

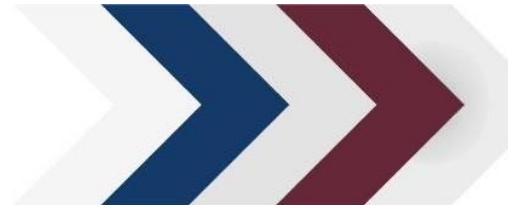
PROGRAMACIÓN I

TP N° 1 LABORATORIOS

UNIDAD 1 INTRODUCCIÓN A LA PROGRAMACIÓN VISUAL

Autor de contenidos:
Nicolás Battaglia

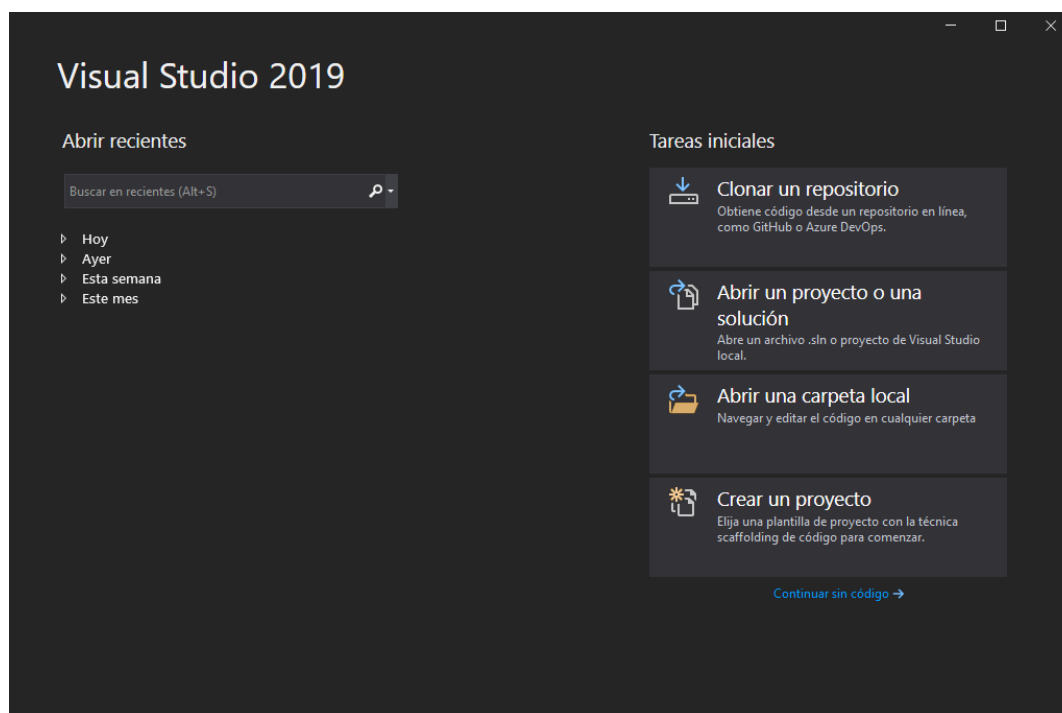
Corregido y ordenado: Pedro López



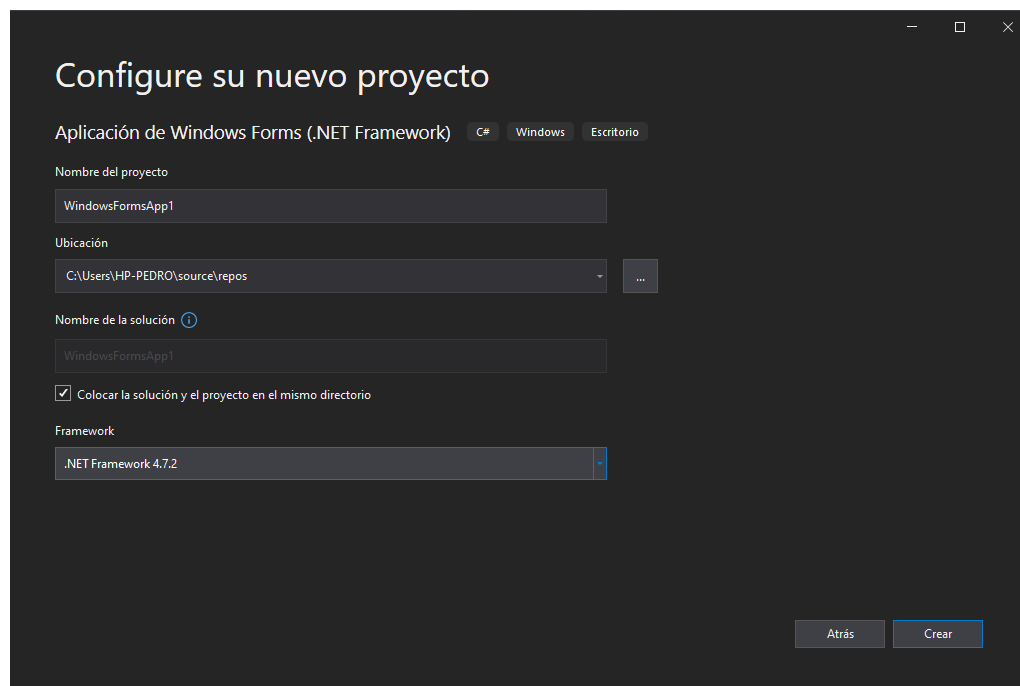
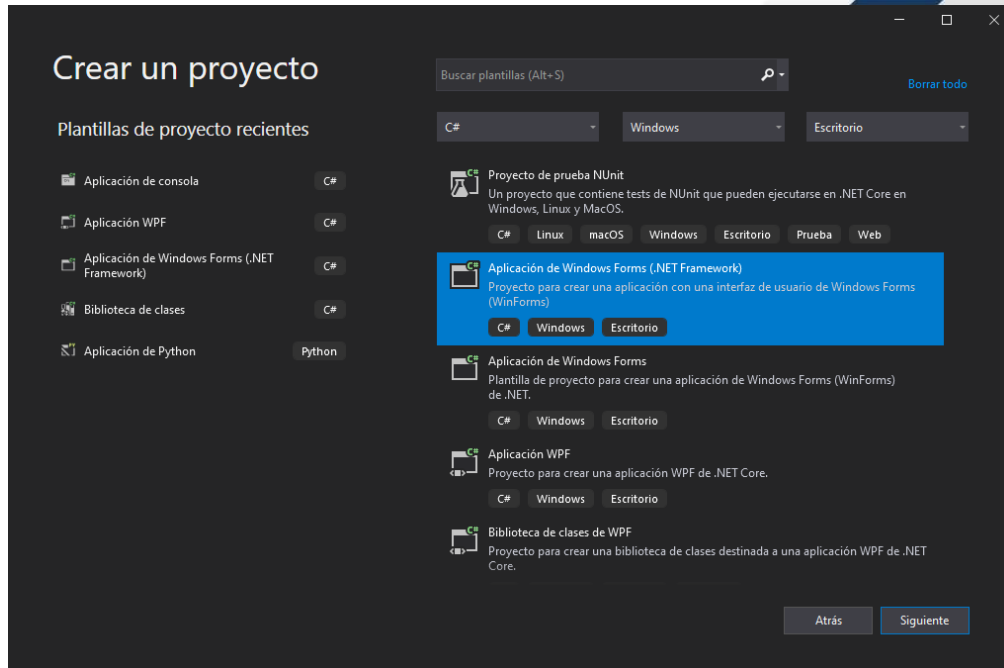
PRACTICA CON VISUAL STUDIO PROGRAMACION WINFORM

