

**PROGRAMACIÓN Y ESTRUCTURAS DE DATOS**

TP INICIAL

LABORATORIOS #2

**unidad 1   
Introducción a la Programación Visual**

**Autor de contenidos:**Nicolás Battaglia

**OBJETIVOS**

Práctica guiada de laboratorio

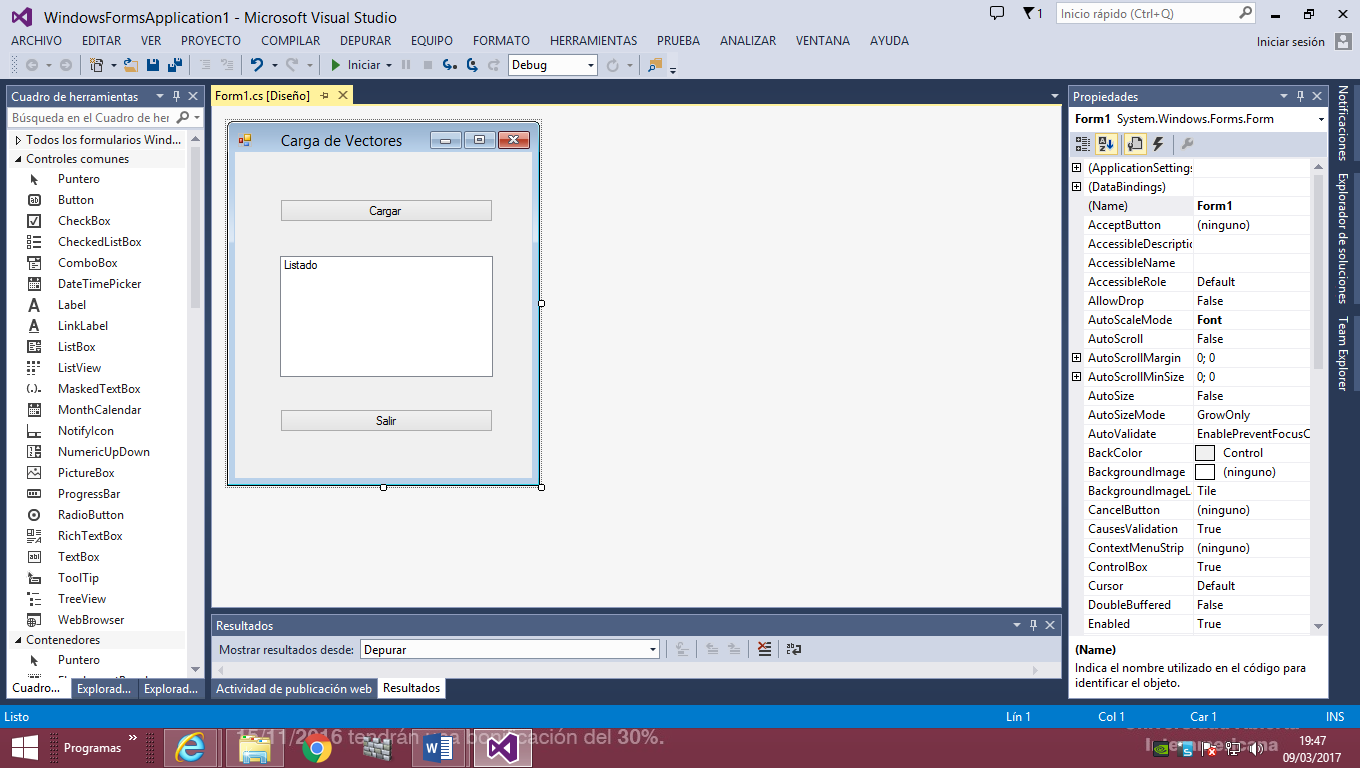
**ENUNCIADO**

**El siguiente TP se divide en 3 partes. En cada una de ellas encontrará una guía para realizar las prácticas de laboratorio**

**PARTE A**

**Objetivo**

En esta clase veremos la forma de cargar y operar las estructuras en memoria de tipo arrays, uni y multidimensionales. Para eso ingresamos en el Visual Studio y generamos un nuevo proyecto en C#, y luego generemos un formulario con el siguiente diseño.



