos Relacionales: este modelo intenta simplificar la construcción de sistemas de información como los antes descritos, este modelo solo incluye en forma simple los siguientes elementos:

b.1) Tablas que son una combinación de las variables registro y de los archivos del modelo anterior.

Es decir cuando un programador moderno define o declara una tabla en un programa realmente esta haciendo dos cosas por el precio de una es decir crea una variable registro en memoria que almacenara los datos y al mismo tiempo ya esta creando un archivo en disco que se llamara igual que la tabla y que automáticamente se convertirá en un espejo de la tabla en memoria.

Otra vez cuando el programador escribe código para capturar los datos y mandarlos a la tabla en pantalla-memoria, realmente también lo esta haciendo para darlos de alta en disco.

b.2) Aplicación, que tiene la misma función que en el modelo anterior.

No confundir este concepto de tablas en base de datos con el concepto de tablas vistos en el capitulo de arreglos.

Como se observa en este modelo es mas sencillo construir sistemas de información puesto que la parte programática se reduce ampliamente.



UNIDAD 5: INT A LAS BASES DE DATOS

TEMA 3: TABLAS

Una Tabla simple representa una unidad de información de una entidad física o lógica que pueda ser sujeta a un proceso de información:

ej:

Tabla Empleado:

- Clave Empleado
 - Nombre Empleado
 - Dirección Empleado
 - Edad Empleado
 - Teléfono Empleado
 - etc. Empleado
-

Tabla Proveedor:

- Clave Proveedor
 - Nombre Proveedor
 - Empresa Proveedor
 - Teléfono Proveedor
 - Fax Proveedor
 - Celular Proveedor
 - etc. Proveedor
-

Tabla Autos:

- Numero de Serie
- Modelo
- Marca

- Tipo
- Color
- Capacidad
- etc.

REGLAS:

Observar que cada tabla, empieza con una clave generalmente de tipo numérica.

Todos los elementos de la tabla solo hacen referencia hacia el mismo ente o sujeto de información.

Cada elemento solo representa o debe contener un y solo un dato de información.

No se respetan o siguen al pie de la letra estos tres postulados **y empiezan los problemas al tiempo de programación.**

- Existe una segunda forma o manera de representar las tablas, ejemplo:

Tabla: Camisas

| NUMCAMISA | MARCA | ESTILO | MEDIDA | COLOR | MATERIAL |
|-----------|----------|----------|---------|--------|-----------|
| 1 | JEANS | SPORT | GRANDE | AZUL | ALGODON |
| 2 | VOLIS | VESTIR | MEDIANA | NEGRA | POLIESTER |
| 3 | GENERICA | CAMISETA | LARGA | MORADO | RARON |

Tabla: Clientes

| NUMCLIENTE | NOMCLIENTE | DIRCLIENTE | TELCLIENTE |
|------------|--------------|-----------------|------------|
| 1 | JUAN PEREZ | AV ABA 2233 | 2345678 |
| 2 | LUIS SANCHEZ | CALLE ZETA 3434 | 4567899 |
| 3 | ROSA MARES | CALLEJON NORTE | 567890 |

Recordar siempre una tabla almacena o representa un conjunto de datos del mismo tipo o entidad, la tabla de alumnos es para almacenar y manipular muchos alumnos, la tabla de productos es para almacenar y manipular muchos alumnos, en resumen si en un problema de información solo se presenta una instancia o renglón de una entidad lógica, entonces no es tabla, es un encabezado.

PROBLEMAS SUGERIDOS

1.-CONSTRUIR EN CUADERNO LAS SIGUIENTES TABLAS, LA MITAD DE ELLAS CON EL PRIMER FORMATO Y LA SEGUNDA MITAD CON EL SEGUNDO FORMATO.

1.- PACIENTES

2.- PERROS

3.- PLUMAS

4.- MERCANCÍAS

5.- PELÍCULAS

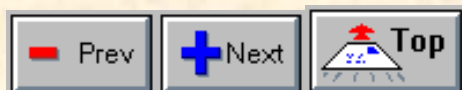
6.- MEDICINAS

7.- MAESTROS

8.- MATERIAS

9.- COMPUTADORAS

10.- BANCOS



UNIDAD 5: INT A LAS BASES DE DATOS

TEMA 4: TABLAS (CONTINUACIÓN)

El trabajo **correcto** con bases de datos relacionales se divide en dos grandes pasos o etapas bien diferenciadas entre si:

En la primera etapa se diseña la tabla, con sus campos, llaves y condiciones especiales, luego se usa un paquete o programa de software especializado en la construcción, mantenimiento y administración de la base de datos, este software se usa para convertir la tabla o tablas ya bien diseñadas en un archivo en disco.

Existe un primer tipo de software especializado en bases de datos, los llamados **servidores de bases de datos**, los tres mas comunes son SQL-SERVER de Microsoft, ORACLE Server de Oracle, MYSQL Open Source, en estos casos la base de datos(o conjunto de tablas que tienen relaciones comunes entre si), residen en un servidor de bases de datos especializado en algun lugar cercano o lejano en una red chica, mediana o grande.

Otros paquetes o software reciben el nombre de DBMS(DATA BASE MANAGEMENT SYSTEM) o sistemas administradores de bases de datos.

Este tipo de software se especializa en la creación, mantenimiento, seguridad, privacidad, etc. de un conjunto de tablas o mejor dicho una base de datos, los DBMS mas comunes son access, postgres, fox, clipper, etc.

Usaremos Microsoft Access como nuestro generador de bases de datos, y recordar que una base de datos es en principio un conjunto de tablas que tienen y mantienen relaciones entre si.

La segunda etapa consiste en construir la aplicación o aplicaciones que ya tendrán acceso o podrán manipular los datos contenidos en la

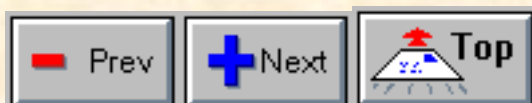
tabla, estas aplicaciones se escriben usando ya sea lenguajes clásicos de programación como BASIC, PASCAL, COBOL, CBUILDER, DELPHI, JAVA, VBSCRIPT, PERL, JSCRIPT, CSHARP, etc.

DISEÑO Y CREACIÓN DE UNA TABLA

El primer paso antes de usar el paquete correspondiente a esta tarea, es diseñar la tabla completamente, esto exige:

- a) Nombre apropiado y determinación de atributos y campos correspondientes.
- b) Seleccionar y determinar el atributo principal o campo clave o llave primaria que se utilizara como el identificador único que permite diferenciar cada instancia o renglón diferente dentro de la tabla.
- c) También se puede seleccionar otros campos que puedan servir mas adelante para ordenar de manera diferente la tabla, es decir una tabla en principio ya está ordenada por campo clave por ejemplo, la matricula de un alumno, el numero de empleado, etc., pero existirán muchas ocasiones donde se puede pedir un orden diferente, por ejemplo, por ciudad, por carrera, por nombre, por edad, etc., la buena ingeniería de bases de datos exige tomar en cuenta estos y otros muchos problemas y detalles.
- d) A estos atributos o campos especiales se les conoce como claves o llaves secundarias, que internamente generan otra tabla especial llamada tabla o archivo de índices, (tabla o archivo que contiene dos campos, el primero es la clave secundaria ordenada y el segundo la posición o renglón donde se encuentra en la tabla original).
- e) Escribir restricciones y condiciones apropiadas para ciertos atributos, por ejemplo el número de empleado deben comenzar en 500, la edad no debe ser mayor de 150 años, etc.

Ya listo el diseño de la tabla, se usara el programa correspondiente para su creación y almacenamiento en este caso Microsoft Access.



UNIDAD 5: INTRODUCCION A LAS BASES DE DATOS

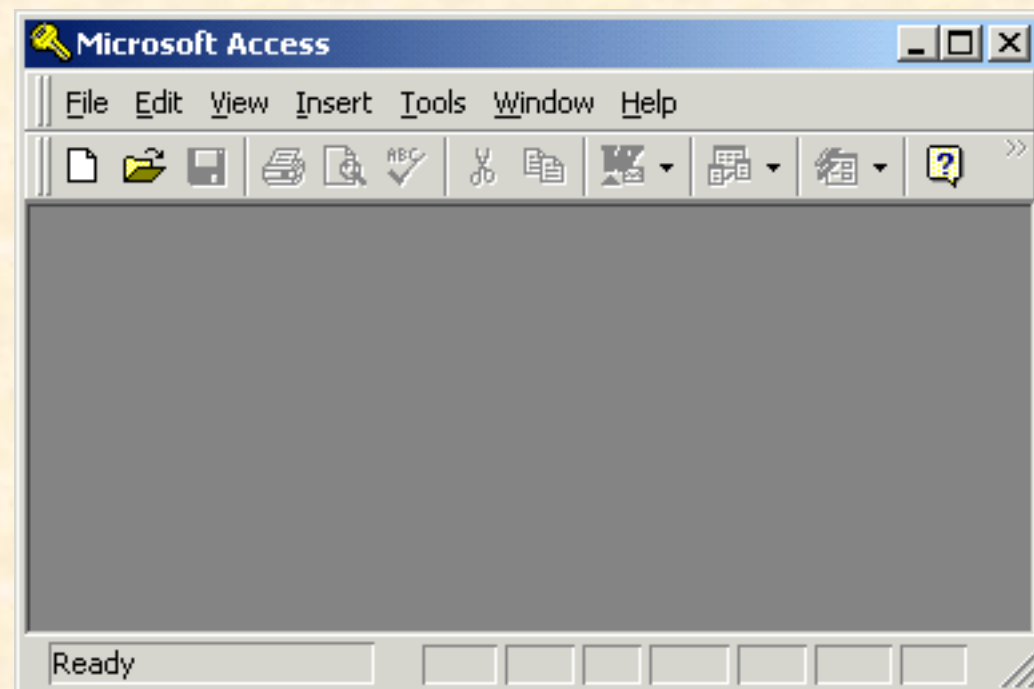
TEMA 5: ACCESS

En este ejercicio construiremos una base de datos llamada **mibase** que solo contendrá una tabla llamada **mitabla** con tres campos que son clave, nombre y edad, que se estarán usando a lo largo de esta unidad a manera de ejemplo.

Se usa access97 en virtud de que es el más sencillo de todas las versiones aunque pueden usar cualquier versión que tengan sin embargo solo se responde por access97.

PROCEDIMIENTO:

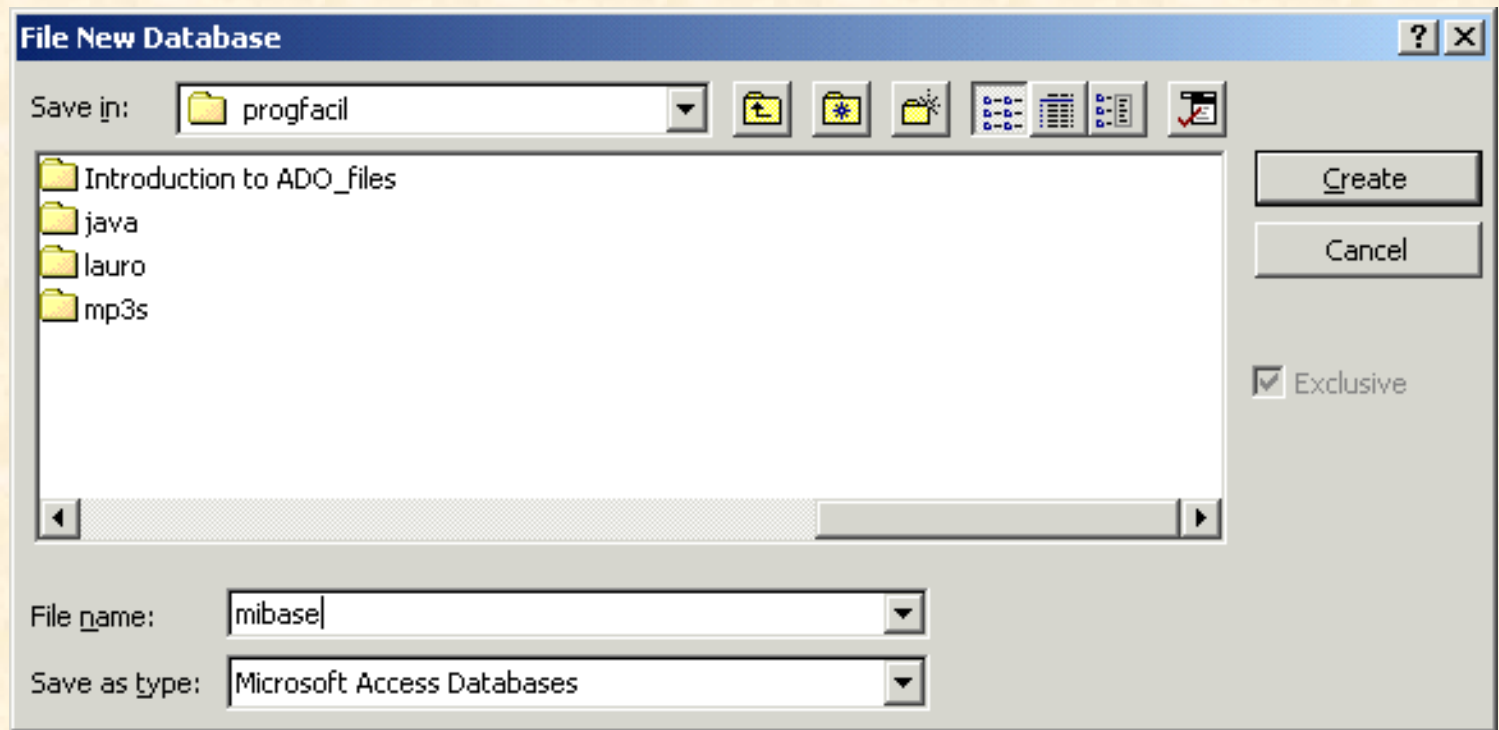
1.- Cargar Access y sale la siguiente pantalla:



2.- Usar la opción FILE-->NEW DATABASE y seleccionar de la pantalla

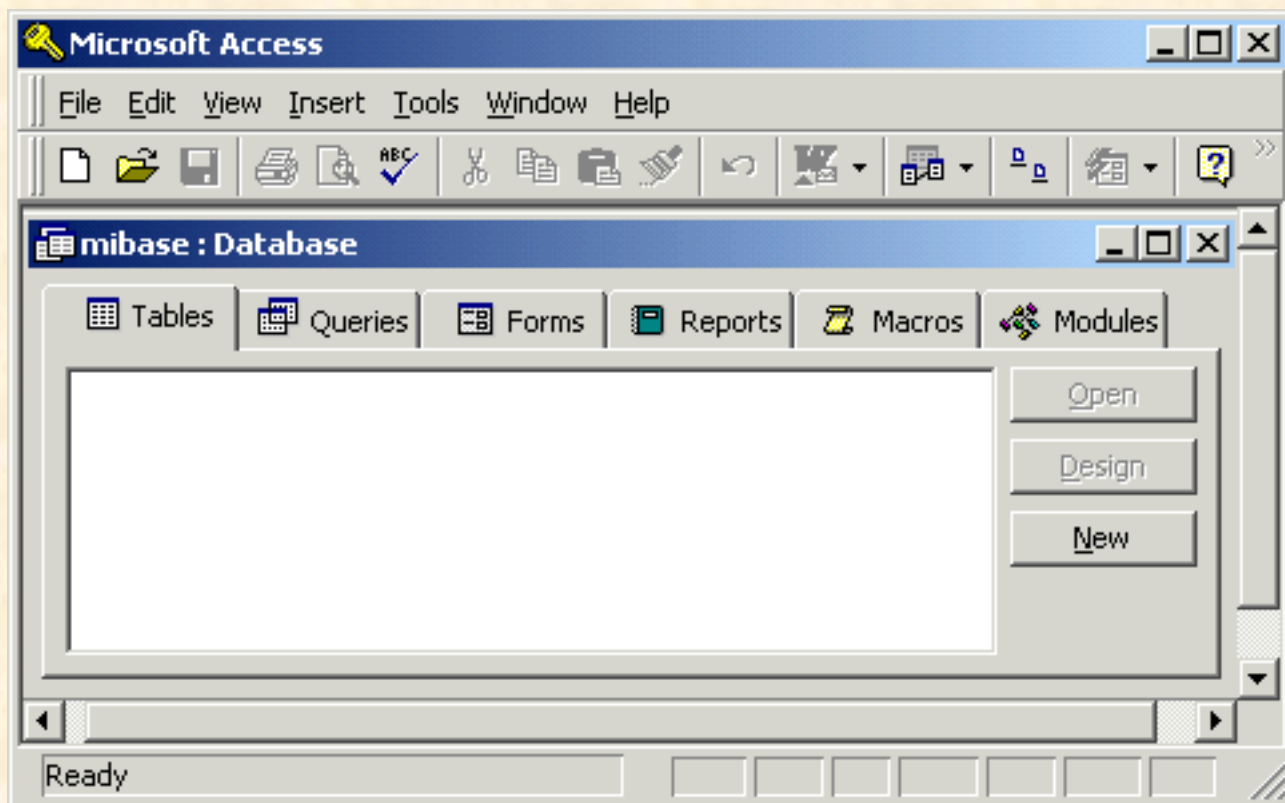
que sale BLANK DATABASE.

3.- Inmediatamente ACCESS pregunta donde se almacenara y como se llamara la base de datos usando la pantalla normal de grabación de archivos:

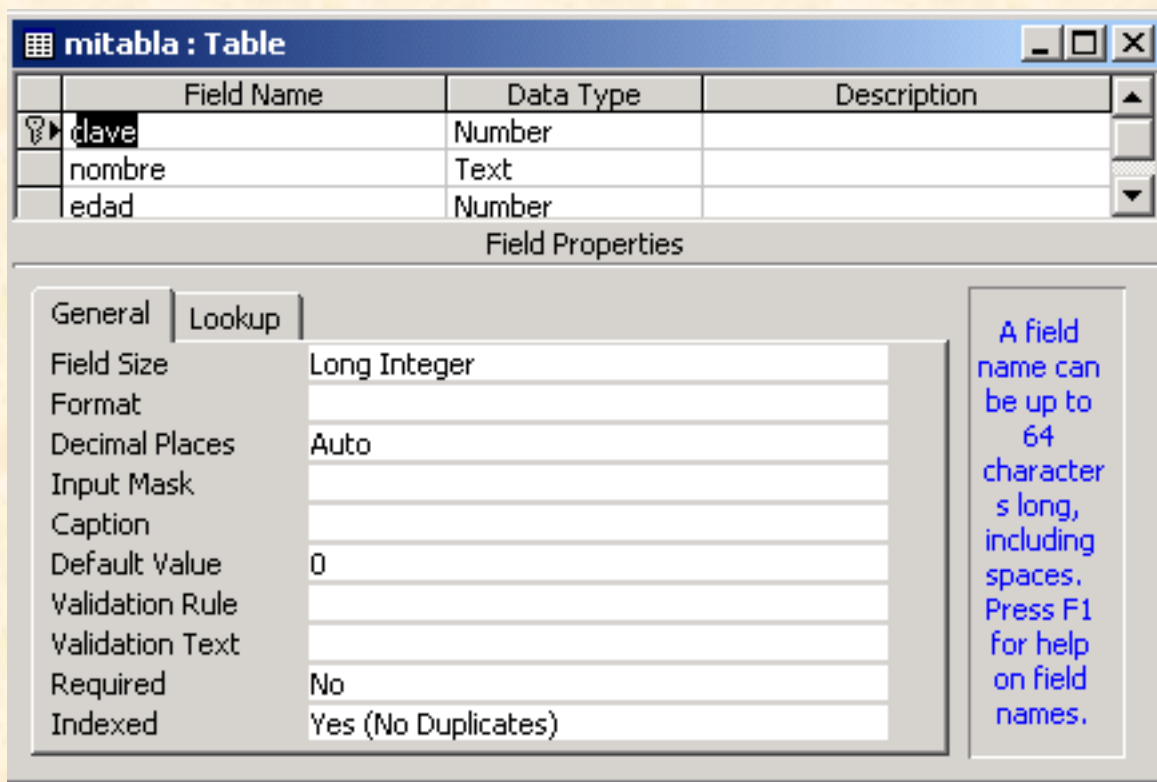


4.- Ponerla en un lugar o folder adecuado y para este ejemplo llamarla **mibase** (como se ve en la pantalla de arriba), luego usar el boton create.

5.- Aparece ahora la siguiente pantalla:



6.- Esta ultima pantalla permite construir una o mas tablas que contendra la base de datos (mibase), observar que tambien permite agregar mas elementos a una base de datos (querys, forms, reports, etc), pero para este ejercicio solo se agrega una tabla a la base de datos, para crear **mitabla** solo usar boton new y access ofrecera construirla de varias maneras distintas de preferencia seleccionar la manera DESIGN VIEW que manda la siguiente pantalla:

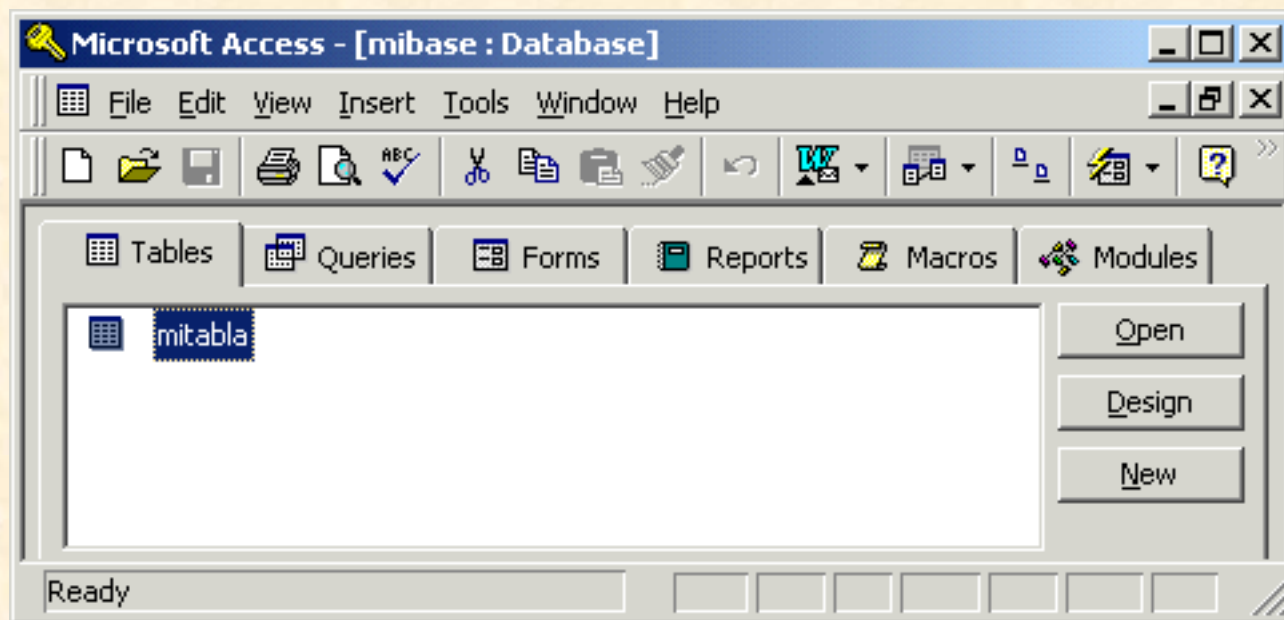


7.- En FIELD NAME escribir el nombre del campo, en DATA TYPE solo hacer click y salen opciones de los diversos tipos de datos que ofrece access, en DESCRIPTION se puede poner una descripción de los propósitos del campo, para el ejemplo que se está mostrando se usa number para la clave, texto con tamaño 30 caracteres (seleccionar abajo) para nombre y number para edad.

8.- Observar que CLAVE tiene una pequeña llave a un lado, esto significa que CLAVE es la llave primaria de la tabla, para marcarla como llave primaria primero seleccionar todo el renglón haciendo un click en el cuadrado gris que está antes de la palabra clave y luego hacer un click derecho y del minimenu que sale usar opción primary key, es muy importante que el tipo de dato de la clave sea de tipo **NUMBER**.

9.- Ahora cerrar la tabla usando la [x] de arriba y access pregunta como se llamara la tabla, llamarla **mitabla**.

10.- Ahora se regresa a la vista de diseño de access y ya estará registrada mitabla en mibase, como lo muestra el siguiente gráfico:



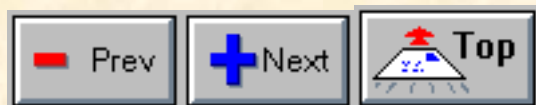
11.- Usar ahora el boton OPEN, para cargar unos cuantos datos o renglones de prueba como lo demuestra el siguiente ejemplo:

| mitabla : Table | | | |
|-----------------|-------|--------|------|
| | clave | nombre | edad |
| ▶ | 1 | oso | 7 |
| | 2 | peach | 8 |
| | 3 | raton | 2 |
| | 4 | pato | 5 |
| | 5 | perico | 10 |
| * | 0 | | 0 |
| Record: 1 of 5 | | | |

12.- Cerrar access con la [x] de arriba y si dios quiere ya tenemos construida MIBASE.MDB que a su vez contiene MITABLA que a su vez contiene unos cuantos renglones de datos.