PARTE A

Ingresar a Visual Studio 2019



Crear un proyecto



Agregar → nombre del proyecto

Seleccionar Ubicación → seleccione una carpeta para guardar sus soluciones.

Verificar → que este tildada la casilla “Colocar la solución y le proyecto en el mismo directorio”

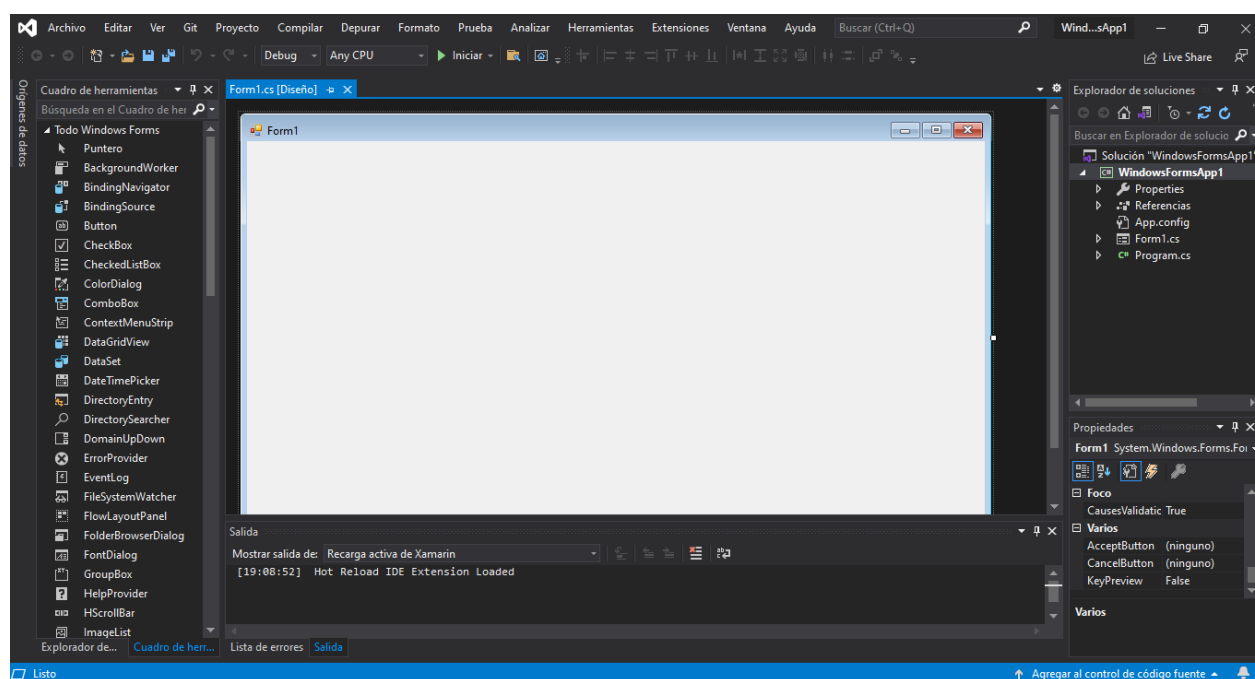
Seleccionar Crear.