Los controles a modificar serán:

|  |  |  |
| --- | --- | --- |
| control | text | Name |
| Button1 | cargar | btnejecutar |
| Button2 | salir | btnsalir |
| Listbox1 |  | Lst1 |

**Nota:** para poder utilizar las funciones de consola dentro de una aplicación de formulario, es necesario agregar las siguientes líneas:

public Form1()// Constructor

{

AllocConsole();//Agregar esta línea dentro del constructor

InitializeComponent();

}

//Agregar estas dos líneas seguido al constructor

[System.Runtime.InteropServices.DllImport("kernel32.dll")]

private static extern bool AllocConsole();

En el botón cargar codificaremos lo siguiente:

private void button1\_Click(object sender, EventArgs e)

{

string dato;

int[] vector; //declaro mi vector

vector = new int[10];//inicializo mi vector

//recorro con el ciclo for las 10 posiciones del vector para cargar el mismo

int i;

for (i = 1; i < 10; i++)

{

Console.WriteLine("Ingrese un valor: ");//Imprimo en pantalla

dato = Console.ReadLine();//Guardo en mi variable dato el valor ingresado por consola

vector[i] = Int32.Parse(dato);//guardo el dato en el vector, en la posición i(i toma el valor del bucle)

}

Console.WriteLine("Los datos del vector fueron impresos");

//Recorro nuevamente el vector, para cargar los datos en mi listBox

for (i = 1; i < 10; i++)

{

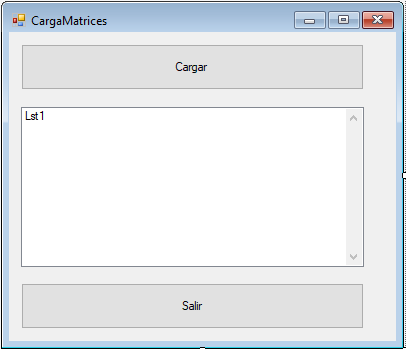
Lst1.Items.Add("En la posición : " + i + " el valor es: " + vector[i]);

}

}

Luego de este formulario , agreguemos otro donde con iguales controles que el anterior que nos permitira cargar una matriz y listarla

## Matrices



string dato;

int[,] matriz; //declaro mi vector

matriz = new int[3,4];//inicializo mi matriz

//recorro la matriz anidando dos ciclos for

int i,j;

for (i = 0; i < 3; i++)//con este bucle recorro las filas

{

for (j = 0; j < 4; j++)//con este bucle recorro las columnas

{

Console.WriteLine("Ingrese un valor: ");//Imprimo en pantalla

dato = Console.ReadLine();//Guardo en mi variable dato el valor ingresado por consola

matriz[i, j] = Int32.Parse(dato);

}

}

Console.WriteLine("Los datos de la matriz fueron impresos");

//Recorro nuevamente la matriz, para cargar los datos en mi listBox

for (i = 0; i < 3; i++)

{

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

{

int ver = matriz[i, j];

