Herramientas de Programación 1

Tema Nº6:

Programa en C#

Indicador de logro Nº6:Aplica los elementos fundamentales de programación utilizando en el lenguaje C# para el desarrollo de aplicaciones

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº6:**

Aplicación de Visual Studio

Las variables en la computadora las podemos imaginar como pequeñas cajas que existen en la memoria de la computadora y su tamaño dependerá de la información que deben guardar. Esto se conoce como tipo de la variable. Para poder acceder a esa caja o variable le ponemos un nombre que sería la etiqueta con la que la identificamos.

Para hacer uso de una variable, lo primero que tenemos que hacer es declararla. La declaración de éstas es algo muy sencillo. Como primer paso tenemos que colocar el tipo y luego el nombre.

Las variables en C# deben nombrarse de acuerdo con unas recomendaciones sencillas:

* Los nombres de las variables deben empezar con letra.
* Es posible colocar números en los nombres de las variables, pero no empezar el nombre con ellos.
* Los nombres de las variables no pueden llevar signos a excepción del guion bajo \_.
* Las variables no pueden llevar tildes en sus nombres.

Cuando nombramos las variables, hay que tener en cuenta que C# es sensible a las mayúsculas y minúsculas, por lo que una variable llamada costo no es la misma que otra variable llamada COSTO u otra llamada Costo.

Es recomendable nombrar a las variables con nombres que hagan referencia a la información que guardarán. Si nos acostumbramos a hacer esto desde que empezamos a programar, evitaremos muchos problemas en el futuro y será mucho más sencillo corregir nuestros programas cuando tengan errores.

Numéricos

Veamos un ejemplo de cómo podemos declarar una variable que guarde valores numéricos enteros. El tipo que guarda estos valores se conoce como int.

**Ejemplos:**

int base\_imponible;

int impuesto;

Pero también es posible declarar las variables en una sola línea. Para esto simplemente separamos los nombres de las variables con comas. No hay que olvidar colocar el punto y coma al final de la sentencia.

**Ejemplos:**

int base\_imponible, impuesto;

C# nos provee de muchos tipos para poder usar con nuestras variables, que podemos conocer en la siguiente tabla:

|  |  |
| --- | --- |
| Propiedad | Descripción |
| int | Cuando queremos guardar valores numéricos enteros con signo, en el rango de -2,147,483,648 hasta 2,147,483,647. |
| uint | Para valores numéricos enteros positivos, su rango de valores es desde 0 hasta 4,294,967,295. |
| long | Guarda valores numéricos enteros realmente grandes con un rango desde -9,223,372,036,854,775,808 hasta 9,223,372,036,854,775,807. |
| uling | Guarda valores numéricos enteros positivos. Su rango de valores varía desde 0  hasta 18,446,744,073,709,551,615. |
| short | Guarda valores numéricos enteros, pero su rango es menor y varía desde -32,768 hasta 32,767. |
| ushort | Puede guardar valores numéricos enteros positivos con un rango desde 0 hasta 65,535. |
| float | Guarda valores numéricos con decimales (4 bytes). Para este tipo el rango es desde ±1.5 x 10-45 hasta ±3.4 x 10-38. |
| double | Guarda valores numéricos con decimales (8 bytes). Para este tipo el rango es desde ±5.0 x 10-324 hasta ±1.7 x 10-308. |
| decimal | Guarda valores numéricos con decimales (16 bytes). Para este tipo el rango es desde ±7.9 x 10-28 hasta ±7.9 x 10-28. |
| string | Este tipo nos permite guardar cadenas. |
| bool | Puede tomar dos valores: verdadero (true) o falso (false) |

Es importante conocer los tipos y la información que pueden guardar, ya que esto nos permitirá guardar la información necesaria y utilizar la menor cantidad de memoria. Podemos aprender los rangos de los tipos o imprimir una tabla y tenerla a mano. Con la práctica podremos utilizar los tipos sin tener que verificar los rangos constantemente.