Al dar aceptar entraremos en el IDE de .Net, donde debemos tener a la vista o en solapasa cada lado de la pantalla lo siguiente:

- Solapa de Cuadro de herramientas.
- Solapa de Explorador de soluciones.
- Solapa de Propiedades.

De no estar las agregaremos desde el menú Ver.



Tendremos también a la vista el primer **Form1** donde trabajar.

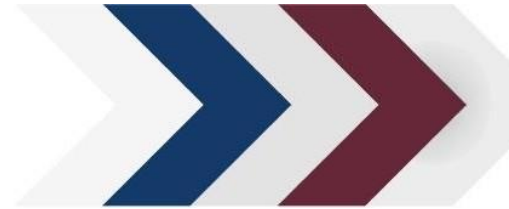
Del cuadro de herramientas y seleccionaremos los siguientes objetos que colocaremos dentro de este formulario:

3 TextBox.

3 Label.

5 Button.

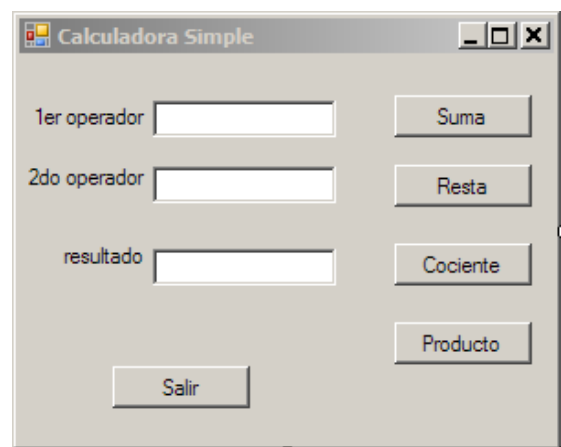
Para esto podemos seleccionarlo y arrastrarlo al Form1 o hacer doble click sobre



cada uno automáticamente se agregarán en el Form1.

Realizaremos el diseño según la pantalla siguiente y cambiaremos algunas propiedades de dichos controles. **Las propiedades a modificar serán:**

Controles	Text	Nombre
Form1	Calculadora simple	frmCalculadora
Label1	1er operador	lblOp1
Label2	2do operador	lblOp2
Label3	Resultado	lblRta
Textbox1		txtOp1
Textbox2		txtOp2
Textbox3		txtRta
Button1	Suma	btnSuma
Button2	Resta	btnResta
Button3	Producto	btnProd
Button4	Cociente	btnCoc
Button5	Salir	btnSalir



Podremos modificar algunas otras propiedades como: **BackColor**, **size** (siempre y cuando locked sea false), **autosize**, etc.

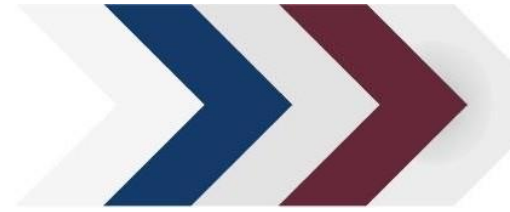
Nota: Jueguen e investiguen, de paso les va a ir sirviendo para el TP N° 2

Para igualar los tamaños de los controles los seleccionaremos, iremos a **formato**, **igualartamaños** y los dejara del tamaño del primero seleccionado (fijense que los puntos alrededor de este objeto están blancos mientras en los otros objetos seleccionados están negros); allí mismo podemos **alinear**, **dar espacios**, **centrar**, etcétera dichos controles.

Muy bien, pasemos al código, en cada botón escribiremos

En el evento Click del botón Resta escribimos

```
txtRta.Text = Convert.ToString(Convert.ToInt32(txtOp1.Text) - Convert.ToInt32(txtOp2.Text));
```



En el evento Click del botón Producto escribimos

```
txtRta.Text = (Int32.Parse(txtOp1.Text) * Int32.Parse(txtOp2.Text)).ToString();
```

En el evento Click del botón Cociente escribimos

```
txtRta.Text = (Int32.Parse(txtOp1.Text) / Int32.Parse(txtOp2.Text)).ToString();
```

En el evento Click del botón Suma escribimos

```
txtRta.Text = (Int32.Parse(txtOp1.Text) + Int32.Parse(txtOp2.Text)).ToString();
```

Es importante recordar que hay distintas formas de conversión, implícitas y explícitas. Dado que los tipos utilizados en el formulario no son de la misma familia como entero y doble, es necesario utilizar una conversión explícita.