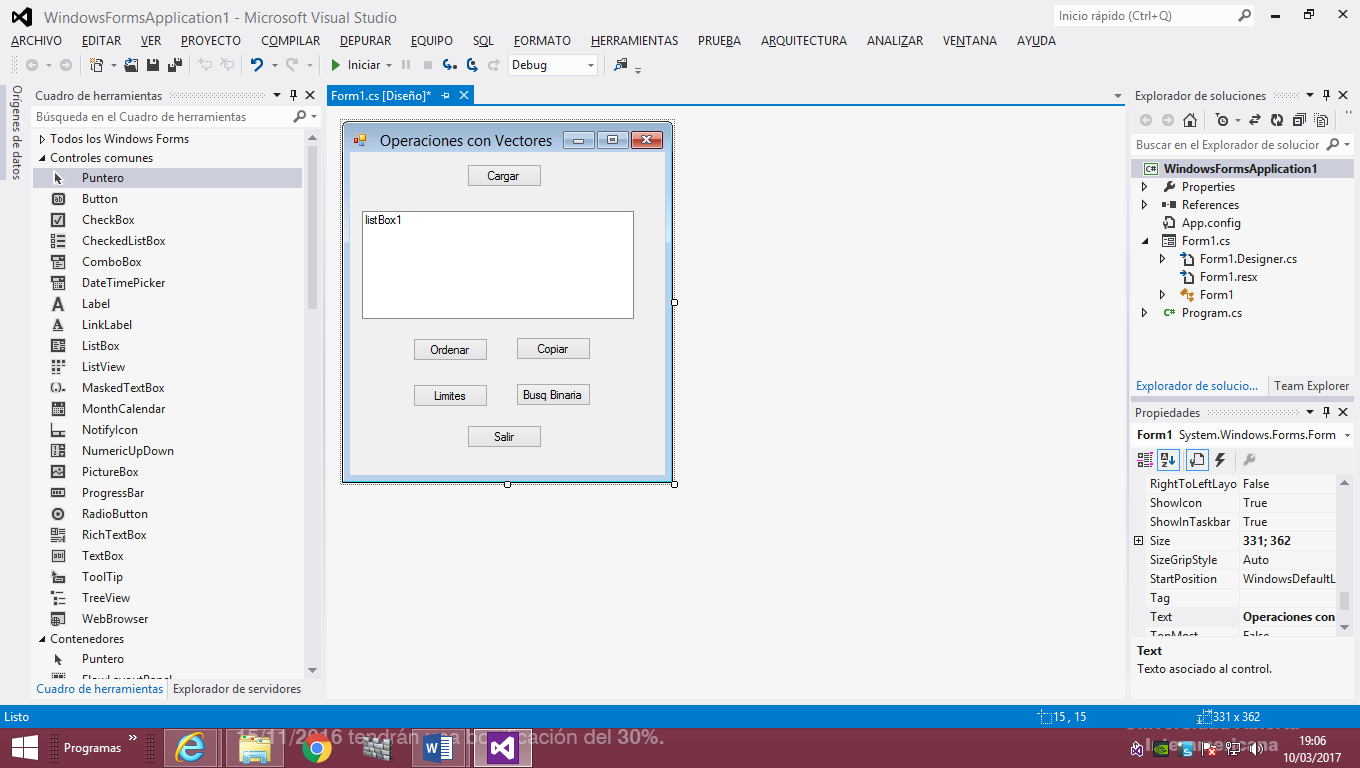
Lst1.Items.Add("En la fila: " + i + " columna:" + j + " el valor es: " + matriz[i,j]);

}

}

Por ultimo genere el formulario splashscreen correspondiente.

## Operaciones con vectores



Escribamos el siguiente código en el botón correspondiente

Para que nuestro vector sea visible en todo nuestro form lo definiremos en el general de la class

public partial class Operacionesvectoress : Form

{

string dato;

int[] vector; //declaro mi vector

int i;

Botón Cargar:

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

{

Lst1.Items.Clear(); //limpio los items de la lista

vector = new int[10];//inicializo mi vector

//recorro con el ciclo for las 10 posiciones del vector para cargar el mismo

for (int i = 1; i < 10; i++)

{

Console.WriteLine("Ingrese un valor: ");

dato = Console.ReadLine();

vector[i] = Int32.Parse(dato);

}

Console.WriteLine("Los datos del vector fueron impresos");

//Recorro nuevamente el vector, para cargar los datos en mi listBox

for (int i = 0; i < 10; i++)

{

Lst1.Items.Add("En la posición : " + i + " el valor es: " + vector[i]);

}

}

Botón ordenar:

{

Lst1.Items.Clear();

Array.Sort(vector);//esta instrucción ordena el vector

for (int i = 0; i < 10; i++)

{

Lst1.Items.Add("El valor es: " + vector[i] + " en la posición : " + i);

}

}

Botón Copiar:

{

int[] vectorDestino = new int[20];

Array.Copy(vector, 0, vectorDestino, 0, 10);

for (int i = 0; i < 10; i++)

{

Lst1.Items.Add(vectorDestino[i] + " en la posición " + i);

}

}

Botón Búsqueda binaria:

**NOTA:** El algoritmo de búsqueda binaria es un excelente método para buscar datos dentro de una estructura (generalmente un arreglo unidimensional). Se le da el nombre de búsqueda binaria por que el algoritmo divide en dos el array, aludiendo al concepto de bit, el cual puede tener dos estados.

La única condición para usar este algoritmo es que los datos dentro del array estén ordenados de menor a mayor.

{

int posicion;

posicion = Array.BinarySearch(vector, 9);

Lst1.Items.Clear();

Lst1.Items.Add("En la posicionposición " + posicion + " se encuentra el número 9");

}

Botón Limites:

{

int superior;

int inferior;

listBox1.Items.Clear();

superior = vector.GetUpperBound(0);//Obtiene el índice del último elemento de la dimensión especificada en la matriz.

inferior = vector.GetLowerBound(0);// Obtiene el índice del primer elemento de la dimensión especificada en la matriz.

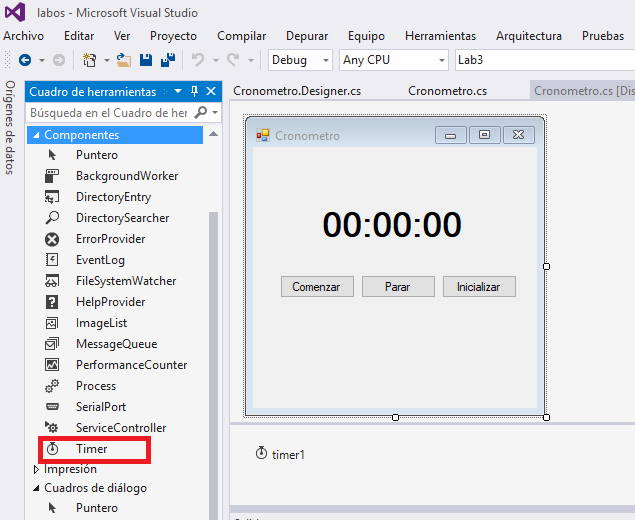
Lst1.Items.Add("El límite inferior es " + inferior + " y el superior es " + superior);

}

## Timer

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.

Para eso ingresamos en el Visual Studio y generamos un nuevo proyecto en C#, y luego generemos un formulario con el siguiente diseño, al cual le añadiremos el componente timer arrastrándolo desde el cuadro de herramientas.



Los controles a modificar serán:

|  |  |  |
| --- | --- | --- |
| Control | text | Name |
| Button1 | Comenzar | btnComenzar |
| Button2 | Parar | btnParar |
| Button3 | Inicializar | btnIniciar |
| Label1 |  | Label1 |

En la clase general inicializo las siguientes variables

public partial class Cronometro : Form

{

int hora =0 , min=0, seg=0;

En el botón comenzar:

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

{

timer1.Enabled = true; // habilito el timer

}

En el botón de parar:

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

{

timer1.Enabled = false;

}

En el Botón Inicializar

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

{

timer1.Enabled = false;

label1.Text = "00:00:00";

seg = 0;

min = 0;

hora = 0;

}

El Timer

private void timer1\_Tick(object sender, EventArgs e)

{

seg++;

if (seg == 60)

{

min++;

seg = 0;

}

else if (min == 60)

{

hora++;

min = 0;

}

label1.Text = hora.ToString().PadLeft(2, '0') + ":" + min.ToString().PadLeft(2, '0') + ":"+ seg.ToString().PadLeft(2, '0');

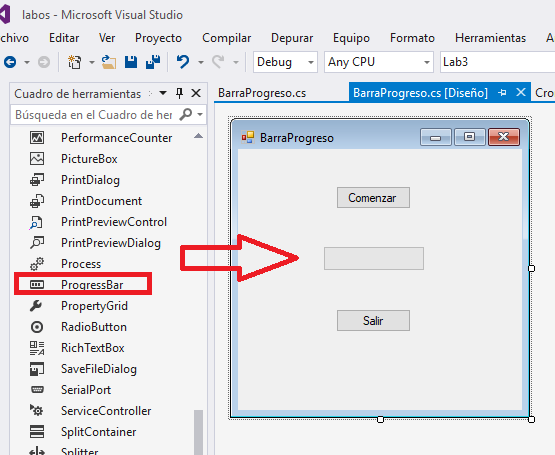
}

/\* Método String.PadLeft (Int32, Char)

Devuelve una nueva cadena que alinea a la derecha los caracteres de la instancia e inserta a la izquierda un carácter Unicode especificado

hasta alcanzar la longitud total especificada. \*/

## Barra de Progreso



En el botón de Comenzar

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

{

long contador;

progressBar1.Value = 0;//inicializo el valkor inicial del progresbar

progressBar1.Maximum = 100;//valor maximo para el progressbar

while ((progressBar1.Value < progressBar1.Maximum))

{

progressBar1.Value++;

// Bucle para simular el delay, esto es mejor hacerlo con hilos

for (contador = 1; (contador <= 1000000); contador++)

{

}

}

}

Botón de Salir

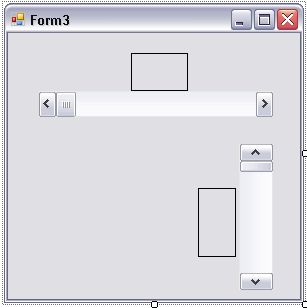
private void btnSalir\_Click(object sender, EventArgs e)

{

Application.Exit();

