Tema: FUNDAMENTOS DE PROGRAMACIÓN EN LENGUAJE C#

Objetivo:

- ✓ Conocer y manejar correctamente los tipos de datos y las diferentes estructuras de programación que hacen parte del lenguaje de Programación “C#” en un proyecto tipo “Windows Form”.

INTRODUCCIÓN

Una aplicación de consola es aquella que se ejecuta en una ventana que solo permite ingresar línea de comandos, es decir, no tiene elementos gráficos tales como botones o cajas de texto con que el usuario interactúe. Lo más común dentro del desarrollo bajo la plataforma .Net es la creación de aplicaciones tipo Windows o aplicaciones Web, sin embargo la mejor forma de sentar unas bases firmes acerca de la programación en este lenguaje es comenzar construyendo aplicaciones sencillas de tipo consola.

I. TIPOS DE DATOS, DEFINICIÓN DE VARIABLES Y CONSTANTES EN C#

Recibir datos de entrada, realizar un proceso u obtener salidas en un programa escrito en lenguaje C# requiere necesariamente del uso de variables, contantes y operadores.

Las variables son datos cuyo valor puede cambiar en cualquier momento durante la ejecución de un programa; en términos computacionales una variable es un espacio de memoria que permite almacenar temporalmente un dato. En el lenguaje C# para poder usar una variable se requiere definirla por medio de un nombre y un tipo de dato de la siguiente forma.

tipodedato NombreDeVariable ;

El tipo de dato especifica la naturaleza del valor que se va a almacenar, por ejemplo si es un numero entero, decimal, pequeño, grande o si es una cadena de caracteres (Texto). Los Tipos de datos comúnmente utilizados en leguaje c# son:

Tipo De Dato	Clase .NET	Descripción	Tamaño (En Bytes)	Intervalo o Rango De Valores
byte	Byte	Entero sin signo	8	0 a 255
sbyte	SByte	Entero con signo	8	-128 a 127
int	Int32	Entero con signo	32	-2.147.483.648 a 2.147.483.647
uint	UInt32	Entero sin signo	32	0 a 4294967295
short	Int16	Entero con signo	16	-32.768 a 32.767
ushort	UInt16	Entero sin signo	16	0 a 65535

long	Int64	Entero con signo	64	-922337203685477508 a 922337203685477507
ulong	UInt64	Entero sin signo	64	0 a 18446744073709551615
float	Single	Tipo de punto flotante de precisión simple	32	-3,402823e38 a 3,402823e38
double	Double	Tipo de punto flotante de precisión doble	64	-1,79769313486232e308 a 1,79769313486232e308
char	Char	Un carácter	16	Símbolos o caracteres Unicode utilizados en el texto
bool	Boolean	Tipo Boolean lógico	8	True o false
string	String	Una secuencia de caracteres	En función de los caracteres	Secuencia de Símbolos o caracteres Unicode utilizados en el texto
decimal	Decimal	Tipo preciso fraccionario o integral, puede representar números decimales con 29 dígitos significativos	128	±1.0 × 10 e−28 a ±7.9 × 10 e28

Las constantes son valores que se conocen y no cambian durante la ejecución de un programa. Las constantes se declaran de forma similar a una variable utilizando como la palabra clave "const" antes del tipo de dato. Las constantes se deben inicializar en el momento que se declaran. Por ejemplo:

```
const int meses = 12;
const int dias = 365;
const double pi = 3,1416;
```

II. OPERADORES EN LENGUAJE C#

1. Operadores Aritméticos:

Operador	Significado
(+)	Suma
(-)	Resta
(*)	Multiplicación
(/)	División
(%)	Modulo

2. Operadores De Asignación

Operador	Significado
(=)	Asignación simple
(+=)	Asignación suma
(-=)	Asignación Resta
(*=)	Asignación Multiplicación
(/=)	Asignación División

3. Operadores Relacionales o De Comparación

Operador	Significado
(==)	Igual que
(!=)	Diferente De
(>)	Mayor que
(>=)	Mayor o Igual que
(<)	Menor que
(<=)	Menor o Igual que
(++)	Incremento
(--)	Decremento