Para ello recurrimos a Convert.XX o bien escribimos el tipo al que queremos convertir acompañado de “.Parse”.

En el evento Click del botón Cociente escribimos lo siguiente:

```
int num = Int32.Parse(txtOp1.Text);
int den = Int32.Parse(txtOp2.Text);
if (den != 0)
{
    txtRta.Text = (Int32.Parse(txtOp1.Text) / Int32.Parse(txtOp2.Text)).ToString();
}
```

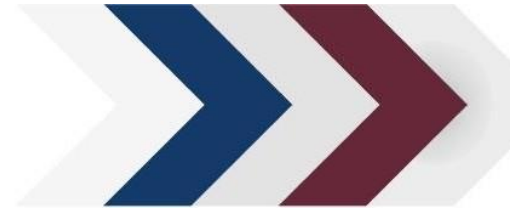
En el evento Click del botón Limpiar escribimos:

```
txtOp1.Text = "";
txtOp2.Text = "";
txtRta.Text = "";
this.txtOp1.Focus();
```

En el evento Click del botón Salir escribimos

```
Close();
```





A continuación: veremos algunos temas sueltos que en general a esta altura de la cursada los alumnos comienzan a preguntar.

Tengan en cuenta que no podemos ver todo de una vez y que cada uno de ustedes tiene sus propias inquietudes

Como hacer

Para modificar el tamaño de algunos controles

Por defecto algunos controles solo tienen activados algunos de sus manejadores, mientras que los otros están atenuados.

Esto se debe a la propiedad autosize que esta por defecto en True.

Para modificar el tamaño se debe modificar la propiedad size, width y height.

Para modificar el texto dentro de un label

Se modifica la propiedad text

¿Qué es un evento y cómo funciona o se puede modificar?

La mayoría de los controles responden ante distintos eventos, cada uno de ellos tiene un evento predeterminado, que es el que suele ocurrir más frecuentemente, el programador puede modificar esto al programar.

¿Qué son las soluciones, proyectos, su diferencia?

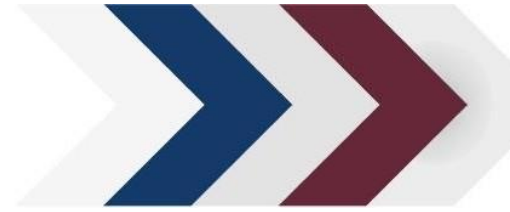
Las soluciones contienen elementos que necesita para crear la aplicación.

Una solución incluye uno o más proyectos, además de los archivos y metadatos que ayudan a definir una solución como un todo. Visual Studio genera una solución automáticamente cuando se crea un nuevo proyecto. Visual Studio almacena la definición de una solución en dos archivos: .sln y .suo.

El archivo de definición de soluciones (.sln) almacena los metadatos que definen la solución.

Link: ([https://msdn.microsoft.com/es-es/library/b142f8e7\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/b142f8e7(v=vs.110).aspx))






El proyecto contiene todo el material necesario para la aplicación: los archivos de código fuente, archivos de recursos, tales como iconos, referencias a archivos externos de los que depende la aplicación, y datos de configuración, tales como las opciones del compilador. Cuando se genera un proyecto, Visual C# invoca al compilador de C# y a otras herramientas internas para crear un ensamblado ejecutable con los archivos del proyecto.

Link: ([https://msdn.microsoft.com/es-es/library/ms173077\(v=vs.90\).aspx](https://msdn.microsoft.com/es-es/library/ms173077(v=vs.90).aspx))

Fijar u ocultar ventanas del IDE

Para fijar u ocultar deberá hacer click sobre el pin correspondiente de esa ventana .

Para buscar código escrito en mi proyecto:

Ctrl + F → Inicia un búsqueda.

F3 → repite la búsqueda.

Ctrl + H → Busca y reemplaza.

Variables

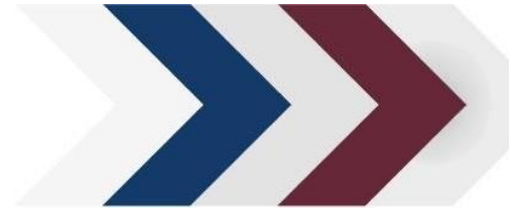
Una variable representa un valor numérico o de cadena o un objeto de una clase. El valor que la variable almacena puede cambiar, pero el nombre sigue siendo el mismo. Una variable es un tipo de *campo*. El código siguiente es un ejemplo sencillo de cómo declarar una variable de entero, asignarle un valor y, a continuación, asignarle un nuevo valor.