}

## Barras de desplazamiento



Los controles a modificar serán:

|  |  |  |
| --- | --- | --- |
| Control | text | Name |
| hScrollBar1 |  | hScrollBar1 |
| vScrollBar1 |  | vScrollBar1 |
| label1 |  | label1 |
| label2 |  | label2 |

en los ScrollBar escribimos lo siguiente:

private void hScrollBar1\_Scroll(object sender, ScrollEventArgs e)

{

this.label1.Text = hScrollBar1.Value.ToString();

}

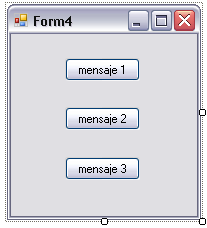
private void vScrollBar1\_Scroll(object sender, ScrollEventArgs e)

{

label2.Text = vScrollBar1.Value.ToString();

}

## Mensajes de Texto



La idea de este ejercicio, es que jueguen con las distintas opciones de MessageBox

private void btnMsj1\_Click(object sender, EventArgs e)

{

MessageBox.Show("mensaje", "titulo del mensaje");

}

private void btnMsj2\_Click(object sender, EventArgs e)

{

MessageBox.Show("mensaje", "titulo del mensaje", MessageBoxButtons.YesNo);

}

private void btnMsj3\_Click(object sender, EventArgs e)

{

MessageBox.Show("mensaje", "titulo del mensaje", MessageBoxButtons.YesNo, MessageBoxIcon.Asterisk);

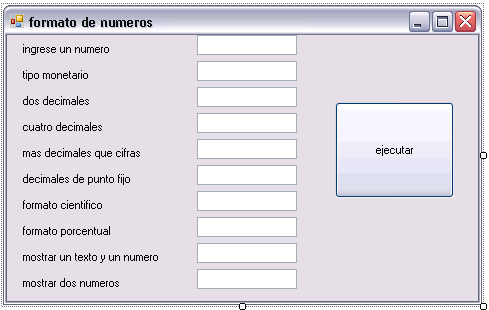
}

**PARTE B**

**Objetivo:**

En esta clase veremos las formas de convertir y modificar formatos numéricos, de tipo fecha y cadenas.

Realicemos el diseño según el siguiente formulario:



En este formulario veremos cómo dar formato y convertir valores numéricos, para lo cual codificaremos en el botón Ejecutar lo siguiente

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

{

double auxiliar;

double auxiliar1;

double auxiliar2;

auxiliar = double.Parse(txtIngreso.Text);

textBox1.Text = string.Format("{0:c}", auxiliar);

auxiliar1 = (auxiliar / 100);

textBox2.Text = string.Format("{0:N}", auxiliar);

textBox3.Text = string.Format("{0:N4}", auxiliar);

textBox4.Text = string.Format("{0:D8}", Convert.ToInt32(auxiliar));

textBox5.Text = string.Format("{0:f3}", auxiliar);

textBox6.Text = string.Format("{0:E3}", auxiliar);

textBox7.Text = string.Format("{0:p}", auxiliar);

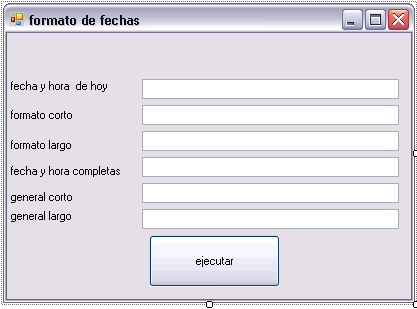
textBox8.Text = string.Format("el valor es {0:c}", auxiliar);

auxiliar2 = (auxiliar + 10);

textBox9.Text = string.Format("{0:c} {1:c}", auxiliar, auxiliar2);

}

Terminado este formulario realizaremos un segundo formulario donde aprenderemos a tomar el valor de la fecha y hora del sistema y darle distintos formatos según sea necesario



En el botón Ejecutar escribiremos:

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

{

DateTime Fecha = DateTime.Now;

textBox1.Text =Convert.ToString(Fecha);

//Fecha Corta

textBox2.Text = string.Format("{0:d}",Fecha);

//Fecha Larga + tiempo

textBox3.Text = string.Format("{0:D}", Fecha);

//Fecha larga + tiempocorto

textBox4.Text = string.Format("{0:f}", Fecha);

//Fecha corta + tiempo corto

textBox5.Text = string.Format("{0:g}", Fecha);