4. Operadores Lógicos

Operador	significado
(&&)	AND , Operador "Y" Lógico
(||)	OR , Operador "O" Lógico
(!)	Not , Operador "Negación"

5. Operador De Concatenación: (+)

Es la operación por la cual dos caracteres se unen para formar una cadena de caracteres (o *string*). También se puede concatenar dos cadenas de caracteres o un carácter con una cadena para formar una cadena de mayor tamaño. Algunos ejemplos serían:

- ✓ 'a' *concatenado 'b'* → *"ab"*
- ✓ "ABCD" *concatenado 'b'* → *"ABCDb"*
- ✓ 'a ' *concatenado "XYZ"* → *"a XYZ"*
- ✓ "ABCD " *concatenado "XYZ"* → *"ABCD XYZ"*

Para realizar una concatenación en Lenguaje C# se usan valores o variable tipo char o string y se unen con el símbolo (+)

Ejemplos:

- ✓ "El valor de a es" + a. ToString ();
- ✓ "Su edad es" + edad. ToString ();

6. Operadores de Conversión:

a) Para convertir el tipo de dato en otro, su precede el tipo que queremos cambiar con el tipo al que queremos convertir, entre paréntesis, de esta forma: variable1 = (tipo) variable2;

Ejemplos:
✓ int a = (int) b;
✓ d = (double) x;

b) En el lenguaje C# los tipos de datos numéricos cuentan con una función que permite convertir cadenas de texto a un tipo numérico especifico. Esta función es muy útil para capturar los datos de entrada numéricos, ya que estos en la mayoría de los casos ingresan al programa como cadenas de texto (string) y para poder hacer operaciones matemáticas con estos es necesaria su conversión. Esta función es conocida como "Parse" y aplica para todos los tipos de datos numéricos, algunos ejemplo de su uso:

Ejemplo1:
```
int n1 = int.Parse(textBox1.Text);
```

Ejemplo2:
```
double nota = double.Parse(textBox2.Text);
```

c) El Lenguaje C# también cuenta con un conjunto de funciones que realizan conversiones entre los diferentes tipos de datos. Estas funciones se encuentran en la "Librería" Convert de C# y se utiliza como se ve en el siguiente ejemplo:

```
double dNumber = 23.15;
int    iNumber = Convert.ToInt32(dNumber);
```

7. Operador Condicional:

Operador Condicional: Es el único operador de C# que tiene tres operandos. Su sintaxis es esta:

<condición> ? <expresión1> : <expresión2>;

Quiere decir que si la condición es true, se evalúa expresión1, y si es falsa, se evalúa expresión2. No se debe confundir el operador condicional con un "if". Este operador devuelve un valor, mientras que el if es una instrucción usada para la toma de decisiones

Ejemplo:

```
b = (a>0)? a : 0;      // a y b de tipos enteros
```

En este ejemplo, si el valor de la variable a es superior a 0 se asignará a b el valor de a, mientras que en caso contrario el valor que se le asignará será 0. Los caracteres "// " sirven para realizar comentarios (Texto que no se ejecuta) a las líneas de código en C#.

III. ENTORNO GRAFICO "WINDOWS FORMS" CON LENGUAJE C#

La plataforma .Net provee para todos los lenguaje que soporta (Visual Basic, C#, C++,etc.) un entorno gráfico basado en formularios (Ventanas) y controles. Cada formulario se comporta como una clase que tiene atributos y métodos; cada control colocado sobre un formulario se comporta como un atributo del formulario; tanto controles como formularios poseen sus *propiedades y eventos* que se emplean en la programación de la interfaz gráfica de usuario (I.G.U). Cada uno de los controles posee un evento principal por el cual se activa el control.