En C#, las variables se declaran con un tipo de datos y un identificador concretos.

En C# tiene que especificar si la variable es de tipo **int**, **float**, **byte**, **short** u otro cualquiera entre más de 20 tipos de datos diferentes. El tipo, especifica entre otras cosas, la cantidad de memoria exacta que se debe asignar para almacenar el valor cuando la aplicación se ejecuta.

El lenguaje C# fuerza ciertas reglas al convertir una variable de un tipo en otro.





Tipo de dato	Intervalo valores
Byte	0 .. 255
Sbyte	-128 .. 127
Short	-32,768 .. 32,767
Ushort	0 .. 65,535
Int	-2,147,483,648 .. 2,147,483,647
UInt	0 .. 4,294,967,295
Long	-9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807
Float	-3.402823e38 .. 3.402823e38
Double	-1.79769313486232e308 .. 1.79769313486232e308
Decimal	-79228162514264337593543950335 79228162514264337593543950335
Char	Un carácter Unicode.
String	Una cadena de caracteres Unicode.
Bool	True o False.
Object	Un objeto.

([https://msdn.microsoft.com/es-ar/library/wew5ytx4\(v=vs.90\).aspx](https://msdn.microsoft.com/es-ar/library/wew5ytx4(v=vs.90).aspx))

Nombrar una variable

- El nombre debe empezar por una letra
- El resto del nombre puede contener letras, números o caracteres subrayados , nose permiten espacios puntos u otros signos de puntuación
- El nombre deberá ser único dentro del alcance de la variable
- El nombre no puede ser una palabra reservada de C#

Normalmente el valor predeterminado para las variables es 0

El tipo de dato universal, es el object



Para cambiar en un formulario o Form

Accion	Propiedad
Los bordes	Formborderstyle
Poner o sacar botones	Maximizebox o minimizebox
Tamaño	Size
Posición	Location
Posición inicial	StartPosition
Folores	BackColor

Ingresar una fecha tipo calendario

1. Pegue en el form un controldatepicker
2. Pegue en el form un controlmonthcalendar



Cual es la diferencia?

Para activar un control de tiempo

A veces, conviene crear un procedimiento que se ejecuta a intervalos de tiempo específicos hasta que finaliza un bucle o que se ejecuta cuando ha transcurrido un intervalo de tiempo establecido. **El componente Timer** hace posible este procedimiento.

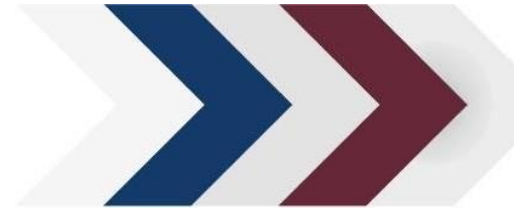
Pegue en el form un control timer. Este control sera invisible durante la ejecución.

El objeto Timer posee una propiedad llamada **Interval** la cual nos permite definir el tiempo que una acción se ejecutará periódicamente.

El objeto Timer posee un método **Tick** en el cual se debe escribir el código que deseamos que se ejecute de acuerdo a la frecuencia definida en Interval.

([https://msdn.microsoft.com/es-es/library/3tszykws\(v=vs.110\).aspx](https://msdn.microsoft.com/es-es/library/3tszykws(v=vs.110).aspx))

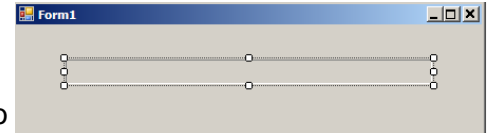




Para activar una barra de control de progreso

Colocar un control **progressbar** en el formulario. Se deben fijar las propiedades:

- ❖ **Máximum** : determina el valor máximo a representar
- ❖ **Minimum**: determina el valor mínimo a representar
- ❖ **Value**: es el valor que se muestra en la barra de progreso
- ❖ **Style**: determina la forma en que se muestra la actividad en segundo plano (investigue Marque)
- ❖ **Step**: es el valor que se establece para definir pasos de tamaño fijo

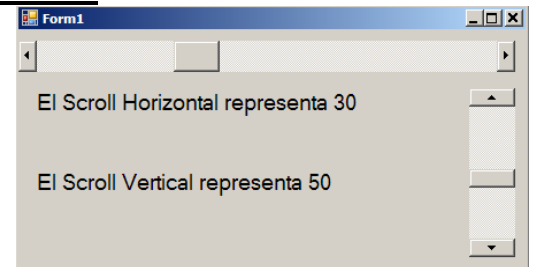


Value se puede definir en forma aleatoria de acuerdo a los cálculos que el programador desee mostrar.

Para colocar una barra de scroll o desplazamiento

Colocar un control Hscrollbar o Vscrollbar en el formulario y controlar las propiedades maximun, minimun y value del mismo.

Para ver en un label el valor de la posición donde está el scroll coloco en el código `label1.text = Hscrollbar.value;`



Para ver mensajes de texto ante una situación dada

Utilice el objeto MessageBox.