Una vez que hemos declarado la variable, debemos inicializarla. Inicializar una variable es asignarle un valor por primera vez. Esto nos permite darle un valor inicial que puede ser útil en la ejecución de nuestro programa (no sólo podemos asignarle a una variable un valor fijo, sino también puede ser desde un texto ingresado por el usuario).

Para asignarle un valor a una variable, ya sea durante la inicialización o durante el transcurso del programa, usamos un operador de asignación, el signo igual.

Veamos a continuación un ejemplo de cómo asignar valores:

**Ejemplos:**

int base\_imponible = 100;

int impuesto = 18;

String

Este tipo nos permite guardar cadenas.

Veamos a continuación un ejemplo de cómo asignar valores:

string mensaje = "Bienvenido";

string cadena = txtCadena.Text;

Revisaremos las funciones para trabajar con cadenas de texto (string).

|  |  |
| --- | --- |
| Función | Descripción |
| Length | Permite obtener la longitud de la cadena. |
| SubString | Recupera una subcadena indicándole la ubicación inicial y la cadena de caracteres. |
| IndexOf | Devuelve el índice de un carácter específico. |
| Replace | Reemplaza un carácter o caracteres de una cadena por otros caracteres. |
| ToUpper | Convierte a mayúscula los caracteres de la cadena. |
| ToLower | Convierte a minúscula los caracteres de la cadena. |
| Trim | Quita los espacion en blanco al inicio y al final de la cadena de caracteres. |

Bool

Este tipo nos permite guardar cadenas.

En C# tenemos también un tipo de datos llamado "booleano", que puede tomar dos valores: verdadero (true) o falso (false):

**Ejemplos:**

bool encontrado, conDescuento;

encontrado = true;

conDescuento = false;

Así como existen operadores aritméticos también existen operadores lógicos, son los siguientes:

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Operación | Ejemplo | Resultado |
| & | Y lógico | test1 & test2 | Verdadero si ambos son ciertos |
| && | Y lógico | test1 && test2 | Igual que &, pero solo evalúa test2 si test1 es verdadero |
| | | O lógico | test1 | test2 | verdadero si alguno de los dos es verdadero |
| || | O lógico | test1 || test2 | Igual que |, pero solo evalúa test2 si test1 es falso |
| ! | Negación | ! test1 | Verdadero si test1 es falso, falso si test1 es verdadero |
| ^ | O exclusivo | test1 ^ test2 | Verdadero si alguno es cierto y el otro NO lo es |

Operadores Aritméticos

Para realizar cálculos aritméticos debemos seguir una precedencia entre los operadores, en el caso del ejemplo primero se ejecuta la multiplicación y luego la división. La siguiente tabla muestra la precedencia de los operadores en C#. Se encuentran listados de mayor a menor precedencia.

|  |  |
| --- | --- |
| Operador | Descripción |
| \* | Multiplicación |
| / | División |
| % | Módulo |
| + | Adición |
| - | Sustracción |

Para que podamos organizar una expresión hacemos uso de los paréntesis. Cada sección que tengamos en los paréntesis se evalúa como una expresión y el resultado se pasa al resto de la expresión. Por ejemplo, mostramos la secuencia de ejecución de una expresión más complicada:

* resultado = (3 + 7) \* (36 + 4 \*(2+5));
* resultado = 10 \* (36 + 4 \* 7);
* resultado = 10 \* (36 + 28);
* resultado = 10 \* 64;
* resultado = 640;

Operadores Lógicos

Así como existen operadores aritméticos también existen operadores lógicos, son los siguientes:

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Operación | Ejemplo | Resultado |
| & | Y lógico | test1 & test2 | Verdadero si ambos son ciertos |
| && | Y lógico | test1 && test2 | Igual que &, pero solo evalúa test2 si test1 es verdadero |
| | | O lógico | test1 | test2 | verdadero si alguno de los dos es verdadero |
| || | O lógico | test1 || test2 | Igual que |, pero solo evalúa test2 si test1 es falso |
| ! | Negación | ! test1 | Verdadero si test1 es falso, falso si test1 es verdadero |
| ^ | O exclusivo | test1 ^ test2 | Verdadero si alguno es cierto y el otro NO lo es |