Windows Forms es una tecnología que se utiliza en C# para crear aplicaciones basadas en Windows que se ejecutan en Framework de .NET, es especialmente adecuado para escenarios de desarrollo rápido de aplicaciones donde la prioridad principal no es una interfaz gráfica de usuario compleja. El Diseñador de Windows Forms se utiliza para crear la interfaz de usuario, y permite obtener acceso a otras características de diseño y ejecución, tales como controles, Barras de herramientas y otros elementos de interfaz de usuario que pueden tener el aspecto y el comportamiento de Microsoft Windows.

IV. PROYECTOS TIPO "WINDOWS FORMS" CON LENGUAJE C#

Para el desarrollo de proyectos tipo Windows Forms se debe tener en cuenta tres momentos importantes denominados: tiempo de diseño, tiempo de Compilación y tiempo de ejecución.

- ✓ *Tiempo de Diseño:* es el intervalo de tiempo donde se realiza el diseño previo de las Interfaces gráficas y se escriben o digitan las líneas de código en el lenguaje de programación.

- ✓ *Tiempo De Compilación* (compile-time): es el intervalo de tiempo en el que se transforma o convierte el código escrito en un lenguaje de programación a una forma de código ejecutable por una máquina. El compilador también realiza un chequeo de sintaxis (Depuración), que incluye entre otros un chequeo de tipos y de reglas propias del lenguaje.

- ✓ *Tiempo de Ejecución* (runtime): es el intervalo de tiempo en el que un programa de computadora se ejecuta en un sistema operativo. Este tiempo se inicia con la puesta en memoria (RAM) del código ejecutable del programa, por lo que el sistema operativo comienza a ejecutar sus instrucciones.

Los tres pasos básicos para crear interfaces de usuario en un proyecto de tipo *Windows Form* son:

1. Agregar los controles a la superficie de diseño.
2. Establecer las propiedades iniciales de los controles.
3. Escribir las tareas o funcionalidades para programar los eventos requeridos.

Paso 1: Agregar Controles

Se utiliza el mouse para arrastrar controles, que son los componentes con representación visual, como botones y cuadros de texto, hasta una superficie de diseño. A medida que se trabaja visualmente, el Diseñador de Windows Forms traduce las acciones en código fuente de C# y las escribe en un archivo de proyecto llamado nombre.designer.cs donde nombre es el nombre asignado al formulario. Cuando se ejecuta la aplicación, ese código fuente (Windows Forms) ajustará la posición y el tamaño de los elementos de la interfaz de usuario de modo que aparezcan tal como en la superficie de diseño.

Paso 2: Establecer propiedades

Después de agregar un control a la superficie de diseño, se puede utilizar la "ventana de Propiedades" para establecer valores personalizados sus propiedades, como son el color de fondo y el texto predeterminado. En el Diseñador de Windows Forms, los valores que especifique en la ventana Propiedades son los valores iniciales que se asignarán a la propiedad cuando se cree el control en tiempo de ejecución. En muchos casos, se puede tener acceso a estos valores o modificarlos mediante programación en tiempo de ejecución; para ello, basta con obtener o establecer por medio de líneas de código la propiedad control. La ventana Propiedades resulta útil en tiempo de diseño porque permite examinar todas las propiedades, eventos y métodos que admite un control.