13.- El ultimo paso es subir a tu sitio mibase.mdb para que ya este lista y preparada para procesarla con ASP.NET y CSHARP.

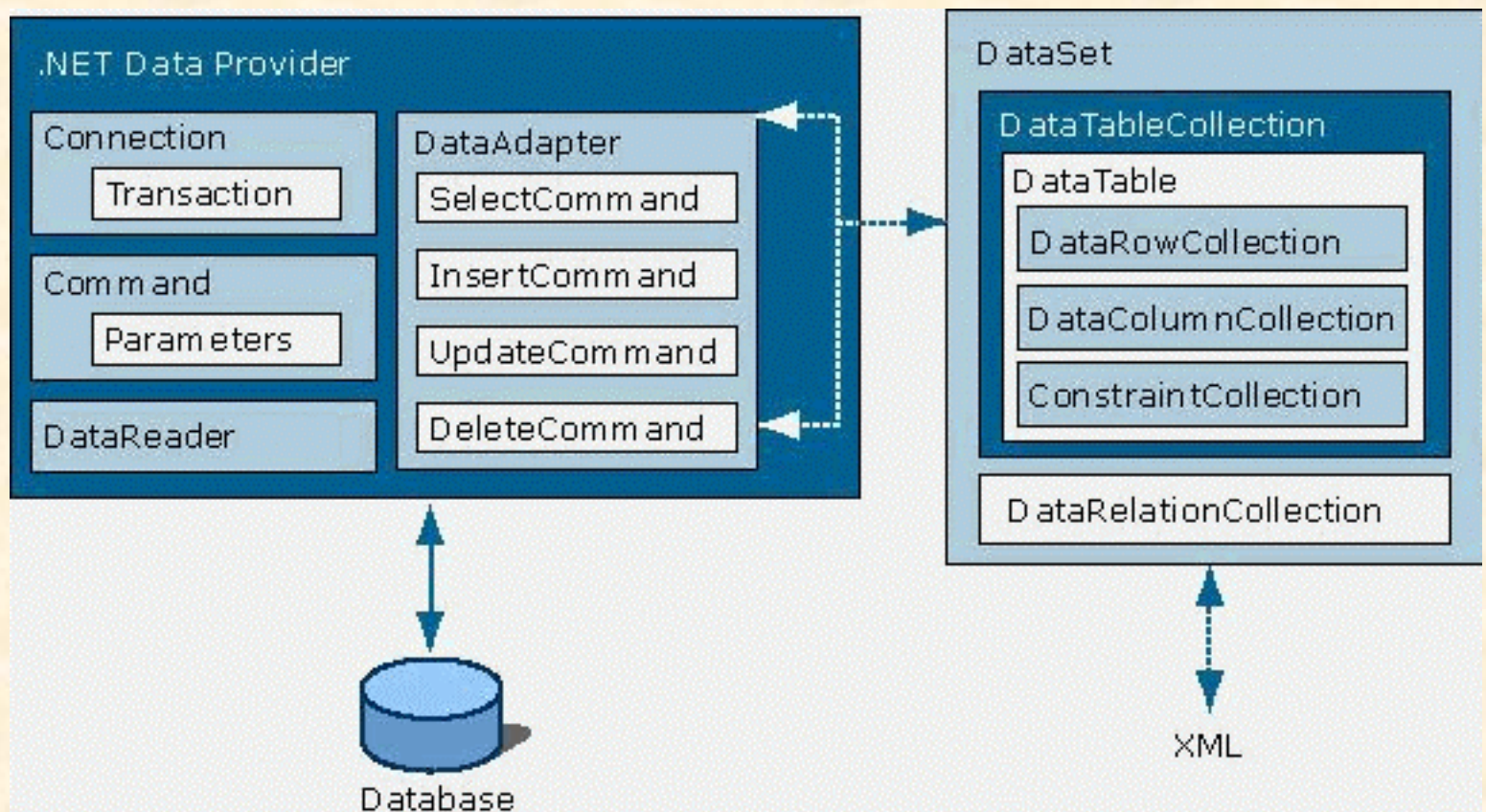
NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.



UNIDAD 5: INTRODUCCION A LAS BASES DE DATOS

TEMA 6: ADO NET ACTIVE DATA OBJECT

EL NUEVO MODELO DE DATOS DE MICROSOFT ES ADO.NET, ESTE MODELO DESCANSA EN UNA SERIE DE OBJETOS ESPECIALIZADOS QUE FACILITAN EL PROCESAMIENTO DE UNA BASE DE DATOS.



fuelle microsoft.net

Como ven esta bastante claro y no ocupa explicación, :-)

Empezando:

El problema es comunicar un programa o aplicación aspx con una base

de datos y mas que comunicar se pretende que el programa o aplicación realice una serie de procesos u operaciones con la base de datos o mejor aun con el conjunto de tablas que contiene una base de datos.

La primera nota a recordar es que una base de datos puede estar fisicamente en el servidor y en algun folder o directorio del disco duro de dicha maquina servidora, por ejemplo, c:\progfacil\misitio\mibase.mbd, como se observa la base que se construyo en access (mibase.mbd) se almaceno en el disco c en el folder progfacil y dentro del subfolder misitio.

Sin embargo tambien es necesario conocer que asi como existen servidores de paginas(web server), servidores de correo (mail server), servidores de ftp (ftp server), etc, tambien existen servidores de bases de datos (database server), los mas comunes son el sqlserver de microsoft, oracle, mysql, etc, estos servidores tambien pueden crear, administrar y procesar una base de datos, por supuesto que el procedimiento que se dio para crearla en access en el tema anterior no se puede usar para crear y cargar una base de datos en un servidor de bases de datos.(esperar libros de bases de datos en programacionfacil en un proximo futuro).

El modo de comunicarse entre nuestro programa o aplicación aspx y la base de datos (ya sea fisica o un dbserver), implica que ambos manejen un lenguaje de programación comun, es decir no se puede mandar una instrucción en csharp, o en basic o pascal a la base de datos y ademas esperar que esta ultima la entienda (para entender esto, una razon muy sencilla es que la base de datos tendria que conocer o comprender todos los lenguajes de programación), para resolver este problema de comunicación es que se usa un lenguaje comun de bases de datos que tanto los lenguajes de programación existentes como las bases de datos entienden, este lenguaje comun de bases de datos es el **SQL** (structured query language) o lenguaje estructurado de consultas.

En otras palabras, ustedes mis estimados lectores tendran que aprender este nuevo lenguaje de programación el SQL, la buena noticia es que es un lenguaje con muy pocas instrucciones y ademas existen muy buenos tutoriales en internet que hay que buscar y estudiar.

PROPAGANDA, ya ven que en este curso estan aprendiendo el lenguaje de programación CSHARP, el lenguaje de programación HTML y ahora el

lenguaje de programación SQL, !wow! tres lenguajes por el precio de uno.

Bueno las principales instrucciones de SQL, que se usan en este curso son SELECT, INSERT, UPDATE y DELETE.

La pregunta es ahora como mandamos las instrucciones sql a la base de datos, la respuesta son los **OBJETOS ADO.NET** que estamos analizando en orden y proposito de uso, los estaremos explicando.

OBJETO CONNECTION:- OBJETO QUE SE UTILIZA PARA ESTABLECER UNA CONECCION O ENLACE A LA BASE DE DATOS.

Este objeto primero se tendra que crear en el programa y luego se tendra que cargar con dos parametros(ver ejemplo mas abajo), el primer parametro es el proveedor o la fuente que proporcionara los datos, los proveedores o fuentes de datos que existen son:

SQLSERVER NET DATA PROVIDER.- QUE SE ESPECIALIZA EN COMUNICARSE Y PROCESAR BASES DE DATOS CONSTRUIDAS CON MICROSOFT SQL SERVER V7.0

OLEDB.NET DATA PROVIDER.- QUE SE ESPECIALIZA EN COMUNICARSE Y PROCESAR BASES DE DATOS QUE A LA FECHA DEL PRESENTE LIBRO UTILIZEN ALGUNOS DE LOS SIGUIENTES DRIVERS, SQLOLEDB (VERSIONES ANTERIORES DE SQL SERVER DE MICROSOFT), MSDAORA (ORACLE), MICROSOFT.JET (ACCESS Y ALGUNOS OTROS DBMS DE MICROSOFT)

ODBC.NET .- BASES DE DATOS QUE USAN ODBC COMO MEDIO DE COMUNICACION CON OTRAS BASES DE DATOS Y APLICACIONES COMO NOTA A CONSIDERAR ODBC. NET NO ESTA INCLUIDA POR DEFAULT EN MICROSOFT.NET, SE TIENE QUE BAJAR DE MICROSOFT.

El segundo parametro es la propia base de datos con la cual se comunicara el programa o aplicación.

Ejemplo del objeto CONNECTION

```
Static OleDbConnection coneccion;
```

```
conexcion = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;  
Data Source=c:\\prografil\\tusitio\\mibase.mdb");
```

Es una sola string y los dos parametros mencionados van separados por el punto y coma.

ATENCIÓN es DATA SOURCE= no usar DATASOURCE= estan advertidos.

ejemplos de los otros proveedores o fuentes mencionados:

```
//Provider=MSDAORA; Data Source=ORACLE8i7; User ID=OLEDB; Password=OLEDB  
  
//Provider=Microsoft.Jet.OLEDB.4.0; Data Source=c:\\bin\\LocalAccess40.mdb;  
  
//Provider=SQLOLEDB;Data Source=MySQLServer;Integrated Security=SSPI;
```

OBJETO COMMAND.-

Ya establecido el canal o enlace entre el programa aspx y la base de datos via el objeto CONEXCION, se debe mandar la instruccion SQL a la propia base de datos, sin embargo en un programa de csharp por supuesto que no puede contener instrucciones de otros lenguajes de programación como el de SQL, es por esto que se deberan usar algunos de los otros objetos de ADO.NET para que estos objetos transporten la instruccion sql hacia la base de datos (y transporte de regreso al servidor los datos de alguna tabla), uno de estos objetos es el objeto COMMAND.

Este objeto puede contener directamente una instrucción SQL y enviarla al objeto conexión ya descrito.

Este objeto command primero se tendra que crear y luego cargarle dos parametros que son:

la instrucción sql y el objeto conexcion que ya se vio en el parrafo anterior. ejemplo

```
OleDbCommand orden;  
  
orden= new OleDbCommand("select * from mitabla", conexion);
```

Si esta muy grande o muy compleja la instruccion sql, es mas conveniente crearla en una variable string y poner la variable como parametro ejemplo:

```
OleDbCommand orden;  
  
String q="select * from mitabla";  
  
orden= new OleDbCommand(q, conexion);
```

Sin embargo ciertas instrucciones de sql (ya estudiaron su tutorial del sql??), requieren que se manden los datos a la base de datos, respetando el tipo de dato con los cuales los creo el software de bases de datos, por ejemplo si edad en access se declaro como NUMBER, la instruccion sql que prentenda cargar dicho campo, tiene la obligación de mandarla con este tipo de dato asociado, instrucciones SQL que permiten cargar o capturar ese campo edad son INSERT o UPADTE (ya estudiarón su tutorial de SQL??).

Para resolver este problema, usaremos en la string q, unas variables especiales llamadas VARIABLES PARAMETROS que se simbolizan usando el simbolo @ antes de la variable y ademas al objeto COMMAND le agregamos dos instrucciones extras que permiten agregar a la string q el dato y el tipo de dato, ejemplo, se tienen seis renglones ya capturados en nuestra tabla y se quiere agregar un septimo renglon con los siguientes datos, clave=7, nombre="rana" peso=3.14 usaremos una instrucción SQL INSERT ej:

```
OleDbCommand orden;

String clave=7;string nombre="rana"; string peso=3.14;

string q="insert into mitabla(clave,nombre,peso) values(@CLAVE, @NOMBRE,
@PESO)";

orden= new OleDbCommand(q, coneccion);

orden.Parameters.Add(new OleDbParameter("@CLAVE", OleDbType.Integer));

orden.Parameters["@CLAVE"].Value = clave;

orden.Parameters.Add(new OleDbParameter("@NOMBRE", OleDbType.VarWChar, 40));

orden.Parameters["@NOMBRE"].Value = nombre;

orden.Parameters.Add(new OleDbParameter("@PESO", OleDbType.Double));

orden.Parameters["@PESO"].Value = edad;
```

Observar que para cada variable parametro, se tienen que cargar dos elementos, el valor y el tipo de dato correspondiente.

Aunque en valor se manda string's en oledbtype se hace un mapeo, relación o conversión al tipo de dato que se uso en access, tener mucho cuidado que exista una relación igual o cuando este programa se ejecute el servidor les va a mandar un error o excepción de sql que les intenta decir que el tipo de dato que mandaron a la base de datos, no es igual al que se uso para crearlo en la base de datos.