Operadores de Comparación

Los operadores de comparación son:

|  |  |  |
| --- | --- | --- |
| Operador | Descripción | Ejemplo |
| == | Igual | if (precio == 0) |
| != | Distinto | if (precio != 0) |
| > | Mayor que | if (edad >= 17) |
| >= | Mayor o igual que | if (edad >= 18) |
| < | Menor que | if (edad < 18) |
| <= | Menor o igual que | if (edad <= 17) |

ARREGLOS

Los arreglos son grupos de variables y estas variables serán referenciadas por el mismo nombre. Para poder acceder a una variable del arreglo usaremos un número de índice, ya que todas las variables adentro de un arreglo serán de un mismo tipo. Un punto muy importante que no debemos olvidar cuando trabajemos con los arreglos es que éstos están basados en índice cero, esto quiere decir que el primer elemento del arreglo se encuentra en la posición 0, no en la posición 1 como podríamos pensar. No olvidar este punto nos evitará muchos problemas de lógica en el futuro.

En un arreglo es conveniente colocar la cantidad correcta de elementos, ya que una vez creado, no puede crecer a menos que utilicemos una función especial. Si hacemos lo último en forma constante, esto indica que nuestro programa tiene problemas de diseño. En el próximo capítulo veremos estructuras que permiten variar la cantidad de elementos a guardar.

Declaración de los arreglos de una dimensión.

**Ejemplos:**

int[] calificaciones = new int[4];

En C# los arreglos son objetos, y deberemos usar new al declararlos. El arreglo, en el ejemplo, es de tipo entero, y usaremos [ ] para indicar su declaración. Luego debemos colocar el nombre con el que lo identificaremos. En nuestro ejemplo se llama calificaciones.

Del lado derecho de la sentencia tenemos la instanciación del arreglo. Indicaremos entre [ ] la cantidad de elementos que deseamos tener. La cantidad puede ser colocada de forma explícita, tal como está en el ejemplo, o por medio del contenido de una variable. Podemos ejemplificar esto de la siguiente forma:

**Ejemplos:**

int n = 4;

int[] calificaciones = new int[n];

La cantidad de datos debe ser un valor válido. No podemos colocar números negativos, ni un tamaño de cero ya que no tendría sentido. Si la cantidad de elementos a crear se pasará por medio de una variable, ésta debe de ser de tipo entera. No tener en cuenta esto traerá problemas al compilar la aplicación.

En algunas ocasiones podemos conocer los valores que colocaremos adentro del arreglo, por lo que podemos declararlo y asignarle sus valores en la misma sentencia.

Esto lo hacemos indicando primero el tipo y los [ ] seguidos del nombre del arreglo y en el lado derecho de la sentencia colocamos entre { } los elementos que se le desean asignar al arreglo. Estos elementos deberán estar separados por comas. Veámoslo ejemplificado de manera más clara:

**Ejemplos:**

int[] calificaciones = {15, 10, 16, 18};

Asignación y uso de valores.

Para poder asignarle un valor a alguno de los elementos del arreglo necesitamos hacer uso del índice del elemento que queremos utilizar, y como dijimos antes, no debemos olvidar que el primer elemento se encuentra en la posición 0. Supongamos que queremos asignarle la calificación 20 al primer alumno y 12 al tercer alumno

**Ejemplos:**

int[] calificaciones = {15, 10, 16, 18};

int n = 2;

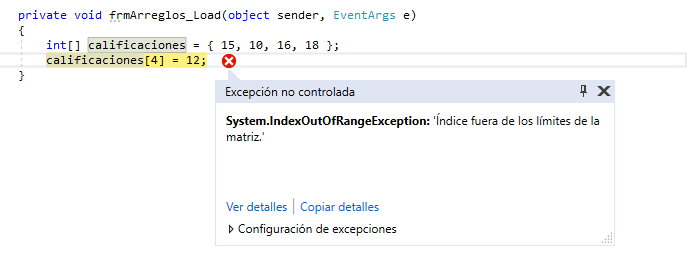
calificaciones[0] = 20;

calificaciones[n] = 12;

//el arreglo quedaría con los siguientes valores {20, 10, 12, 18};

Si ingresamos un índice que supere la cantidad de elementos en el arreglo hara que se genere una excepción.