El método Show expone los parámetros asignados.

El objeto MessageBox tiene 21 sobrecargas que permiten gran flexibilidad a la hora de mostrar resultados.

`MessageBox.show("mensaje", "titulo del mensaje");`

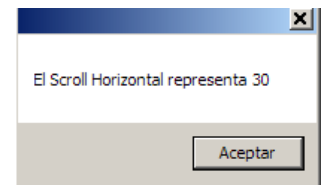
`MessageBox.show("mensaje", "titulo del mensaje", messageboxbuttons.yesno);`

`MessageBox.show("mensaje", "titulo del mensaje", messageboxbuttons.yesno, asterik);`

¿Qué diferencia existe entre ellos?

En el evento Load del formulario incluir el siguiente código:

```
hScrollBar1.Value = 30;
vScrollBar1.Value = 50;
label1.Text = "El Scroll Horizontal representa " + (this.hScrollBar1.Value).ToString();
label2.Text = "El Scroll Vertical representa " + (this.vScrollBar1.Value).ToString();
MessageBox.Show(label1.Text);
```





PARTE B

Objetivo:

Realizar ejercitación en C# con elementos conocidos, como lo son los condicionales IF y SWITCH..

Así mismo se incorporarán los objetos radiobutton, groupbox y checkbox para la implementación de estos temas.

Nota importante:

En este laboratorio tendremos varios formularios, es por eso que para agregar un formulario a nuestra solución deberemos ir al explorador de proyecto, pararnos sobre el nombre del proyecto, botón derecho, agregar, nuevo elemento, Windows form.

En el caso de la pantalla de bienvenida elegir a esta en lugar de Windows forms menú

Ingrese a C#, cree un proyecto y realice el siguiente formulario

Agregue los buttons y radiobuttons de acuerdo al siguiente diseño

Las propiedades que modificaremos en los controles serán

control	Text	Name
Form	Calculadora con radioboton	frmRadioButton
Button1	Limpiar	btnLimpiar
Button2	Con if	btnIf
Button3	Con Switch	btnSwitch
Button4	Salir	btnSalir
Label1	1er operador	lbl1
Label2	2do operador	lbl2





Label3	Resultado	Lbl3
Textbox1		Txt1
Textbox2		Txt2
Textbox3		TxtRta
GroupBox	Operaciones	GroupBox1
Radiobutton1	Suma	Optsuma
Radiobutton2	Resta	Optresta
Radiobutton3	Producto	Optprod
Radiobutton4	cociente	OptCoc

En el button **Limpiar** colocaremos el siguiente código (**ACLARAR PORQUE USAMOS THIS**)

```
private void btnLimpiar_Click(object sender, EventArgs e)
{
    this.Txt1.Text = null;
    this.Txt2.Text = null;
    this.TxtRta.Text = null;
}
```

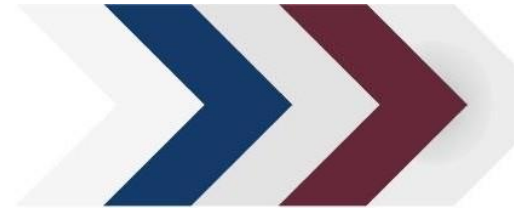
En el button **con if** colocaremos el siguiente código

```
private void btnIf_Click(object sender, EventArgs e)
{
    if (this.Optresta.Checked)
    {
        this.TxtRta.Text = Int32.Parse(Txt1.Text)Int32.Parse(Txt2.Text)).ToString();
    }

    if (this.OptSuma.Checked)
    {
        this.TxtRta.Text = (Int32.Parse(Txt1.Text)-Int32.Parse(this.Txt2.Text)).ToString();
    }

    if (OptCoc.Checked)
    {
        if (Int32.Parse(this.Txt2.Text) != 0)
        {
            this.TxtRta.Text = (Int32.Parse(this.Txt1.Text) /Int32.Parse(this.Txt2.Text)).ToString();
        }
    }
}
```





```

        if (OptProd.Checked)
        {
            this.TxtRta.Text = (Convert.ToInt32(this.Txt1.Text) * Convert.ToInt32(this.Txt2.Text)).ToString();
        }

        if (this.OptSuma.Checked)
        {
            this.TxtRta.Text = (Convert.ToInt32(this.Txt1.Text) + Convert.ToInt32(this.Txt2.Text)).ToString();
        }
    }