Los OLEDBTYPE mas comunes son:

BigInt A 64-bit signed integer (DBTYPE_I8). This maps to Int64.

Binary A stream of binary data (DBTYPE_BYTES). This maps to an Array of type Byte.

Boolean A Boolean value (DBTYPE_BOOL). This maps to Boolean.

BSTR A null-terminated character string of Unicode characters (DBTYPE_BSTR). This maps to String.

Char A character string (DBTYPE_STR). This maps to String.

Currency A currency value ranging from -2^{63} (or -922,337,203,685,477.5808) to $2^{63} - 1$ (or +922,337,203,685,477.5807) with an accuracy to a ten-thousandth of a currency unit (DBTYPE_CY). This maps to Decimal.

Date Date data, stored as a double (DBTYPE_DATE). The whole portion is the number of days since December 30, 1899, while the fractional portion is a fraction of a day. This maps to DateTime.

DBDate Date data in the format yyyyymmdd (DBTYPE_DBDATE). This maps to DateTime.

DBTime Time data in the format hhmmss (DBTYPE_DBTIME). This maps to TimeSpan.

DBTimeStamp Data and time data in the format yyyyymmddhhmmss (DBTYPE_DBTIMESTAMP). This maps to DateTime.

Decimal A fixed precision and scale numeric value between $-10^{38} - 1$ and $10^{38} - 1$ (DBTYPE_DECIMAL). This maps to Decimal.

Double A floating point number within the range of $-1.79E + 308$ through $1.79E + 308$ (DBTYPE_R8). This maps to Double.

Empty No value (DBTYPE_EMPTY). This maps to Empty.

Error A 32-bit error code (DBTYPE_ERROR). This maps to Exception.

Filetime A 64-bit unsigned integer representing the number of 100-nanosecond intervals since January 1, 1601 (DBTYPE_FILETIME). This maps to DateTime.

Guid A globally unique identifier (or GUID) (DBTYPE_GUID). This maps to Guid.

IDispatch A pointer to an IDispatch interface (DBTYPE_IDISPATCH). This maps to Object. Note This data type is not currently supported by ADO.NET. Usage may cause unpredictable results.

Integer A 32-bit signed integer (DBTYPE_I4). This maps to Int32.

IUnknown A pointer to an IUnknown interface (DBTYPE_UNKNOWN). This maps to Object. Note This data type is not currently supported by ADO.NET. Usage may cause unpredictable results.

LongVarBinary A long binary value (OleDbParameter only). This maps to an Array of type Byte.

LongVarChar A long string value (OleDbParameter only). This maps to String.

LongVarChar A long null-terminated Unicode string value (OleDbParameter only). This maps to String. **Numeric** An exact numeric value with a fixed precision and scale (DBTYPE_NUMERIC). This maps to Decimal. **PropVariant** An automation PROPVARIANT (DBTYPE_PROP_VARIANT). This maps to Object.

Single A floating point number within the range of -3.40E +38 through 3.40E +38 (DBTYPE_R4). This maps to Single.

SmallInt A 16-bit signed integer (DBTYPE_I2). This maps to Int16.

TinyInt A 8-bit signed integer (DBTYPE_I1). This maps to SByte.

UnsignedBigInt A 64-bit unsigned integer (DBTYPE_UI8). This maps to UInt64.

UnsignedInt A 32-bit unsigned integer (DBTYPE_UI4). This maps to UInt32.

UnsignedSmallInt A 16-bit unsigned integer (DBTYPE_UI2). This maps to UInt16.

UnsignedTinyInt A 8-bit unsigned integer (DBTYPE_UI1). This maps to Byte.

VarBinary A variable-length stream of binary data (OleDbParameter only). This maps to an Array of type Byte. **VarChar** A variable-length stream of non-Unicode characters (OleDbParameter only). This maps to String.

Variant A special data type that can contain numeric, string, binary, or date data, as well as the special values Empty and Null (DBTYPE_VARIANT). This type is assumed if no other is specified. This maps to Object.

VarNumeric A variable-length numeric value (OleDbParameter only). This maps to Decimal.

VarWChar A variable-length, null-terminated stream of Unicode characters (OleDbParameter only). This maps to String.

WChar A null-terminated stream of Unicode characters (DBTYPE_WSTR). This maps to String.

Fuente:microsoft.net

Aun mas, con el ejemplo anterior el objeto COMMAND esta construido y preparado y cargado pero todavia no se manda desde el programa a la base de datos, es decir le faltan activar las siguientes tres propiedades, ejemplo;

```
OleDbCommand orden;
```

```
String clave=7;string nombre="rana";string peso=3.14;
```

```
string q="insert into mitabla(clave,nombre,peso) values(@CLAVE, @NOMBRE, @PESO)";
```

```
orden= new OleDbCommand(q, coneccion);
```

```
orden.Parameters.Add(new OleDbParameter("@CLAVE", OleDbType.Integer));
```

```
orden.Parameters["@CLAVE"].Value = clave;
```

```
orden.Parameters.Add(new OleDbParameter("@NOMBRE", OleDbType.VarWChar, 40));
```

```
orden.Parameters["@NOMBRE"].Value = nombre;
```

```
orden.Parameters.Add(new OleDbParameter("@PESO", OleDbType.Double));
```

```
orden.Parameters["@PESO"].Value = edad;
```

```
orden.Connection.Open();
```

```
orden.ExecuteNonQuery();
```

```
orden.Connection.Close()
```

sencillo abrir la coneccion, mandar o ejecutar la instrucción y

cerrar la coneccion.

OBJETOS DATAADAPTER Y DATASET:(dos por uno)

Son los otros dos objetos de ADO.NET que tambien permiten transportar una instruccion sql desde el servidor hasta la base de datos y transportar de regreso hacia el servidor los datos contenidos en alguna de las tablas .

Con los objetos CONNECTION y COMMAND ya se pueden efectuar cualquiera de la operaciones SQL descritas(ya estudiarón su tutorial de SQL), el problema es que pasa con el usuario cuando va a ver base de datos o mejor aun las tablas que estan en la base de datos en disco.

DATASET:- Es una copia en memoria (d la maquina cliente) de la base de datos(y todas sus tablas) que se encuentra en disco.

DATAADAPTER.- En principio es muy similar al objeto COMMAND es decir se usa para transportar instrucciones SQL a la base en disco, de hechos sus formatos e instrucciones son muy similares a los vistos para el objeto COMMAND, **su diferencia principal es que dataadapter esta mas especializado y contiene una serie de metodos que facilitan la intereaccion entre el DATASET y la Base de Datos en disco**

En particular muchos de los programas que se veran en temas posteriores solo usan los objetos CONNECTION, DATAADAPTER y DATASET.

Otra vez, dataadpater se especializa en transportar instrucciones sql a la base de datos en disco pero ademas se utiliza para cargar la tabla en memoria o dataset del cliente.

Ejemplo:

```
// abriendo la coneccion
```

```
coneccion = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=c:\\progfacil\\tusitio\\mibase.mdb");
```

```
// cargando el adapter con la instruccion sql

canal=new OleDbDataAdapter("select * from mitabla", conexion);

// cargando el dataset

tabla= new DataSet();

canal.Fill(tabla, "mitabla");
```

Como se observa en este ejemplo muy sencillito, el dataadapter(canal) esta funcionando de manera muy similar al primer ejemplo que se vio del objeto COMMAND pero tengan la seguridad que tambien se pueden usar variables parametros y agregarles los dos tipos de parametros a este objeto dataadpater.

Observar que su propiedad FILL carga el DATASET(tabla) con una de las tablas en disco, recordar que en la base de datos puede contener muchas tablas.

Ademas esa propiedad FILL es equivalente a las tres ultimas instrucciones del objeto COMMAND, es decir open, executenonquery y close, mas facil verdad.

DATAREADER y DATASET:

Observar que tambien se usan en forma conjunta, primero es muy similar en uso y función que el objeto DATAADAPTER, la diferencia entre datareader y dataadapter es el tipo de base de datos con las cuales se pueden comunicar, dataadpater se especializan en bases de datos relacionales y datareader se especializa en archivos, que no se estudian en este curso.

Tambien es importante mencionar que datareader es el objeto de ADO.NET mas parecido al objeto RESULTSET que uso mucho en el ADO anterior de microsoft.

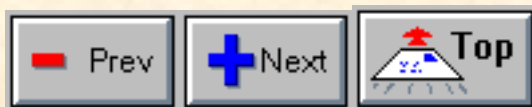
EN general se han visto de manera sencilla los principales objetos ADO.ASP(connection, command, datareader, dataadapter, dataset), sin embargo la tabla o las tablas o la base de datos que se tiene en disco o sirviendola algun servidor de bases de datos, se ha quedado en la memoria de la maquina del cliente, ADO.NET ha terminado su trabajo y su función.

Para mandar el dataset a el browser se tendra que pasar a algun tipo de objeto visible que soporte el browser, los objetos que se pueden usar para mandar el dataset a pantalla son:

1.- COMPONENTE TABLE DE HTML(USADO EN EL CURSO DE CSHARP-CGI DE PROGRAMACIONFACIL.COM)

2.- COMPONENTE HTMLTABLE DE ASP

3.- COMPONENTE DATAGRID DE ASP.NET(USADO EN EL CURSO DE CSHARP-NET DE PROGRAMACIONFACIL.COM)



UNIDAD 5: INTRODUCCION A LAS BASES DE DATOS

TEMA 7: CONSULTA O DESPLIEGUE O SELECCION

Existen una serie de operaciones y procesos que son muy comunes contra una tabla en una base de datos en disco la mas comun es desplegar todos los renglones de la tabla que estan almacenados en disco, a este proceso le llamaremos SELECCION, consulta o despliegue (muy original).

Como se indico anteriormente la comunicación con la base de datos se tendran que dar usando el lenguaje especializado de bases de datos llamado SQL(structured query language), la instrucción sql que se usa para resolver este problema tiene el siguiente formato:

```
SELECT [listacampos, * o ALL] FROM TABLA;
```

El procedimiento que se intenta seguir cuando se construya un programa asp.net que tenga que manipular una tabla en disco debera seguir los siguientes pasos:

-
- 1.- Crear una conección o enlace a la base de datos.
 - 2.- Abrir la conección a la base de datos.
 - 3.- Crear el enlace o adapter y cargarlo con la instruccion sql (o cargar primero la instruccion sql en un objeto command y mandarlo a travez del adapter)
 - 4.- Crear el dataset y cargarlo a travez del adapter
 - 5.- Cargar el DataGridView con el dataset y enlazarlo(binding)

6.- Procesar el datagrid (editar un renglon, agregar un renglon, modificar un renglon, etc)

7.- Cerrar la conexión

Codigo prog26.aspx

```
<%@ PAGE LANGUAGE=C# %>

<%@ Import Namespace="System" %>

<%@ Import Namespace="System.Data" %>

<%@ Import Namespace="System.Data.OleDb" %>

<FORM RUNAT=SERVER>

<ASP:DATAGRID ID=TABLAGRID RUNAT=SERVER

Width=400

BackColor=#ccccff

BorderColor=black

ShowFooter=false

CellPadding=3

CellSpacing=0

Font-Name=Verdana

Font-Size=8pt

HeaderStyle-BackColor=#aaaadd

EnableViewState=false
```

```
</>

</FORM>

<script runat=server>

// creando y cargando coneccion, adpater, dataset como variables globales

OleDbConnection coneccion;

DataSet tabla;

OleDbDataAdapter canal;

void Page_Load(object sender, EventArgs e)

{

//enlazando coneccion a la base de datos

coneccion = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=c:\\prografacil\\tusitio\\mibase.mdb");

// recordar espacio en blanco en DATA SOURCE=

// cargando el adapter con la instruccion sql

canal = new OleDbDataAdapter("select * from mitabla", coneccion);

// cargando el dataset

tabla= new DataSet();

canal.Fill(tabla, "mitabla");

// cargando el datagrid

TABLAGRID.DataSource=tabla;

TABLAGRID.DataMember="mitabla";
```

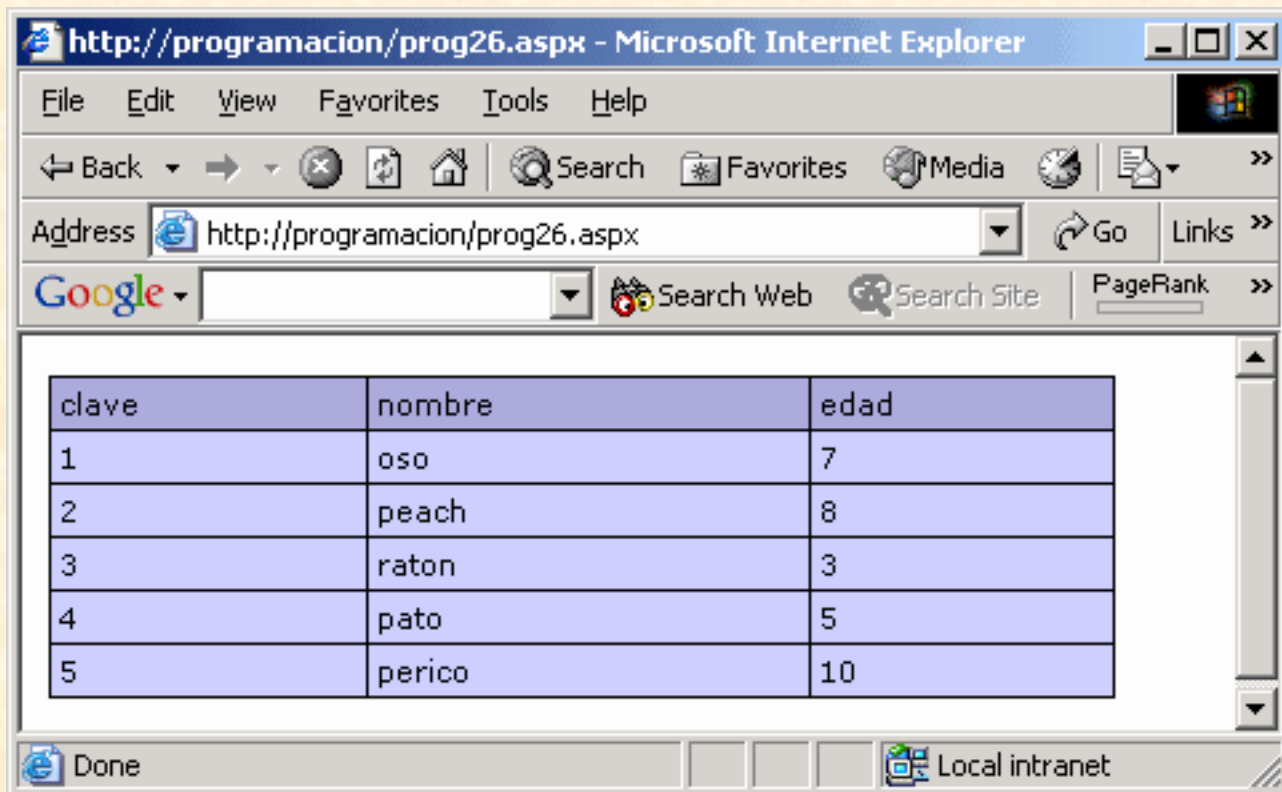
```
TABLAGRID.DataBind();
```

```
}
```

```
</script>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

corrida prog26.aspx



notas:

- 1.- Se sigue el procedimiento generico para procesar tablas usando ADO.NET
- 2.- Observar y siempre incluir los namespaces indicados.
- 3.- Se usa una PAGINA (page) y su evento `pageload()` para desplegar la

tabla(mitabla) que esta en la base de datos(mibase), se puede cargar e inicializar usando un objeto button=desplegar con este mismo codigo cargado en su evento onclick, pero es criterio de ustedes como se despliega la tabla.

4.- Recordar que DATAGRID es un WEBCONTROL por tanto hay que crearlo e inicializarlo al principio del programa, tambien recordar que datagrid tiene muchas propiedades que le mejoran la interfase con que se despliega y es en esta parte donde se cargan dichas propiedades.

5.- Se empieza creando las variables globales a ocupar y abriendo la conección a la base de datos, si se les hace muy grande la string del proveedor, pueden cargarla primero en una variable string y carguen la string en el constructor de la conección, pero esto es opcional.

5.1) Recordar que hay otros proveedores de bases de datos para cuando se quieran accesar bases de datos diferentes de access.

6.- Tomar nota como se hace una referencia a la base de datos, esto es en c:\progfacil\tusitio\base.mdb (ojo con las diagonales)

7.- Se crea el adapter y se carga el constructor con la instrucción sql y la coneccion, aqui es necesario entender que existen varias maneras de hacer esto:

Cargar una string con el sql y crear y usar un objeto command directamente por ejemplo objeto comand(tringsql); y luego todavia se tendria que ejecutar con executenonquery(que ejecuta una string que no regresa datos por ejemplo insert o update para un adapter) o executereader(si en lugar de usar adapter se usa un reader) o executescalar(metodo que regresa un solo dato de la base de datos)

usar algunas de las funciones descritas del adapter por ejemplo adpatercommandselect(stringsql)

pero lo mas sencillo fue usar el metodo que se puso en el programa, es decir crear el adapter y pasarle directamente la instrucción sql.

8.- Luego se creo el dataset y se cargo con toda la base de datos en disco, entender esto bien, dataset puede quedar cargado con todas las tablas que tenga la base de datos por eso se usa un FILL para pasar al dataset solo una de las tablas(mi tabla), esto da origen a dos

notas:

8.1.- al programar mas adelante se ocupara explicitamente indicarle al compilador con cual tabla se va a trabajar, es por esta razón que se veran instrucciones tales como `tabla.tables["clientes"].etc.etc.` aqui se esta diciendo al compilador que del dataset(TABLA) se va a realizar una proceso con la tabla de clientes.

8.2.- Para procesar dos o mas tablas, entonces se tendra que usar mucho el formato que se vio en la nota 8.1

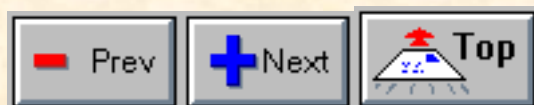
9.- Al final se carga el datagrid, se pega(binding) al dataset y se cierra la base de datos.

Problemas sugeridos:

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- construir y desplegar una primera base de datos que contenga la primera tabla que diseñarón en el tema de tablas.

2.- Construir una segunda base de datos que contenga cuando menos tres de las tablas ya diseñadas y desplegar cualquiera de ellas usando una sola forma html, donde el usuario selecciona cual quiere desplegar.



UNIDAD 5: INTRODUCCIÓN A LAS BASES DE DATOS

TEMA 8: INSERCIÓN O ADICIÓN DE REGISTROS

Insertar o agregar registros o renglones nuevos a una tabla en disco, es un proceso sencillo que usa la siguiente instrucción sql:

```
INSERT INTO TABLA(CAMPO1,CAMPO2..) VALUES(VALOR1,VALOR2..);
```

Recordar que solo se está usando lo mínimo de cada instrucción sql, es conveniente estudiar un tutorial de sql.

Prog27.aspx

```
<%@ PAGE LANGUAGE=C# %>
```

```
<%@ Import Namespace="System" %>
```

```
<%@ Import Namespace="System.Data" %>
```

```
<%@ Import Namespace="System.Data.OleDb" %>
```

```
<FORM RUNAT=SERVER>
```

```
CLAVE<ASP:TEXTBOX ID=CLAVE SIZE=3 RUNAT=SERVER/>
```

```
NOMBRE<ASP:TEXTBOX ID=NOMBRE SIZE=10 RUNAT=SERVER/>
```

```
EDAD<ASP:TEXTBOX ID=EDAD SIZE=3 RUNAT=SERVER/>
```

```
<ASP:BUTTON ONCLICK=INSERTAR TEXT=INSERTAR RUNAT=SERVER /><BR>
```

```
<ASP:DATAGRID ID=TABLAGRID RUNAT=SERVER
```

```
Width=400
```

```
BackColor=#ccccff
```

```
BorderColor=black
```

```
ShowFooter=false
```

```
CellPadding=3
```

```
CellSpacing=0
```

```
Font-Name=Verdana
```

```
Font-Size=8pt
```

```
HeaderStyle-BackColor=#aaaadd
```

```
EnableViewState=false
```

```
/>
```

```
</FORM>
```

```
<script runat=server>
```

```
// creando y cargando coneccion, adpater, dataset como variables globales
```

```
OleDbConnection coneccion;
```

```
DataSet tabla;
```

```
OleDbDataAdapter canal;
```

```
void Page_Load(object sender, EventArgs e)
```

```
{
```

```
coneccion=new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data  
Source=c:\\prografil\\lauro\\mibase.mdb");
```

```
canal=new OleDbDataAdapter("select * from mitabla", conexion);

tabla= new DataSet();

canal.Fill(tabla, "mitabla");

TABLAGRID.DataSource=tabla;

TABLAGRID.DataMember="mitabla";

TABLAGRID.DataBind();

//cargando el nuevo textbox con la nueva clave clave correspondiente

int cren=tabla.Tables["mitabla"].Rows.Count;

int nvaclave=Int32.Parse(tabla.Tables["mitabla"].Rows[cren-1][0].ToString())
+1;

CLAVE.Text=nvaclave.ToString();

}

void INSERTAR (Object sender, EventArgs e)

{

// creando y cargando un objeto OLEDBCOMMAND

// instruccion sql insert into mitabla(listacampos) values(listadatos)

// @variable es una variable de tipo parametro

string q="insert into mitabla(clave,nombre,edad) values(@CLAVE, @NOMBRE,
@EDAD) ";

OleDbCommand orden= new OleDbCommand(q, conexion);

orden.Parameters.Add(new OleDbParameter("@CLAVE", OleDbType.Integer));

orden.Parameters["@CLAVE"].Value = CLAVE.Text;
```

```
orden.Parameters.Add(new OleDbParameter("@NOMBRE", OleDbType.VarWChar, 20));

orden.Parameters["@NOMBRE"].Value = NOMBRE.Text;

orden.Parameters.Add(new OleDbParameter("@EDAD", OleDbType.Integer));

orden.Parameters["@EDAD"].Value = EDAD.Text;

orden.Connection.Open();

orden.ExecuteNonQuery();

orden.Connection.Close();

// REFRESCANDO DATASET con los nuevos datos de la tabla en disco

canal=new OleDbDataAdapter("select * from mitabla", coneccion);

// creando el dataset y cargandolo

tabla= new DataSet();

canal.Fill(tabla, "mitabla");

// cargando el datagrid

TABLAGRID.DataSource=tabla.Tables["mitabla"].DefaultView;

TABLAGRID.DataBind();

// cargando otra vez la caja de CLAVE y limpiando las otras cajas

int cren=tabla.Tables["mitabla"].Rows.Count;

int nvaclave=Int32.Parse(tabla.Tables["mitabla"].Rows[cren-1][0].ToString())
+1;

CLAVE.Text=nvaclave.ToString();

NOMBRE.Text=" ";

EDAD.Text=" ";
```



```
conexion.Close();
```

```
}
```

```
</script>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

corrida prog27.aspx

CLAVE 8 NOMBRE EDAD

INSERTAR

| clave | nombre | edad |
|-------|--------|------|
| 1 | oso | 7 |
| 2 | peach | 8 |
| 3 | raton | 3 |
| 4 | pato | 5 |
| 5 | perico | 10 |
| 6 | gato | 7 |
| 7 | burro | 4 |

notas:

Se agregaron tres textboxes arriba del datagrid para capturar los

nuevos datos a insertar en la tabla.

En `page_Load` es el mismo código del programa anterior solo al final se usa el método `row.count` de `dataset.tables` [recuerdan la nota 8 del tema anterior] para conocer cuantos renglones tiene la tabla

Con esta información ya se puede leer la primera columna (la cero 0 desde luego) para sacar el dato de la ultima clave

Luego se lee el ultimo renglon de la tabla con el método `dataset.tables[0].rows[reng][col]`

Es importante que se entienda que con este formato `dataset.tables[0].rows[reng][col]` se puede **leer o cargar(GET-SET)** cualquier celda o columna de la tabla

Se desconta uno de la cantidad de renglones porque en tablas el primer renglon es el(???? Cero ????se acuerdan)

El método devolvió el valor de la ultima clave que esta en el ultimo renglon de la tabla, pero en string

Al final se incremento en uno la variable entera para obtener el valor de la nueva clave o clave siguiente, misma que se cargo en el textbox correspondiente

En función `INSERTAR()`, se crea la string `q` con el formato apropiado `sql` (como se dijo al principio de este tema), observar que existen tres variables que llevan un `@` antes, estas variables se llaman `VARIABLES PARAMETROS`, y se cargan con el objeto `command.parameters()`

Otra vez, en este ejemplo para mandar la instrucción `sql` a la base de datos se crea y se usa un objeto `command` (llamado `orden`) que lleva como datos la string `q` y la conexión, pero se agregan tres métodos `command.parametro (orden.parameters())`, en estos métodos se cargan las variables `parametro` primero con el valor de dato del textbox `asp` y luego se transforman al tipo de dato apropiado usando los `oledbtype` (que hay que estudiar porque se tienen que asociar directamente a los tipos de datos que se usaron en access)

Ya con el objeto `COMMAND(orden)` listo y cargado para comunicar la instrucción `sql` a la base de datos se abre la conexión a la base de

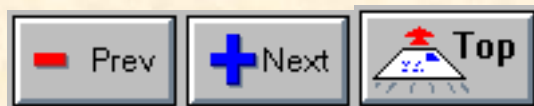
datos se manda el `executenonquery` (no se quiere regresar nada en esta parte, recordar la nota respectiva que se dio en un tema anterior) y se cierra la conexión y si Dios quiere ya se mandó el nuevo renglón a la base de datos en disco.