**Ejemplos:**



**Ejemplos:**

**RECURSOS**

1. Hardware

* Una computadora con 4 GB e RAM y procesador de 2.0 GHz.

1. Software

* Visual Studio Professional 2015 / 2017

**PROCEDIMIENTO**

**Laboratorio 6:**

6.1 Estructura básica de un programa Windows Forms.

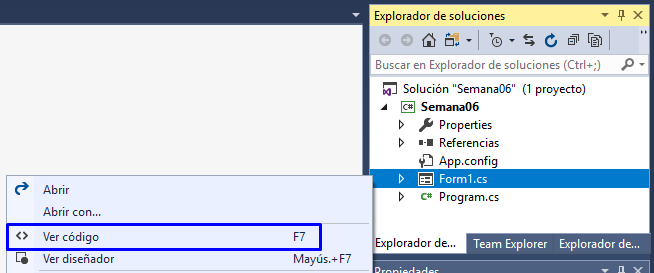
**Objetivos**

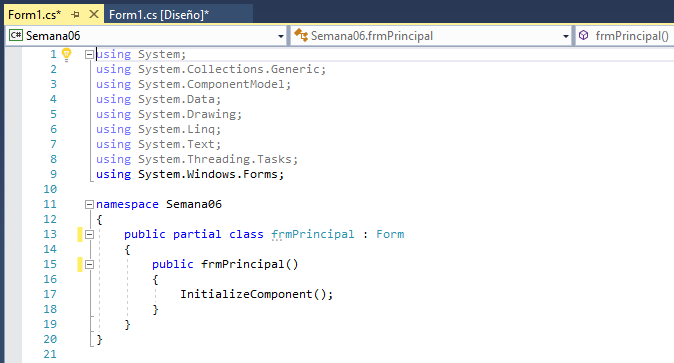
* Entender cuál es la estructura de un proyecto tipo Windows Forms desarrollado en C#..

**Introducción**

Durante esta actividad, vamos a revisar la estructura de un proyecto/solución Windows Forms en Visual Studio.

Vamos a explicar la estructura básica de un programa con formularios Windows. Para ello vamos a crear el proyecto Semana06, al formulario inicial le ponemos el nombre frmPrincipal, revisamos el código inicial del formulario.





Las primeras líneas son las sentencias using, estas hacen referencia a varias librerías (componentes de software) que facilitan el desarrollo de nuestras aplicaciones. Estas se van activando y agregando al proyecto a medida que vamos utilizando alguna de sus funcionalidades. En posteriores semanas veremos cómo se van activando las que se muestran sombreadas y también podremos agregar otras que aún no están incluidas en el proyecto.

Un namespace (espacio de nombres) identifica generalmente a un proyecto, se utiliza para agrupar clases, las clases dentro de un namespace pueden ser referenciadas directamente dentro del proyecto.

Las clases en C# se declaran mediante la palabra clave [class](https://docs.microsoft.com/es-es/dotnet/csharp/language-reference/keywords/class) seguida por un identificador único.

La palabra clave class va precedida del nivel de acceso. Como en este caso se usa [public](https://docs.microsoft.com/es-es/dotnet/csharp/language-reference/keywords/public), cualquier usuario puede crear instancias de esta clase. El nombre de la clase sigue a la palabra clave class. El nombre de la clase debe ser un [nombre de identificador](https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/inside-a-program/identifier-names) de C# válido. Los identificadores deben comenzar con una letra, o \_.

El resto de la definición es el cuerpo de la clase, donde se definen los datos y el comportamiento. Los campos, las propiedades, los métodos y los eventos de una clase se denominan de manera colectiva miembros de clase.

En el caso del formulario, además debemos explicar lo siguiente:

* El identificador partial indica que una clase puede ser declara en distintos lugares (archivos).
* La sentencia: Form indica que nuestro formulario frmPrincipal hereda de la clase padre Form de C#.