//Fecha corta + tiempo largo

textBox6.Text = string.Format("{0:G}", Fecha);

}

## 

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

{

//asigno el valor a la variable del texbox1

string variable = textBox1.Text;

//asigno al texbox2 el tamaño de la palabra ingresada

textBox2.Text = variable.Length.ToString();

//si en el txtdato es > 0, devuelvo, el caracter en la posicion indicada

if ((txtdato.Text.Length > 0))

{ int pos = Convert.ToInt32(txtdato.Text);

textBox3.Text = (variable[pos]).ToString();

}

//a la cadena le inserto 12345

textBox4.Text = variable.Insert(3, "12345");

//quito los espacion en blanco la final de la cadena

textBox5.Text = variable.TrimStart();

//

if ((variable.Length < 10))

{

//le agrego 0 a la derecha si no posee 10 caracteres, hasta llegar a ellos

// FORMA DE HACERLO ELEGANTE

textBox7.Text = variable.PadRight(10,'0');

}

//conateno la cadena con el textbox 9

textBox8.Text = string.Concat(textBox9.Text, variable);

if (textBox6.Text == "ABC")

{

textBox6.Text = "Si";

}

else

{

textBox6.Text = "No";

}

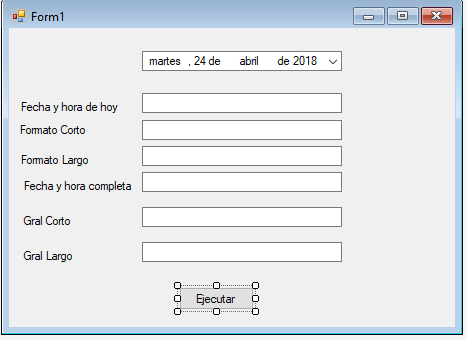
}

**PARTE C**

**Objetivo:**

En esta clase seguiremos trabajando con el tipo de datos fecha y realizaremos también algunas operaciones con ellas.

Realicemos el diseño según el siguiente formulario:



En el botón Ejecutar escribiremos:

private void button1\_Click(object sender, EventArgs e)

{

DateTime Fecha = this.dateTimePicker1.Value;

textBox1.Text = Convert.ToString(Fecha);

//Fecha Corta

textBox2.Text = string.Format("{0:d}", Fecha);

//Fecha Larga + tiempo

textBox3.Text = string.Format("{0:D}", Fecha);

//Fecha larga + tiempocorto

textBox4.Text = string.Format("{0:f}", Fecha);

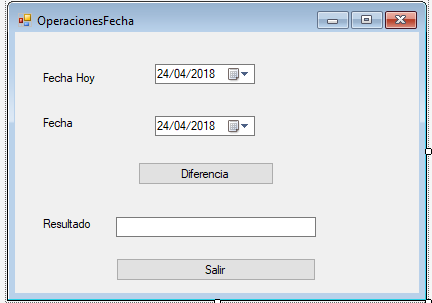
//Fecha corta + tiempo corto

textBox5.Text = string.Format("{0:g}", Fecha);

//Fecha corta + tiempo largo

textBox6.Text = string.Format("{0:G}", Fecha);

}

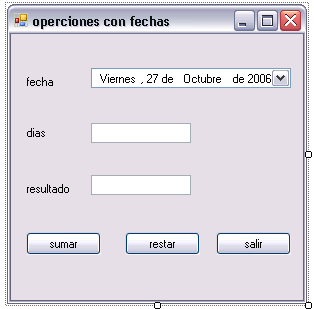


El código a colocar en el botón Diferencia sería

DateTime fecha1 = this.dateTimePicker1.Value;

DateTime fecha2 = this.dateTimePicker2.Value;

textBox1.Text = fecha1.Subtract(fecha2).ToString();



El código a colocar en Suma sería

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

{

DateTime fecha1 = new DateTime(dateTimePicker1.Value.Year, dateTimePicker1.Value.Month, dateTimePicker1.Value.Day);

txtresultado.Text = (fecha1.AddDays(Convert.ToInt32(txtdias.Text)).ToString("dd/MM/yyyy")).ToString();

}

Y el código a colocar en Resta sería

DateTime fecha1 = new DateTime(dateTimePicker1.Value.Year, dateTimePicker1.Value.Month, dateTimePicker1.Value.Day);

txtresultado.Text = (fecha1.AddDays(-1\* Convert.ToInt32(txtdias.Text)).ToString("dd/MM/yyyy")).ToString();