Al final de esta función `INSERTAR`, como ya se hizo un cambio en la base de datos, se tiene que volver a cargar el dataset con la nueva información (es el mismo código que se tiene en `page_load`) más tantito código para limpiar y cargar los textboxes.

Problemas sugeridos:

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUÍ EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- construir muchos programas de inserción en las tablas de las bases de datos que tengan construidas



UNIDAD 5: INTRODUCCION A LAS BASES DE DATOS

TEMA 9: BUSQUEDA

En este tema se analiza la busqueda de un registro o renglón determinado en este proceso el usuario del programa quiere que se despliegue un y solo un registro de información proporcionando un dato de busqueda generalmente la clave del registro.

La solucion es sencilla, solo usar otra vez la instruccion select, con el siguiente formato:

```
SELECT [ *, all, campos] FROM TABLA WHERE clave=claveabuscar;
```

Se recuerda que deben buscar y estudiar un buen tutorial de sql.

Codigo prog28.aspx

```
<%@ PAGE LANGUAGE=C# %>
```

```
<%@ Import Namespace="System" %>
```

```
<%@ Import Namespace="System.Data" %>
```

```
<%@ Import Namespace="System.Data.OleDb" %>
```

```
<FORM RUNAT=SERVER>
```

```
CLAVE A BUSCAR<ASP:TEXTBOX ID=CLAVE SIZE=3 RUNAT=SERVER/>
```

```
<ASP:BUTTON ONCLICK=BUSCAR TEXT=BUSCAR RUNAT=SERVER /><BR>
```

```
<ASP:DATAGRID ID=TABLAGRID RUNAT=SERVER
```

```
Width=400
```

```
BackColor=#ccccff
```

```
BorderColor=black
```

```
ShowFooter=false
```

```
CellPadding=3
```

```
CellSpacing=0
```

```
Font-Name=Verdana
```

```
Font-Size=8pt
```

```
HeaderStyle-BackColor=#aaaadd
```

```
EnableViewState=false
```

```
/>
```

```
</FORM>
```

```
<script runat=server>
```

```
// creando y cargando coneccion, adpater, dataset como variables globales
```

```
OleDbConnection coneccion;
```

```
DataSet tabla;
```

```
OleDbDataAdapter canal;
```

```
void BUSCAR (Object sender, EventArgs e)
```

```
{
```

```
coneccion=new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data  
Source=c:\\prografacil\\lauro\\mibase.mdb");
```



```
string q="select * from mitabla where clave = @CLAVE";

canal=new OleDbDataAdapter(q, coneccion);

canal.SelectCommand.Parameters.Add(new OleDbParameter("@CLAVE", OleDbType.
Integer));

canal.SelectCommand.Parameters["@CLAVE"].Value = CLAVE.Text;

// creando el dataset y cargandolo

DataSet tabla= new DataSet();

canal.Fill(tabla, "mitabla");

// cargando el datagrid

TABLAGRID.DataSource=tabla;

TABLAGRID.DataMember="mitabla";

TABLAGRID.DataBind();

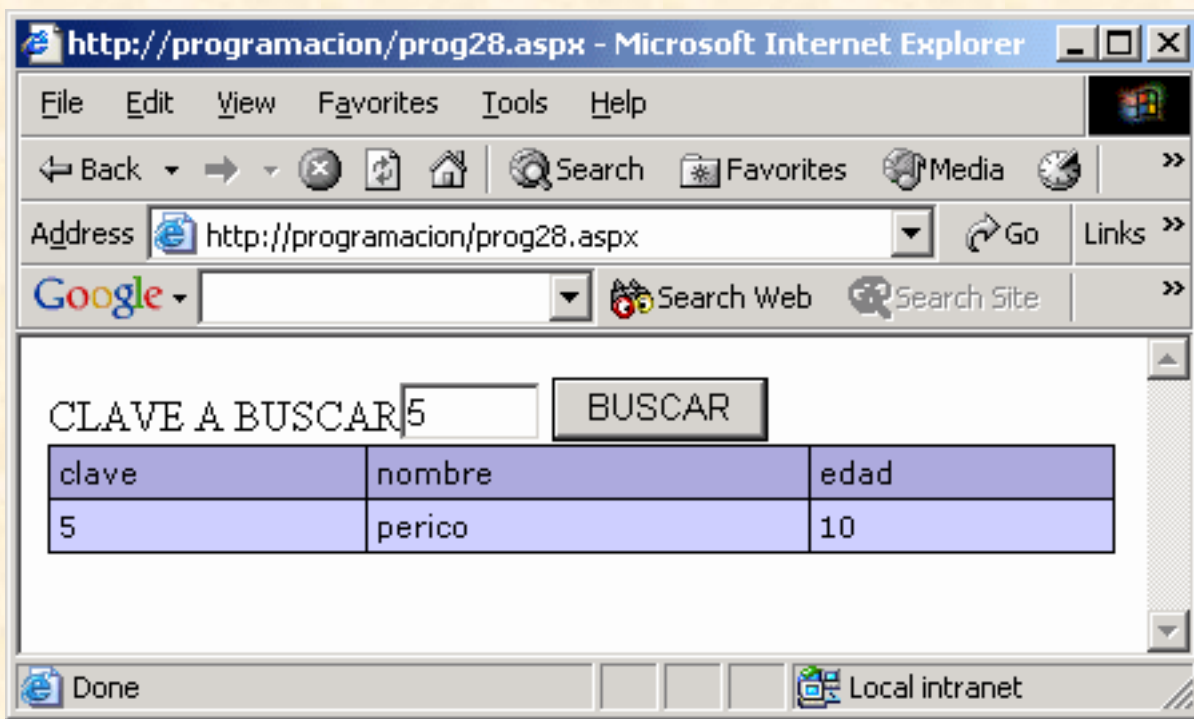
}

</script>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

nota: no hay nada nuevo es una combinación de los dos programas anteriores con las mismas notas, solo se usa un textbox asp para pedir la clave, aunque se puede usar cualquier campo para buscar.

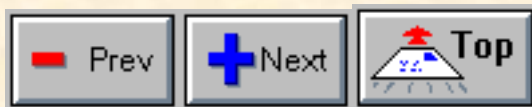
Corrida prog28.aspx



Problemas sugeridos:

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- hacer programas de busquedas para las bases hechas



UNIDAD 5: INTRODUCCION A LAS BASES DE DATOS

TEMA 10: FILTROS

Otro problema similar al anterior es el de filtros es decir en muchas ocasiones es necesario obtener información acerca de un subconjunto de renglones de la tabla.

Por ejemplo todos los estudiantes que sean mayores de 17 años, todos los clientes que sean de Tijuana, etc., a esto le llamamos filtros o condiciones.

Tambien se resuelve de manera similar al anterior, es decir usando la instrucción **select etc, from tabla, where CONDICION;**

Codigo prog29.aspx

```
<%@ PAGE LANGUAGE=C# %>

<%@ Import Namespace="System" %>

<%@ Import Namespace="System.Data" %>

<%@ Import Namespace="System.Data.OleDb" %>

<FORM RUNAT=SERVER>

EDAD MAYOR QUE<ASP:TEXTBOX ID=EDAD SIZE=3 RUNAT=SERVER/>

<ASP:BUTTON ONCLICK=FILTRAR TEXT=FILTRAR RUNAT=SERVER /><BR>

<ASP:DATAGRID ID=TABLAGRID RUNAT=SERVER
```

Width=400

BackColor=#ccccff

BorderColor=black

ShowFooter=false

CellPadding=3

CellSpacing=0

Font-Name=Verdana

Font-Size=8pt

HeaderStyle-BackColor=#aaaadd

EnableViewState=false

/>

</FORM>

<script runat=server>

OleDbConnection coneccion;

DataSet tabla;

OleDbDataAdapter canal;

void FILTRAR (Object sender, EventArgs e)

{

coneccion=new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=c:\\prografacil\\lauro\\mibase.mdb");

string q="select * from mitabla where edad >= @EDAD";

```
// creando el canal o adapter y cargandolo con la instruccion sql

// recordar que tambien se puede usar el objeto command pero es mas largo

OleDbDataAdapter canal=new OleDbDataAdapter(q, coneccion);

canal.SelectCommand.Parameters.Add(new OleDbParameter("@EDAD", OleDbType.
Integer));

canal.SelectCommand.Parameters["@EDAD"].Value = EDAD.Text;

// creando el dataset y cargandolo

DataSet tabla= new DataSet();

canal.Fill(tabla, "mitabla");

// cargando el datagrid

TABLAGRID.DataSource=tabla;

TABLAGRID.DataMember="mitabla";

TABLAGRID.DataBind();

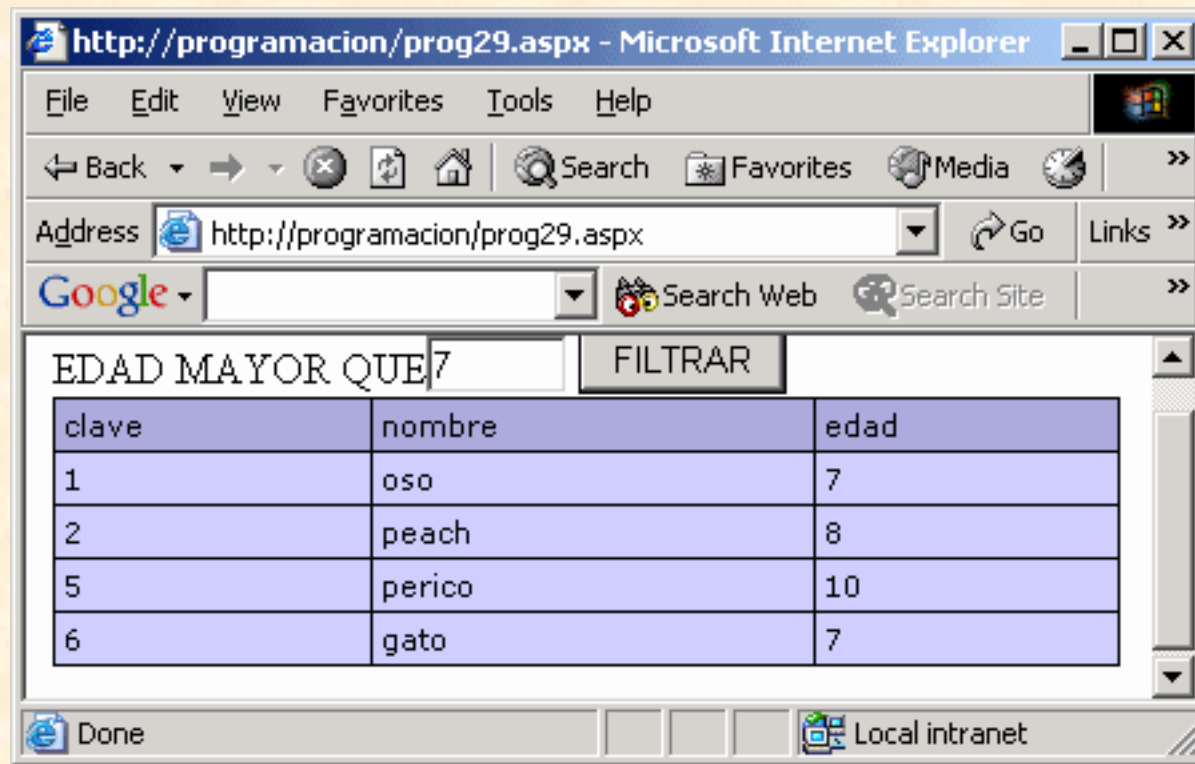
}

</script>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

Nota: siguen siendo combinaciones de los programas anteriores pero seria prudente mejor usar dos combobox uno para la variable, otro para el operador relacional y un text para el dato y mandar estos tres datos al prog29.aspx (se ocupan varios command.parameters()), pero eso queda de tarea.

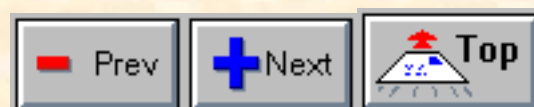
Corrida prog29.aspx



Problemas sugeridos:

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- preparar programas de filtrado para sus bases de datos, recordar que sus formas aspx deben construirlas con 2 combos y un text, suerte



UNIDAD 5: INTRODUCCION A LAS BASES DE DATOS

TEMA 11: OPERACIONES CON CAMPOS

Este es tambien un caso comun con elementos de una tabla, sin embargo es tambien facil de resolver.