```

En el botón **con Switch** colocaremos el siguiente código

```

private void btnswitch_Click(object sender, EventArgs e)
{
    bool mayor_cero = true; int opcion = 0;
    if (this.Txt1.Text != null && this.Txt2.Text != null)
    {
        if (Int32.Parse(this.Txt2.Text) != 0)
        {
            mayor_cero = true;
        }
        else
        {
            mayor_cero = false;
        }

        if (this.OptSuma.Checked) opcion = 1;
        if (this.Optresta.Checked) opcion = 2;
        if (this.OptProd.Checked) opcion = 3;
        if (this.OptCoc.Checked)
        {
            if (mayor_cero) opcion = 4;
        }
        switch (opcion)
        {
            case 1:
                this.TxtRta.Text = (Int32.Parse(Txt1.Text) + Int32.Parse(this.Txt2.Text)).ToString();
                break;
            case 2:
                this.TxtRta.Text = (Int32.Parse(Txt1.Text) - Int32.Parse(Txt2.Text)).ToString();
                break;
            case 3:
                this.TxtRta.Text = (Int32.Parse(this.Txt1.Text) * Int32.Parse(this.Txt2.Text)).ToString();
                break;
            case 4:
                this.TxtRta.Text = (Int32.Parse(this.Txt1.Text) / Int32.Parse(this.Txt2.Text)).ToString();
                break;
            default:
                string mensaje = "La operación no se puede realizar";
                string titulo = "Importante";
                MessageBoxButtons botones = MessageBoxButtons.YesNo;
                DialogResult resultado;

```



```

        resultado = MessageBox.Show(mensaje, titulo, botones);
        if (resultado == System.Windows.Forms.DialogResult.Yes)
        {
            limpiar();
        }
        break;
    } // cierra switch
} // cierra if
} // cierra método

```

A partir de aquí el nombre de los objetos deberán ser deducidos por ustedes del código que sigue al diseño de los mismos

Muy bien. Finalizado este formulario agregaremos otro a nuestro proyecto con el siguiente diseño.

Los dos radiobutton se encuentran insertados dentro de respectivos groupbox.

Así podremos tenerlos simultáneamente seleccionados y el resultado no lo mostraremos en un textbox sino en un listbox.

Atencion !!! El objeto lst1 no es un textbox agrandado, sino un objeto listbox seleccionado del cuadro de herramientas.

Dentro del button mostrar codificaremos lo siguiente



```
if (opt1.Checked == true)
{
    lst1.Items.Add(textBox1.Text);opt1.Checked = false;
}
if (opt2.Checked == true)
{
    lst1.Items.Add(textBox2.Text);opt2.Checked = false;
}
```

Dentro del button Limpiar codificaremos lo siguiente

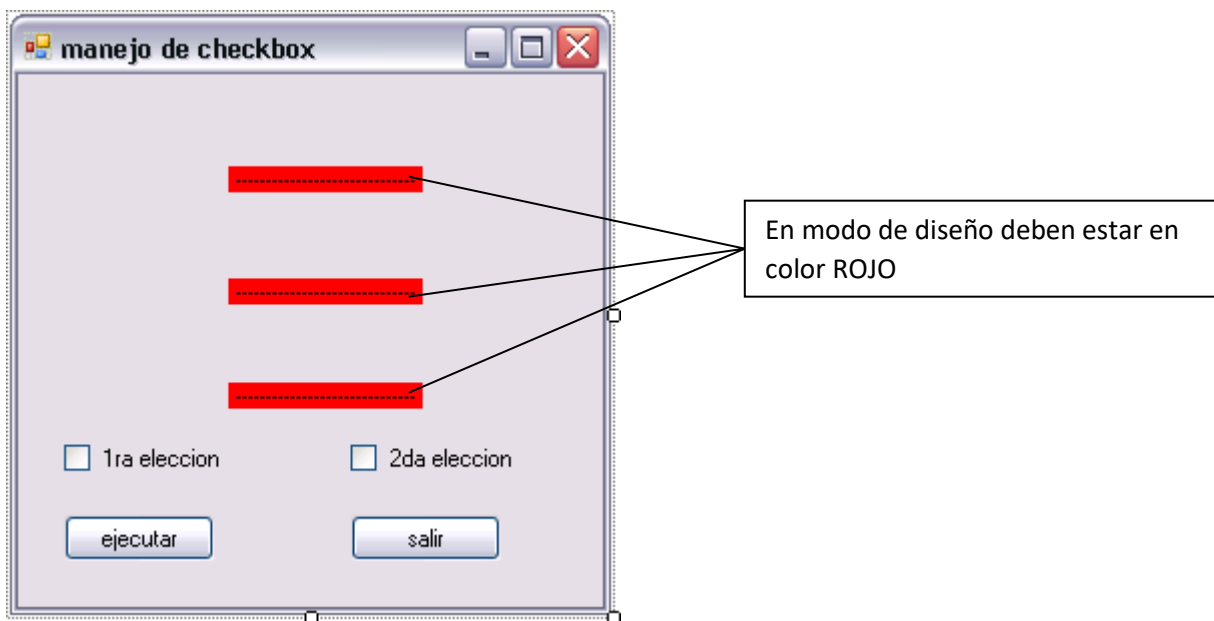
```
lst1.Items.Clear();
```

Quedando el código del boton Salir:

```
close();
```

Por ultimo generaremos otro formulario donde trabajaremos con el control checkbox , este nos permitirá sin necesidad de tener los groupbox , seleccionar varias opciones a la vez.