6.2 Tipo de datos int, double, string y bool.

Objetivos

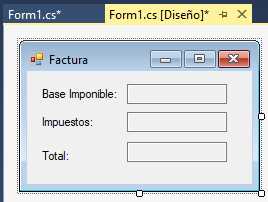
* Entender cómo aplicar correctamente los tipos distintos de datos a las variables de nuestras aplicaciones.

Introducción

Durante esta actividad, vamos a desarrollar aplicaciones que hacen uso de los tipos de datos: int, double, string y bool.

Paso 1. Tipo de dato int.

1. En el proyecto Semana06 diseñamos nuestro primer formulario de acuerdo a:



|  |  |  |
| --- | --- | --- |
| Control | Propiedad | Valor |
| Frm1 | (Name) | frmPrincipal |
|  | Text | Factura |
| TextBox1 | (Name) | txtBaseImponible |
|  | ReadOnly | True |
| TextBox2 | (Name) | txtImpuestos |
|  | ReadOnly | True |
| TextBox3 | (Name) | txtTotal |
|  | ReadOnly | True |

1. Agregamos código al evento Load.

private void frmPrincipal\_Load(object sender, EventArgs e)

{

int base\_imponible = 100;

int impuesto = 18;

int total = 0;

total = base\_imponible + impuesto;

txtBaseImponible.Text = base\_imponible.ToString();

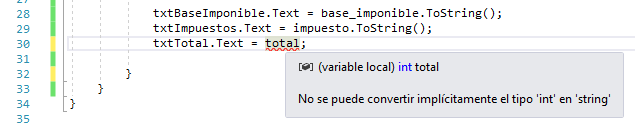
txtImpuestos.Text = impuesto.ToString();

txtTotal.Text = total.ToString();

}

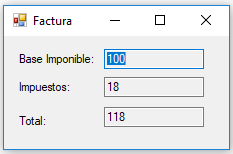
Para el cálculo del Total estamos haciendo uso del operador aritmético suma (+).

Debemos resaltar que en una asignación tanto el miembro de la izquierad debe ser del mismo tipo que el de a derecha, caso contrario el compilador nos mostrara mensajes de error como el siguiente:



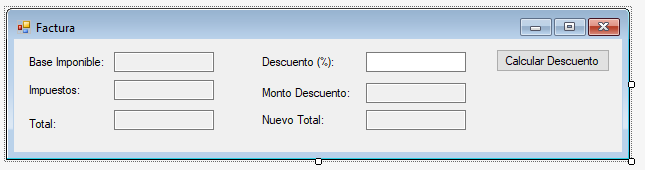
En este ejemplo, la variable total es de tipo int y la propiedad Text del control TextBox es de tipo string. Debemos convertir el valor entero a string para poder asignarlo correctamente, esta conversión lo hace la función ToString().

1. Ejecutamos la aplicación.



Paso 2. Tipo de dato double.

1. Para el siguiente ejemplo vamos agregar nuevos controles al formulario.



|  |  |  |
| --- | --- | --- |
| Control | Propiedad | Valor |
| Button1 | (Name) | btnCalcularDescuento |
|  | Text | Calcular Descuento |
| TextBox4 | (Name) | txtPorcentajeDescuento |
| TextBox5 | (Name) | txtMontoDescuento |
|  | ReadOnly | True |
| TextBox6 | (Name) | txtNuevoTotal |
|  | ReadOnly | True |

1. Agregamos código al evento Click del botón.

private void btnCalcularDescuento\_Click(object sender, EventArgs e)

{

double total = Convert.ToDouble(txtTotal.Text);

double porcentajeDescuento = Convert.ToDouble(txtPorcentajeDescuento.Text);

double montoDescuento = total \* porcentajeDescuento / 100;

double nuevoTotal = total - montoDescuento;

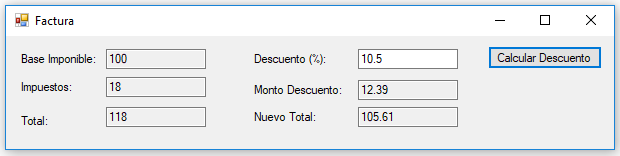
txtMontoDescuento.Text = montoDescuento.ToString();

txtNuevoTotal.Text = nuevoTotal.ToString();