Es necesario recordar primero algunas cosas elementales:

1.- Recordar que el numero de columna en una tabla empieza en 0, esto es que para realizar alguna operación por ejemplo la columna edad del ejemplo que estamos siguiendo, su numero de columna es la 2.

2.- La operación que se plantee se puede realizar con todos los renglones de la tabla o con un solo renglon de la tabla(del dataset), para procesar todos los renglones se usa un ciclo for, si solo se quiere procesar un solo renglon o una celda o columna nada mas, solo recordar GET-SET y solo usar un `tabla.tables.rows[r][c]` con los metodos strings apropiados.

3.- Para realizar aritmetica con toda una columna, solo usar el GET-SET de `tabla.tables.rows[ren]col` para leer(get) o cargar(set), en leer recordar que saldra una string y en cargar recordar que se tendra que cargar tambien una string, otra vez;

```
string alfa=tabla.Tables["Clientes"].Rows[4][5].ToString(); -->carga  
como string la variable alfa con el dato que se tiene en la sexta  
columna del quinto renglon de la tabla clientes.
```

```
tabla.Tables["alumnos"].Rows[2][3]="MAMA"; --> carga con la string  
MAMA la cuarta columna del tercer renglon de la tabla alumnos.
```

5.- En el ejemplo se realiza la operación con todos los renglones de la tabla **y no olvidar que se tiene que usar la instruccion sql Update**

para que la nueva información se actualize en disco, recordar que los cambios que se hacen a la tabla, es realmente al dataset, que a su vez es una tabla o base de datos en la memoria de la maquina del cliente o usuario y estos cambios hay que actualizarlos o pasarlos o UPDATE a la base de datos en disco.

El siguiente programa le aumenta 50 a todas las edades.

Prog30.aspx

```
<%@ PAGE LANGUAGE=C# %>

<%@ Import Namespace="System" %>

<%@ Import Namespace="System.Data" %>

<%@ Import Namespace="System.Data.OleDb" %>

<FORM RUNAT=SERVER>

<ASP:BUTTON ONCLICK=SUMAR TEXT=EDAD+50 RUNAT=SERVER /><BR>

<ASP:DATAGRID ID=TABLAGRID RUNAT=SERVER

Width=400

BackColor=#ccccff

BorderColor=black

ShowFooter=false

CellPadding=3

CellSpacing=0

Font-Name=Verdana

Font-Size=8pt
```

```
HeaderStyle-BackColor=#aaaadd
```

```
EnableViewState=false
```

```
/>
```

```
</FORM>
```

```
<script runat=server>
```

```
// creando la coneccion a la base de datos variable global
```

```
OleDbDataAdapter canal;
```

```
DataSet tabla;
```

```
OleDbConnection coneccion;
```

```
void Page_Load(object sender, EventArgs e)
```

```
{
```

```
coneccion =new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data  
Source=c:\\prografacil\\tusitio\\mibase.mdb");
```

```
canal=new OleDbDataAdapter("select * from mitabla", coneccion);
```

```
tabla= new DataSet();
```

```
canal.Fill(tabla, "mitabla");
```

```
TABLAGRID.DataSource=tabla;
```

```
TABLAGRID.DataMember="mitabla";
```

```
TABLAGRID.DataBind();
```

```
}
```

```
void SUMAR (Object sender, EventArgs e)
```

```
{
```

```
int temp,r;

//trabajando con el dataset

int cren=tabla.Tables["mitabla"].Rows.Count;

for (r=0; r<=cren-1; r++)

{

// cargando clave del renglon actual a actualizar

// recordar que se visitara cada renglon de la tabla en memoria

int nvaclave=Int32.Parse(tabla.Tables["mitabla"].Rows[r][0].ToString());

//sacando el valor de la columna EDAD y cargando en la variable temp
recordad GET-SET

temp= Int32.Parse(tabla.Tables["mitabla"].Rows[r][2].ToString() );

//sumandole 50 a todas las edades

temp=temp+50;

// cargando ahora la columna EDAD con el nuevo valor de temp (RECORDAR GET-SET)

tabla.Tables["mitabla"].Rows[r][2]=temp.ToString();

// actualizando tambien la base de datos

String q = "UPDATE mitabla SET edad = "+temp+" where clave= "+nvaclave;

OleDbCommand orden = new OleDbCommand(q, coneccion);

orden.Connection.Open();

orden.ExecuteNonQuery();

orden.Connection.Close();
```



```
// termina el for(renglon) que se usa para visitar todos los renglones de
la tabla

};

//refrescando el datagrid patra ver en browser los cambios en disco

TABLAGRID.DataSource=tabla;

TABLAGRID.DataMember="mitabla";

TABLAGRID.DataBind();

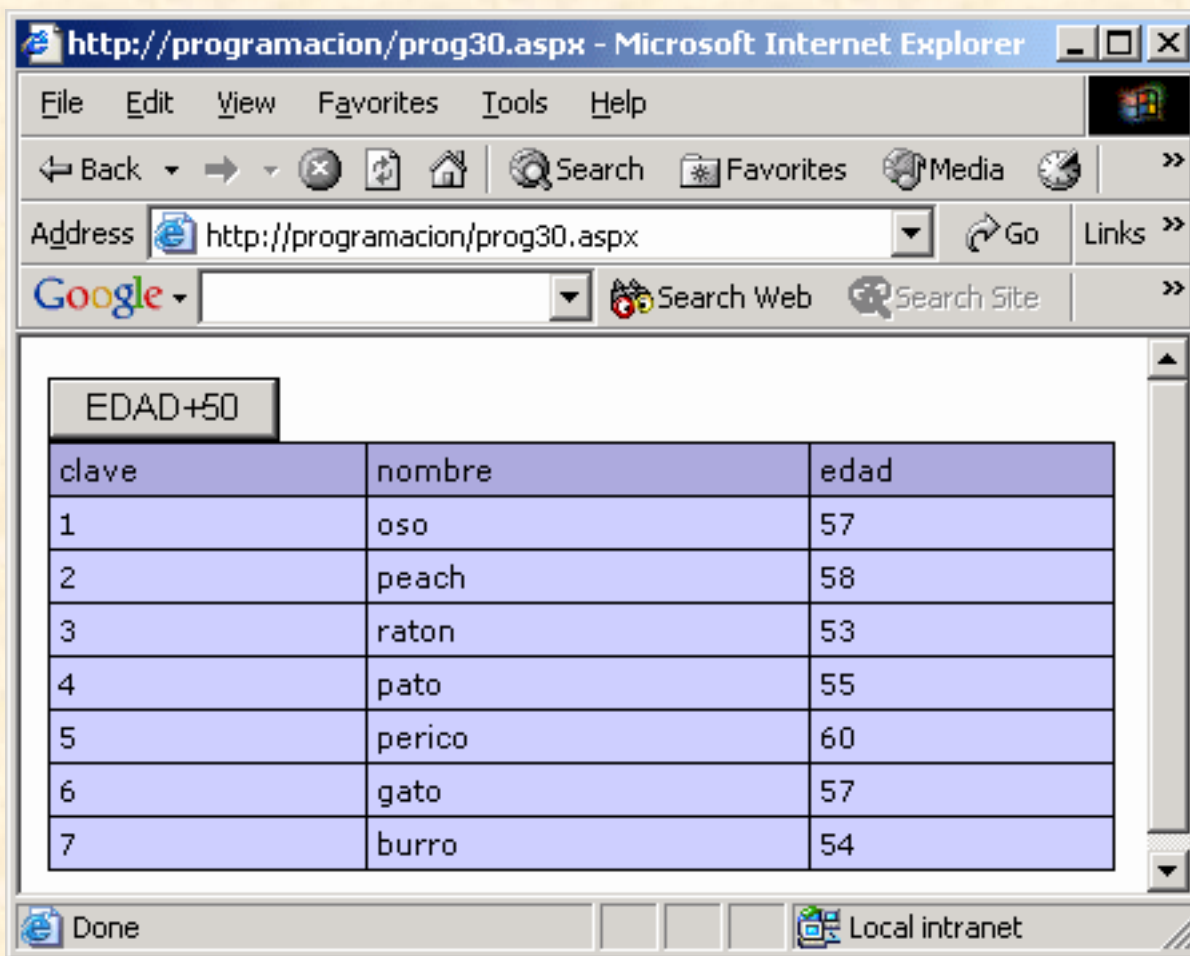
}

</script>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

nota: como se observa se puede construir directamente la string q, y no usar command.parameters(), si se esta muy seguro que los tipos de datos que se mandan a disco son los apropiados para access.

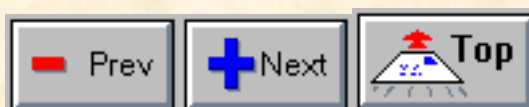
Corrida prog30.aspx



problema sugerido:

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- construir una tabla en access que traiga matricula, nombre, calif1, calif2, calif3 y promedio, cargar en access unos 5 renglones de alumnos, no cargar promedio, el promedio lo deberan calcular con un aspx.



UNIDAD 5: INTRODUCCION A LAS BASES DE DATOS

TEMA 12: BAJA O ELIMINACION

Eliminación es otro proceso simple y comun con las bases de datos el modelo con ADO.NET que estamos usando hace este tipo de operaciones muy faciles:

La instrucción sql a usar es: DELETE FROM TABLA WHERE CONDICION

Prog31.aspx

```
<%@ PAGE LANGUAGE=C# %>
```

```
<%@ Import Namespace="System" %>
```

```
<%@ Import Namespace="System.Data" %>
```

```
<%@ Import Namespace="System.Data.OleDb" %>
```

```
<FORM RUNAT=SERVER>
```

```
CLAVE A BORRAR<ASP:TEXTBOX ID=CLAVE SIZE=3 RUNAT=SERVER/>
```

```
<ASP:BUTTON ONCLICK=BORRAR TEXT=BORRAR RUNAT=SERVER /><BR>
```

```
<ASP:DATAGRID ID=TABLAGRID RUNAT=SERVER
```

```
Width=400
```

```
BackColor=#ccccff
```

```
BorderColor=black
```

```
ShowFooter=false
```

```
CellPadding=3
```

```
CellSpacing=0
```

```
Font-Name=Verdana
```

```
Font-Size=8pt
```

```
HeaderStyle-BackColor=#aaaadd
```

```
EnableViewState=false
```

```
/>
```

```
</FORM>
```

```
<script runat=server>
```

```
OleDbConnection coneccion;
```

```
DataSet tabla;
```

```
OleDbDataAdapter canal;
```

```
void Page_Load(object sender, EventArgs e)
```

```
{
```

```
coneccion=new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data  
Source=c:\\prografacil\\tusitio\\mibase.mdb");
```

```
canal=new OleDbDataAdapter("select * from mitabla", coneccion);
```

```
DataSet tabla= new DataSet();
```

```
canal.Fill(tabla, "mitabla");
```

```
TABLAGRID.DataSource=tabla;
```



```
TABLAGRID.DataMember="mitabla";

TABLAGRID.DataBind();

}

void BORRAR (Object sender, EventArgs e)
{

// instruccion sql DELETE FROM TABLA WHERE CLAVE=DATO

string q="delete from mitabla where clave=@CLAVE";

OleDbCommand orden= new OleDbCommand(q, coneccion);

orden.Parameters.Add(new OleDbParameter("@CLAVE", OleDbType.Integer));

orden.Parameters["@CLAVE"].Value = CLAVE.Text;

orden.Connection.Open();

orden.ExecuteNonQuery();

orden.Connection.Close();

// REFRESCANDO DATASET

canal=new OleDbDataAdapter("select * from mitabla", coneccion);

DataSet tabla= new DataSet();

canal.Fill(tabla, "mitabla");

TABLAGRID.DataSource=tabla.Tables["mitabla"].DefaultView;

TABLAGRID.DataBind();

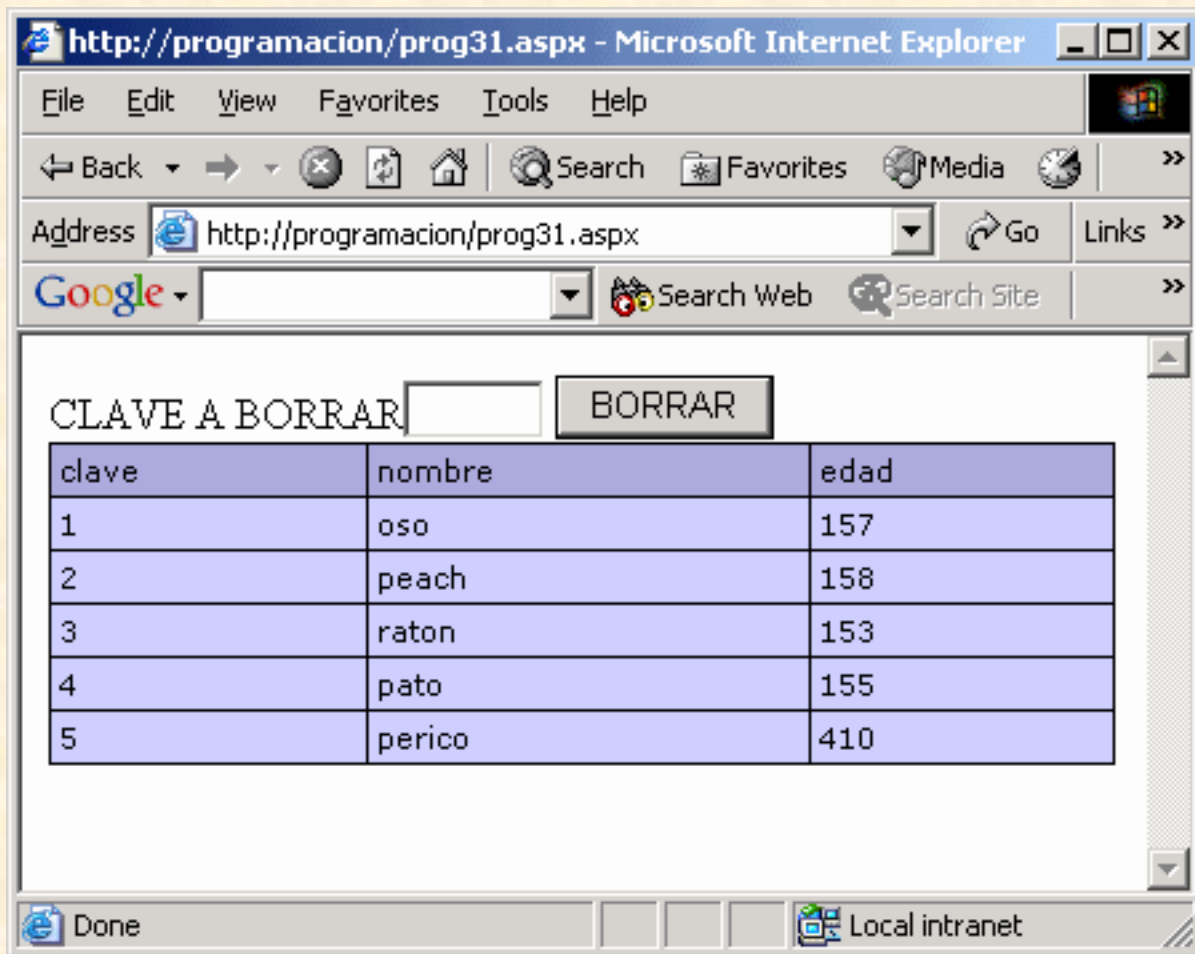
CLAVE.Text=" ";

}
```

</script>

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

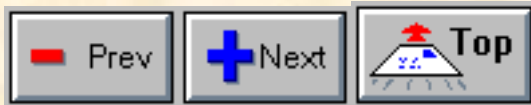
corrida prog31.aspx



problemas sugeridos:

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- construir este proceso para las tablas y bases de datos que tengan construidas.



UNIDAD 5: INTRODUCCION A LAS BASES DE DATOS

TEMA 13: EDICION DE REGISTROS

Editar registros significa cambiar el contenido de algunos de los campos o columnas por nueva información o para corregir algun error de captura original o para agregar alguna columna que no existia por modificación de la tabla o la base de datos.

En general se tiene otro problema de sql UPDATE, sin embargo ahora se aprovechan algunos elementos nuevos del objeto datagrid, como son la capacidad que tiene de crear columnas de edicion a los renglones que muestra el dataset, estas columnas de edición traen sus propios metodos, mismos que se pueden cargar con codigo para procesar.

Prog32.aspx

```
<%@ Import Namespace="System" %>
```

```
<%@ Import Namespace="System.Data" %>
```

```
<%@ Import Namespace="System.Data.OleDb" %>
```

```
<script language="C#" runat="server">
```

```
OleDbConnection coneccion =
```

```
new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\\progfamil\\tusitio\\mibase.mdb");
```

```
void Page_Load(object sender, EventArgs e)
```

```
{ // solo para cuando se carga por primera vez la forma
```

```
if (!IsPostBack)
```

```
DespTabla();

}

public void DespTabla()

{

OleDbDataAdapter canal=new OleDbDataAdapter("select * from mitabla", coneccion);

DataSet tabla= new DataSet();

canal.Fill(tabla, "mitabla");

TABLAGRID.DataSource=tabla;

TABLAGRID.DataMember="mitabla";

TABLAGRID.DataBind();

}

public void DataGrid_Edit(Object sender, DataGridCommandEventArgs e)

{

// cargando el renglon donde se pidio la edicion

// con la nueva columna y cajas de texto para las columans normales

TABLAGRID.EditItemIndex = (int) e.Item.ItemIndex;

DespTabla();

}

public void DataGrid_Cancel(Object sender, DataGridCommandEventArgs e)

{

// para regresar al estado original o normal

// solo poner edititemindex en -1 (ningun renglon)
```



```
TABLAGRID.EditItemIndex = -1;

DespTabla();

}

public void DataGrid_Update(Object sender, DataGridCommandEventArgs e)

{

String q = "UPDATE mitabla SET clave= @CLAVE, nombre= @NOMBRE, edad = @EDAD
where clave= @CLAVE";

OleDbCommand orden = new OleDbCommand(q, coneccion);

orden.Parameters.Add(new OleDbParameter("@CLAVE", OleDbType.Integer));

orden.Parameters.Add(new OleDbParameter("@NOMBRE", OleDbType.VarWChar, 20));

orden.Parameters.Add(new OleDbParameter("@EDAD", OleDbType.Integer));

//cargando textbox de clave con el valor de la clave

orden.Parameters["@CLAVE"].Value = TABLAGRID.DataKeys[(int)e.Item.
ItemIndex];

// creando y cargando los demas textboxes que le aparecieron al usuario

String[] nomcajas = {"@CLAVE", "@NOMBRE", "@EDAD"};

for (int i=2; i <= 3; i++)

{

String datocajas = ((TextBox)e.Item.Cells[i].Controls[0]).Text;

orden.Parameters[nomcajas[i-1]].Value = Server.HtmlEncode(datocajas);

}

orden.Connection.Open();

orden.ExecuteNonQuery();
```

```
// poniendo otra vez el datagrid en nada o estado original
```

```
TABLAGRID.EditItemIndex = -1;
```

```
orden.Connection.Close();
```

```
DespTabla();
```

```
}
```

```
</script>
```

```
<body style="font: 10pt verdana">
```

```
<form runat="server">
```

```
<h3><font face="Verdana">EDICION O ACTUALIZACION DE REGISTROS</font></h3>
```

```
<span id="Message" EnableViewState="false" style="font: arial 11pt;"  
runat="server"/><p>
```

```
<ASP:DataGrid id="TABLAGRID" runat="server"
```

```
Width="400"
```

```
BackColor="#ccccff"
```

```
BorderColor="black"
```

```
ShowFooter="false"
```

```
CellPadding=3
```

```
CellSpacing="0"
```

```
Font-Name="Verdana"
```

```
Font-Size="8pt"
```

```
HeaderStyle-BackColor="#aaaadd"
```

```
OnEditCommand="DataGrid_Edit"

OnCancelCommand="DataGrid_Cancel"

OnUpdateCommand="DataGrid_Update"

DataKeyField="clave"

>

<Columns>

<asp:EditCommandColumn EditText="Edit" CancelText="Cancel"
UpdateText="Update" ItemStyle-Wrap="false"/>

</Columns>

</ASP:DataGrid>

</form>

</body>
```

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

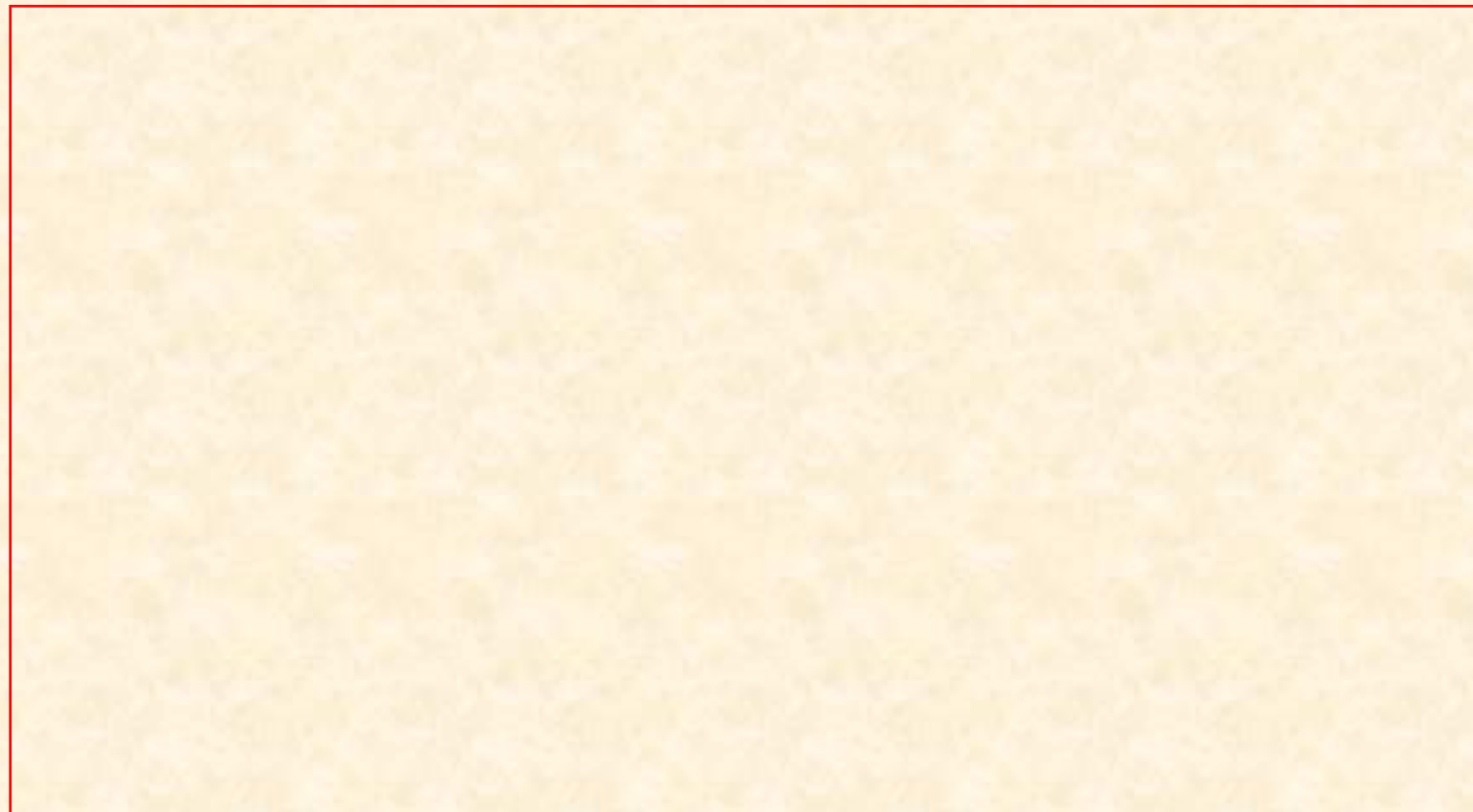
Para entender el codigo veamos la corrida completa:

Pantalla uno prog32.aspx



Observar que ahora el datagrid incluye una columna de edicion especial, revisar la parte del codigo de propiedades del datagrid en el programa y las nuevas propiedades que se le agregaron.

Pantalla dos prog32.aspx



Observar que la columna de edicion del renglon seleccionado (click en edit de cualquier renglon) ahora tiene dos opciones(update y cancel) y el renglon de edicion se convirtio en puros textbox (ya se modificarón algunos valores), update y cancel tienen su propio codigo en el programa, revisarlo y usando opcion update se tiene ahora;

Pantalla tres prog32.aspx

Un registro editado o modificado, analizar con cuidado el código del programa, que está documentado, suerte

NOTA: A TODOS NUESTROS AMABLES VISITANTES QUE ESTEN INTERESADOS EN PLANTAR Y EJECUTAR ESTOS PROGRAMAS AQUI EN WWW.PROGRAMACIONFACIL.COM POR FAVOR HACER CLICK EN ESTE [ANUNCIO IMPORTANTE](#) para poder abrirles una cuenta e instrucciones de uso apropiadas.

1.- construir aspx's de edición para sus tablas y bases de datos