El diseño de este nuevo formulario será el siguiente:



los controles y propiedades a modificar seran

control	text	name	backcolor
Button1	Ejecutar	btnejecutar	
Button2	Salir	btnsalir	

Checkbox1	1ra elección	Chk1	
Checkbox2	2da elección	Chk2	
Label1		Lbl1	Rojo
Label2		Lbl2	Rojo
Label3		Lbl3	Rojo

Escribamos ahora en el botón ejecutar lo siguiente

```

if ((this.Chk1.Checked == true) & (this.Chk2.Checked == false))
{
    Lbl1.BackColor = Color.Aquamarine;
    Lbl2.BackColor = Color.Beige;
    Lbl3.BackColor = Color.Black;
}

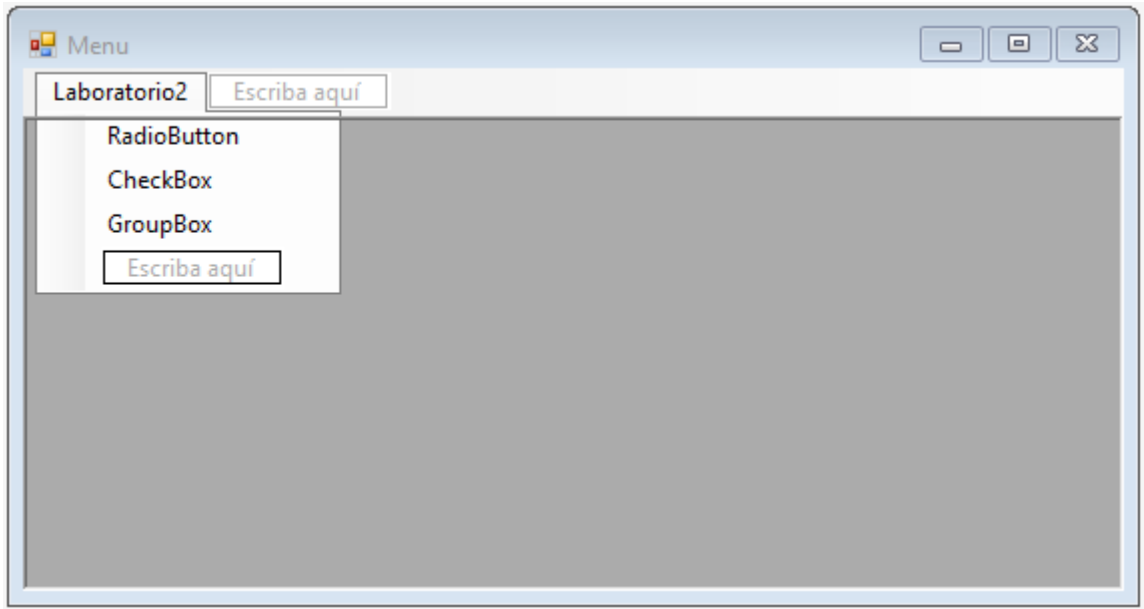
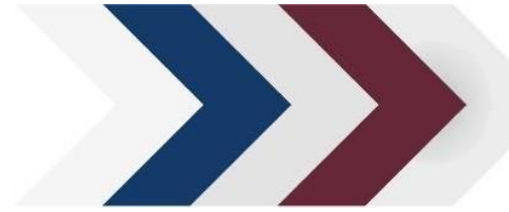
if ((Chk1.Checked == false) & (Chk2.Checked == true))
{
    Lbl1.BackColor = Color.Green;
    Lbl2.BackColor = Color.LightPink;
    Lbl3.BackColor = Color.Linen;
}

if ((Chk1.Checked == true) & (Chk2.Checked == true))
{
    Lbl1.BackColor = Color.Blue;
    Lbl2.BackColor = Color.White;
    Lbl3.BackColor = Color.Blue;
}

```

Por ultimo veremos de unir todos estos formularios con un menú

Para ellos incorporaremos un formulario MDI (contenedor) al que llamaremos Menu, para ello enen propiedades **IsMdiContainer = true**, y al que le agregaremos un control **menustrip** donde codificaremos en cada uno de los ítems del menustrip el nombre del formulario que se desea mostrar



En cada submenú se escribe el siguiente código

```
//Creo el formulario que deseo mostrar
frmRadioButton ofrmRadioButton = new frmRadioButton();

//Digo que es Hijo del MDI
ofrmRadioButton.MdiParent = this;

//Muestro el form
ofrmRadioButton.Show();
```

En forma genérica

```
//Creo el formulario que deseo mostrar
Nombre_del_formulario_real Nombre_elegido = new Nombre_del_formulario_real();

//Digo que es Hijo del MDI
Nombre_elegido.MdiParent = this;

//Muestro el form
Nombre_elegido.Show();
```

Si quiero armar una pantalla de Bienvenida en el MDI o cualquier form.
Voy a la propiedad backgroundImage y selecciono la imagen que deseo