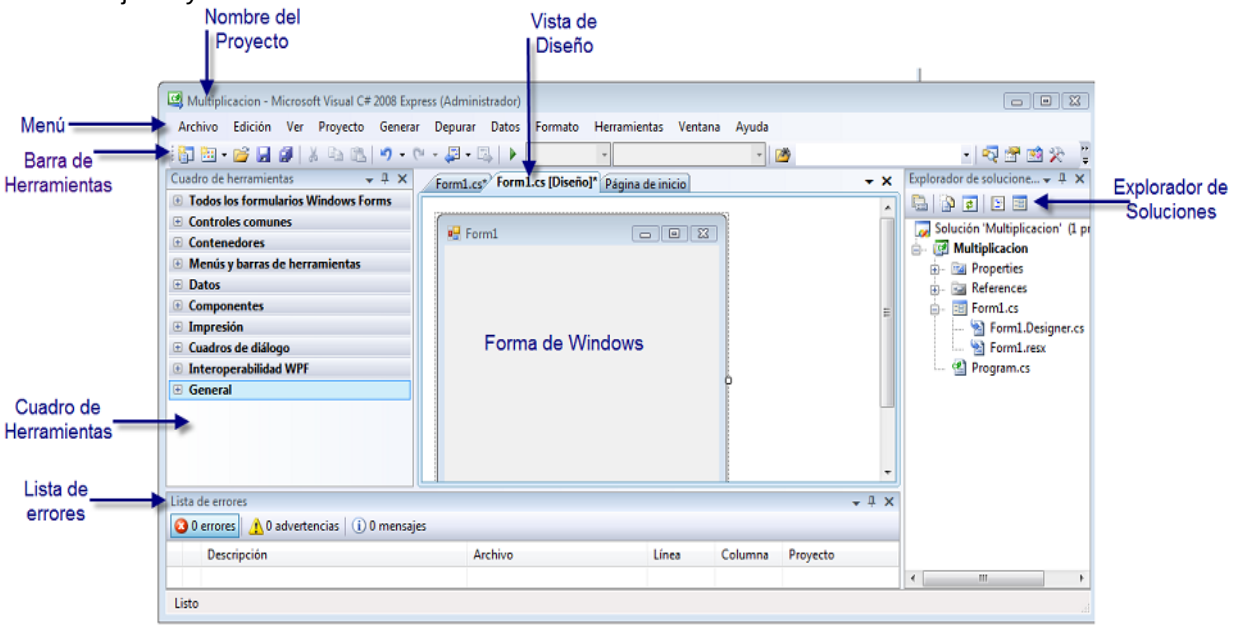
Paso 3: Programar eventos

Los programas con interfaces gráficas de usuario son controlados por eventos. Estos programas esperan a que un usuario haga algo, como escribir texto en un cuadro de texto, hacer clic en un botón o cambiar la selección de un cuadro de lista. Cuando esto sucede, el control (Botón, cuadro de texto, etc), envía un evento a la aplicación y se tiene la opción de controlar un evento escribiendo un método especial en la aplicación al que se llamará cuando se reciba el evento. Podemos utilizar la ventana Propiedades para especificar qué eventos desea programar en el código. Si seleccionamos un control en el diseñador y hacemos clic con el icono con forma de "Rayo" en la ventana Propiedades para ver los eventos del control. Al agregar un evento desde la ventana Propiedades, el diseñador escribe automáticamente el cuerpo del método vacío. Debemos escribir el código de manera que el método haga algo útil. La mayoría de los controles tienen muchos eventos, normalmente al dar doble click sobre el control en tiempo de diseño aparecerá directamente en la sección de código donde se ha de realizar la programación del evento.

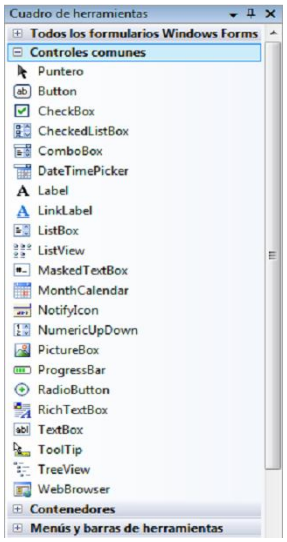
V. PASOS PARA CREAR UN PROYECTO TIPO WINDOWS FORMS

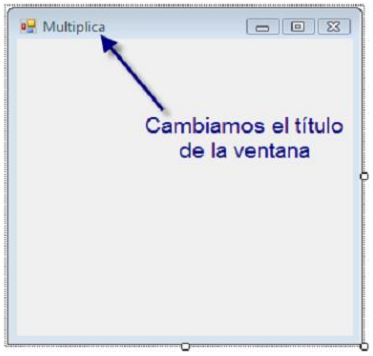
1. **Crear Proyecto:** Por medio del menú "Archivo –Nuevo proyecto– Visual C# - Windows– Aplicación Windows Forms" podrá acceder a un proyecto tipo Windows totalmente nuevo, al cual es recomendable colocarle un nombre para identificarlo fácilmente más adelante.

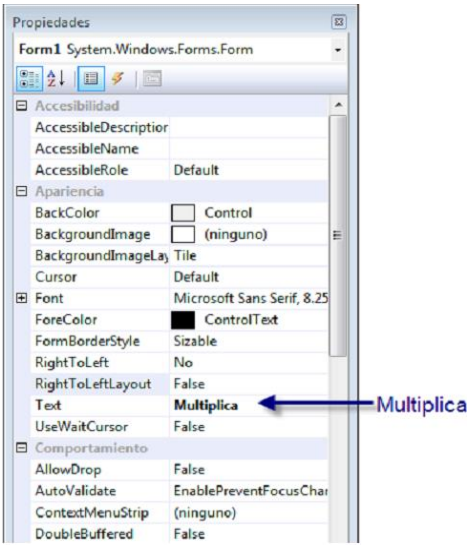
2. **Ventana De Trabajo:** Una vez creado el proyecto, aparece una ventana correspondiente al área de trabajo tal y como se muestra a continuación:



3. Cuadro de Herramientas y Ventana de Propiedades







EJERCICIO N°1.

1. Cree un proyecto nuevo en Visual Studio .Net; como lenguaje seleccione C# y como plantilla seleccione "*Aplicación de Windows Forms"*..

2. Desde el cuadro de herramientas, seleccione "*Controles Comunes" y* Sobre el formulario coloque los siguientes controles:

- ✓ Una Etiqueta (Label)
- ✓ Un cuadro de texto (Textbox)
- ✓ Un botón (Button)
- ✓ Un cuadro de Chequeo (CheckBox)
- ✓ Un cuadro desplegable (Combobox)
- ✓ Un botón de selección (Radiobutton)
- ✓ Una lista de selección (Listbox).
- ✓ Un Cuadro de Imagen(PictureBox)
- ✓ Un Cuadro de Fecha(DateTimePicker)
- ✓ Un Visualizador WEB(WebBrowser)

3. Manipule las propiedades más comunes de cada control, como por ejemplo (Name, Size, Height, Width, Backcolor, Font, Text, Location). *Observe y analice los resultados.*

4. Haga doble clic sobre cada control para acceder al editor de código. Observe que esta acción lo sitúa dentro del evento principal de cada control.

5. Cree un listado que describa a cada uno de los controles (¿Para qué sirve?, ¿Cómo se usa?), apunte cual es el evento principal de cada control y mencione bajo que circunstancia este se desencadena o ejecuta.

6. Agregue las principales propiedades de cada control junto a su significado y/o breve descripción

EJERCICIO N°2

1. Cree un nuevo proyecto en .Net seleccione el lenguaje C# y en plantillas seleccione "*Aplicación de Windows Forms"*

2. Sobre el formulario de trabajo coloque:
   - ✓ Una etiqueta (Label), en la propiedad Text escriba "Números";
   - ✓ Coloque dos cuadros de texto (TextBox);
   - ✓ Coloque dos botónes (Button) y en la propiedad Text escriba mostrar;
   - ✓ Finalmente coloque 4 etiquetas (Label) vacías.

3. Haga doble clic sobre el botón para acceder al editor de código y escriba el siguiente código:

Boton 1

```
int n1, n2;
int a;
float b;
double c;
decimal d;
n1 = int.Parse(textBox1.Text);
n2 = int.Parse(textBox2.Text);
a = n1 / n2;
label2.Text =a.ToString();
b = n1 / n2;
label3.Text = b.ToString();
c = n1 / n2;
label4.Text = c.ToString();
d = n1 / n2;
label5.Text = d.ToString();
```

Boton 2

```
int a, n1, n2;
float b, n3,n4;
double c,n5,n6;
decimal d,n7,n8;
n1 = int.Parse(textBox1.Text);
n2 = int.Parse(textBox2.Text);
n3 = float.Parse(textBox1.Text);
n4 = float.Parse(textBox2.Text);
n5 = double.Parse(textBox1.Text);
n6 = double.Parse(textBox2.Text);
n7 = decimal.Parse(textBox1.Text);
n8 = decimal.Parse(textBox2.Text);
a = n1 / n2;
label2.Text =a.ToString();
b = n3 / n4;
label3.Text = b.ToString();
c = n5 / n6;
label4.Text = c.ToString();
d = n7 / n8;
label5.Text = d.ToString();
```

4. Ingrese valores numéricos en ambos textbox, por ejemplo 10 y 9 , 10 y 7 , 10 y 3 …etc, haga Click en cada botón. *Observe, Analice los resultados obtenidos y escriba sus propias conclusiones.*

VI. ESTRUCTURAS CONDICIONALES – WINDOWS FORM - C#

3.1 Condicional Simple (If): Cuando se cumple una determinada condición, se ejecuta una o un conjunto de instrucciones específicas. Su estructura es:

```
if (condición Lógica)
{
    Instrucción 1;
    Instrucción 2;
    …
    Instrucción n;

}
```

EJERCICIO N° 3.1:

1. Cree un proyecto nuevo en C#, coloque una etiqueta y escriba Digite el numero; coloque una caja de texto vacía; coloque un botón y escriba calcular.

2. Haga doble clic sobre el botón para acceder al editor código y escriba el siguiente código:

```
int n, r;
n = int.Parse(textBox1.Text);
r = n % 2;
if (r != 0)
{
    MessageBox.Show("El numero no es par");
}
```

3.2 Condicionales Compuestos o Anidados (if – else)- (If – else if – else):
Se utilizan para evaluar diferentes condiciones lógicas en una misma estructura de toma de decisiones. Su estructura es:

If – else	If – else if - else
if(condición) **{** **Instrucción1;** **Instrucción2;** **…** **Instrucción n;** **}** **else** **{** **Instrucción1;** **Instrucción2;** **…** **Instrucción n;** **}**	**if (condición 1)** **{** **Instrucción1;** **Instrucción2;** **…** **Instrucción n;** **}** **else if (condición 2)** **{** **Instrucción1;** **Instrucción2;** **…** **Instrucción n;** **}** **else** **{** **Instrucción1;** **Instrucción2;** **…** **Instrucción n;** **}**

EJERCICIO N° 3.2:

1. Utilice el proyecto anterior para el ejercicio.
2. Modifique el código del botón y escriba el siguiente código:

```
int n, r;
n = int.Parse(textBox1.Text);
r = n % 2;
```

```
if (r != 0)
{
    MessageBox.Show("El numero no es par");
}
else
{
    MessageBox.Show("el numero es par");
}
```

3.3 Selección de Casos “switch”:

Se utiliza cuando se requiere en misma estructura evaluar varios valores que puede tomar una variable. Su estructura general es:

```
switch (variable)
    {
        case valor_1:
          {
                Instrucción1;
                Instrucción2;
                …
                Instrucción n;
                break;

          }
        case valor_2:
          {
                Instrucción1;
                Instrucción2;
                …
                Instrucción n;
                break;
          }

        case valor_n:
          {
                Instrucción1;
                Instrucción2;
                …
                Instrucción n;
                break;
          }

          default:
            {
                Instrucción1;
                Instrucción2;
                …
                Instrucción n;
                break;
          }
    }
```

EJERCICIO N° 3.3.