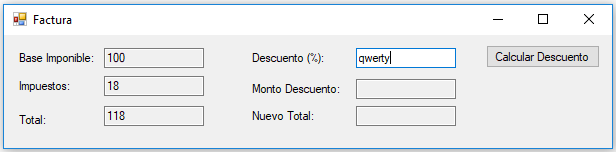
}

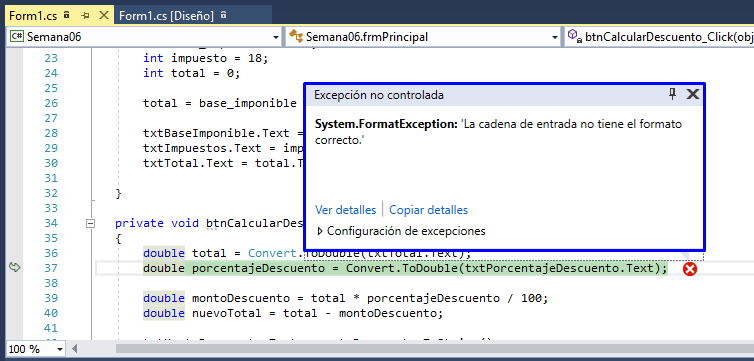
Como vamos a trabajar con números decimales declaramos varias variables del tipo double, lo primero que hacemos es convertir el porcentaje ingresado por el usuario al tipo de dato double ya que lo que nos devuelve una caja de texto es un dato de tipo string. Para la conversión entre tipos de datos se utiliza generalmente la clase Convert.

1. Ejecutamos la aplicación.



Debemos resaltar que no estamos haciendo validación del dato de entrada, podría ser que el usuario ingrese un dato no numérico y pueda hacer que el programa muetre un error. Por ejemplo si ingresamos caracteres no numéricos Visual Studio respondería con una Excepcion no controlada.



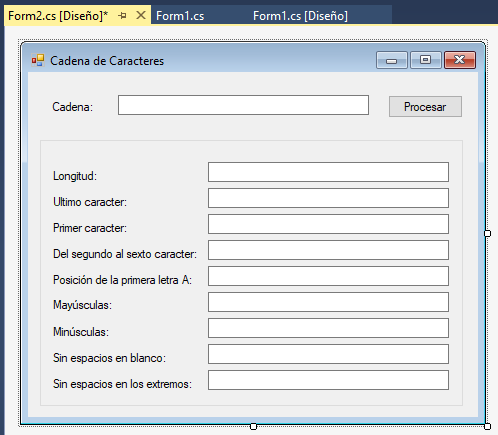


Esto responde a que la cadena ingresada no es un número y por lo tanto no se pudo hacer correctamente la conversión.

El manejo de excepciones se trata en la clase de la última semana.

Paso 3. Tipo de dato string.

1. Creamos el siguiente formulario.



1. Agregamos el siguiente código al evento Click del botón btnProcesar

private void btnProcesar\_Click(object sender, EventArgs e)

{

string cadena = txtCadena.Text;

//Calculamos la longitud de la cadena

textBox2.Text = Convert.ToString(cadena.Length);

//Mostramos el primer caracter

textBox3.Text = cadena.Substring(0, 1);

//Mostramos el último caracter

textBox4.Text = cadena.Substring(cadena.Length - 1, 1);

//Mostramos el del segundo al sexto caracter

textBox5.Text = cadena.Substring(2, 4);

//Buscamos la posición de la primera letra "A"

textBox6.Text = Convert.ToString(cadena.IndexOf("A", 0));

//Convertimos las letras a mayúsculas

textBox7.Text = cadena.ToUpper();

//Convertimos las letras a minúsuculas

textBox8.Text = cadena.ToLower();

//Eliminar todos los caracteres en blanco

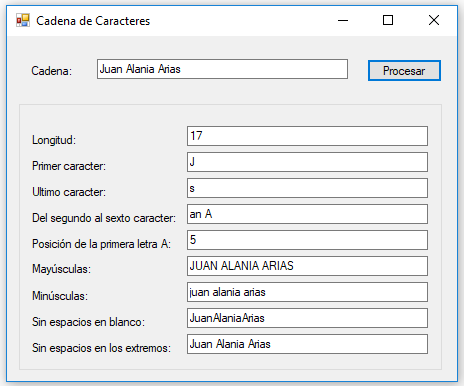
textBox9.Text = cadena.Replace(" ", "");

//Eliminar los caracteres en blanco de los extremos

textBox10.Text = cadena.Trim();

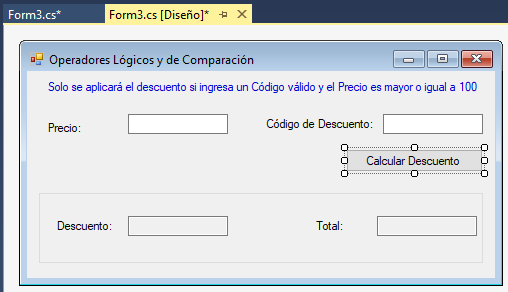
}

1. Ejecutamos la aplicación.



Paso 4. Tipo de dato bool.

1. Para el siguiente ejemplo creamos un nuevo formulario con el siguiente diseño:



|  |  |  |
| --- | --- | --- |
| Control | Propiedad | Valor |
| Form3 | (Name) | frmOperadores |
|  | Text | Operadores Lógicos y de Comparación |
| TextBox1 | (Name) | txtPrecio |
| TextBox2 | (Name) | txtCodigoDescuento |
| TextBox3 | (Name) | txtDescuento |
|  | ReadOnly | True |
| TextBox4 | (Name) | txtTotal |
|  | ReadOnly | True |
| Button1 | (Name) | btnCalcularDescuento |
|  | Text | Calcular Descuento |

1. Agregamos código el evento Click del botón de acuerdo a:

private void btnCalcularDescuento\_Click(object sender, EventArgs e)

{

double precio = Convert.ToDouble(txtPrecio.Text);

string codigoDescuento = txtCodigoDescuento.Text.Trim();

double porcentajeDescuento = 0;

double montoDescuento = 0;

double total = precio;

bool tieneDescuento = false;

if(codigoDescuento == "101" || codigoDescuento == "102" || codigoDescuento == "103")

{

porcentajeDescuento = 10;

}

if (codigoDescuento == "201" || codigoDescuento == "202" || codigoDescuento == "203")

{

porcentajeDescuento = 20;

}

if (porcentajeDescuento > 0)

{

tieneDescuento = true;

}

if(tieneDescuento && precio >= 100)

{

montoDescuento = precio \* porcentajeDescuento / 100;

total = precio - montoDescuento;

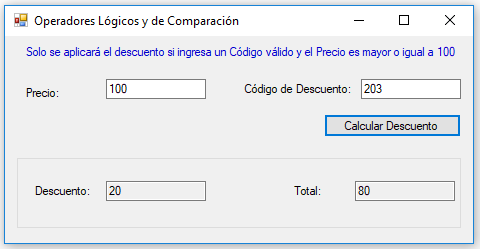
}

txtDescuento.Text = montoDescuento.ToString();

txtTotal.Text = total.ToString();

}

1. Ejecutamos la aplicación y verificamos que solo se aplica el descuento si el usuario ingresa un Código de Descuento válido y el Precio es mayor o igual a 100.



**6.3 Arreglos**.

**Objetivos**

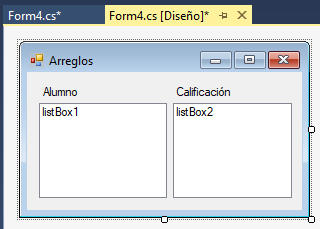
* Entender cómo aplicar correctamente los arreglos en nuestras aplicaciones.

**Introducción**

Durante esta actividad, vamos a desarrollar aplicaciones que hacen uso de arreglos con distintos tipos de datos.