1. Cree un proyecto nuevo en C#, coloque una etiqueta (label) y un cuadro desplegable (combobox).

2. Haga doble clic sobre el formulario para acceder al editor de código al evento Load del formulario (Form1_Load…) y escriba el siguiente código:

```
comboBox1.Items.Add("Seleccione");
comboBox1.Items.Add("Azul");
comboBox1.Items.Add("Amarillo");
comboBox1.Items.Add("Verde");
comboBox1.Items.Add("Rojo");
```

4. haga doble clic sobre el "combobox" para acceder al editor de código al evento "`comboBox1_SelectedIndexChanged`" y escriba el siguiente código:

```
switch (comboBox1.Text)
{
    case "Verde":
      {
        label1.BackColor = System.Drawing.Color.Green;
        break;
      }
    case "Azul":
      {
        label1.BackColor = System.Drawing.Color.Blue;
        break;
      }
    case "Amarillo":
      {
        label1.BackColor = System.Drawing.Color.Yellow;
        break;
      }
    case "Rojo":
      {
        label1.BackColor = System.Drawing.Color.Red;
        break;
      }
    default:
      {
         MessageBox.Show("Por favor, Seleccione un Color");
         break;
      }
  }
```

Nota: si la variable a evaluar es de valor numérico los valores en el case no se colocan entre comillas.

VII. ESTRUCTURAS REPETITIVAS – WINDOWS FORMS - C#

Se usan cuando se requiere que un procedimiento se realice un número determinado de veces y se termine cuando se cumpla una condición específica.

4.1 La instrucción for: Su estructura general es:

```
for (variable = valorinicial; condición_de_fin; incremento)
 {
            Instrucción1;
            Instrucción2;
            …
 }
```

4.2 La instrucción while: Su estructura general es:

```
while (condición)
 {
            Instrucción1;
            Instrucción2;
            Incremento;
 }
```

4.3 La instrucción do – while: su estructura general es:

```
do
 {
   Instrucción1;
   Instrucción2;
  Incremento;

 } while (condición);
```

EJERCICIO N° 4:

1. Cree un proyecto nuevo en C#. Sobre el formulario coloque una etiqueta que diga “Número a calcular su factorial”; una caja de texto vacía; tres botones que digan respectivamente: for, while, do_while; y finalmente una etiqueta (label) vacía.

2. Haga doble clic sobre el botón que dice for y escriba el siguiente código:

```
int n, f=1;
n = int.Parse(textBox1.Text);
for(int i=1;i<=n;i++)
{
    f*=i;
}
label1.Text = f.ToString();
```

3. Haga doble clic sobre el botón que dice while y escriba el siguiente código:

```
int n, f=1, i=1;
n = int.Parse(textBox1.Text);

while (i <= n)
```

```
{
    f *= i;
    i++;
}
label1.Text = f.ToString();
```

4. Haga doble clic sobre el botón que dice do_while y escriba el siguiente código:

```
int n, f=1, i=1;
n = int.Parse(textBox1.Text);
do
{
    f *= i;
    i++;
}while (i <= n);
label1.Text = f.ToString();
```

TALLER:

1. Hacer el listado de los principales controles de Windows Form y realice una descripción del uso del control indicando a su vez el evento principal mostrando cuándo o como se desencadena **(Referirse al Ejercicio N°1 de esta guía)**

2. Realice una aplicación tipo Windows en lenguaje C# en la cual satisfaga los siguientes requerimientos:

    1. Permita ingresar un numero por medio de un cuadro de texto (TextBox)

    2. Presente un "ComboBox" con las siguientes opciones:

        - ✓ Factorial
        - ✓ Potencia
        - ✓ Seno
        - ✓ Coseno
        - ✓ Tangente
        - ✓ Raíz Cuadrada
        - ✓ Serie de Fibonacci.

    3. Cuando el usuario seleccione una de las opciones y presione click sobre algún botón, se debe mostrar en alguna etiqueta (Label) el resultado correcto de acuerdo a la opción seleccionada.

Notas Importantes:

- ✓ *Investigue sobre la librería Math y utilicela para las operaciones Seno, Coseno, Tangente y Raiz Cuadrada.*
- ✓ *Los cálculos de las operaciones Seno, Coseno y Tangente deben expresarse en radianes, es decir, que por ejemplo Seno (30) = 0,5*
- ✓ *Para el caso de la potencia, debe aparecer un textbox adicional para el exponente y este se debe ocultar para las demás operaciones. Esta Operación debe realizarse por medio de un ciclo repetitivo, es decir, no se permite el uso de la librería Math para esta operación.*