Paso 1. Arreglos.

1. Para el siguiente ejemplo vamos agregar un nuevo formulario con el siguiente diseño:



|  |  |  |
| --- | --- | --- |
| Control | Propiedad | Valor |
| Form4 | (Name) | frmArreglos |
|  | Text | Arreglos |
| ListBox1 | (Name) | lsbAlumnos |
| ListBox2 | (Name) | lsbCalificaciones |

1. Agregamos código al evento Load del formulario de acuerdo a:

private void frmArreglos\_Load(object sender, EventArgs e)

{

int[] calificaciones = { 15, 10, 16, 18 };

string[] alumnos = new string[4];

alumnos[0] = "Pedro Mora";

alumnos[1] = "Adriana Cuestas";

alumnos[2] = "Julio Lizana";

alumnos[3] = "Isabel Mandujano";

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

{

lsbAlumnos.Items.Add(alumnos[i]);

}

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

{

lsbCalificaciones.Items.Add(calificaciones[i].ToString());

}

/\*

//los dos for anteriores se pueden reducir en uno solo

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

{

lsbAlumnos.Items.Add(alumnos[i]);

lsbCalificaciones.Items.Add(calificaciones[i].ToString());

}

\*/

}

1. El procesamiento de la información de un arreglo o cualquier colección se trata con estructuras iterativas, en el ejemplo utilizamor la estructura for, todas las instrucciones que están dentro de la estructura se van ejecutar una cierta cantidad de veces, esta cantidad esta definida dentro de los parámetros de la estructura:

* int i = 0

Se declara una variable que nos ayudara a la contabilización de las veces que se va a ejecutar las instrucciones de la estructura, esta variable empieza en cero (0)

* i < 4

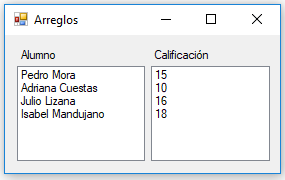
Las instrucciones se van a ejecutar mientras esta expresión sea verdadera, una vez que sea falsa saldrá del for y ejecutara las instrucciones siguientes.

* i++

Indica el incremento de la variable i cada vez que se ejeucta la estrutura iterativa, en este caso se incrementa en 1

En la semana siguiente vamos a explicar a detalle estas estructuras y otras mas.

1. Ejecutamos la aplicación.



**Actividad:**

Ingresa a la plataforma virtual. Revisar y analizar los siguientes enlaces, luego realiza la actividad propuesta:

* **Sintaxis básica en C#.**
* <https://www.youtube.com/watch?v=SRT6uq5UTbc&list=PLU8oAlHdN5BmpIQGDSHo5e1r4ZYWQ8m4B&index=3>
* <https://www.youtube.com/watch?v=18-j9OOHVQc&list=PLU8oAlHdN5BmpIQGDSHo5e1r4ZYWQ8m4B&index=4>
* <https://www.youtube.com/watch?v=-RwdksWQpnQ&list=PLU8oAlHdN5BmpIQGDSHo5e1r4ZYWQ8m4B&index=5>
* <https://www.youtube.com/watch?v=CyG6oaNq8HE&list=PLU8oAlHdN5BmpIQGDSHo5e1r4ZYWQ8m4B&index=6>
* <https://www.youtube.com/watch?v=ZheVmw7CCnw&list=PLU8oAlHdN5BmpIQGDSHo5e1r4ZYWQ8m4B&index=7>
* **Arreglos.**
* <https://www.youtube.com/watch?v=I1ADyLMTAwo&list=PLU8oAlHdN5BmpIQGDSHo5e1r4ZYWQ8m4B&index=36>
* https://www.youtube.com/watch?v=G3o96kwDbA4&list=PLU8oAlHdN5BmpIQGDSHo5e1r4ZYWQ8m4B&index=37

1. En un nuevo formulario, crea un arreglo o unidimensional donde le indiques el tamaño por teclado y crear rellenar el arreglo con los múltiplos de un numero pedido por teclado también. Mostrar los elementos del arreglo en un ListBox. Por ejemplo:

