Introducción a la Programación en **C#**

Introducción.

Programa: Conjunto de instrucciones que entiende un ordenador para realizar una actividad.
Para la resolución de un problema hay que plantear un algoritmo.
Algoritmo: Son los pasos a seguir para resolver un problema.
Pseudocódigo: Escribimos los pasos del algoritmo en borrador en nuestro lenguaje general común.
Diagrama de flujo: es la representación gráfica de un ALGORITMO. Resulta mucho más fácil entender un gráfico.

Tipos y diferencias entre C, C++, C#

- **C** creado en 1972, lenguaje más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones. Se trata de un lenguaje de *medio nivel* pero con muchas características de *bajo nivel*. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C.
- **C++** es un lenguaje de los años 1980. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido: (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.
- **C#** (pronunciado "Si Sharp" en inglés) es un lenguaje de programación orientado a objetos desarrollado por Microsoft para su plataforma .NET. Su sintaxis básica deriva de C/C++ . El nombre C Sharp fue inspirado por la notación musical, sugiriendo que C# es superior a C/C++.

Código: escritura de las instrucciones del programa en un lenguaje de programación.

- **Función**: El objeto de función es para poder dividir un programa grande en un subconjunto de programas o funciones más pequeñas.
- **Función Main**: Función principal que puede iniciarse con la siguiente estructura: *void main (void) o Int main (int x)*
- **Palabras clave**: Son palabras reservadas por el programa y no podemos emplear como nombres de identificadores.
- **Identificadores**: Nombre de una función, variable o constante. No puede contener espacios en blanco, acentos ni caracteres extraños. Distingue mayúsculas de minúsculas. No puede empezar por un número.
- **Comentarios**: /* varias líneas */ o // hasta final de línea
- **Operador de visibilidad ::** Permite acceder a una variable global cuando está oculta por otra local.

Plataforma .NET: Es un conjunto de tecnologías diseñadas para intercomunicarse y transformar a Internet en una plataforma de cómputo distribuido. Proporciona Interfaces de programación y herramientas para diseñar, crear, ejecutar y distribuir soluciones para la plataforma .NET, por ejemplo, Visual Studio.
El ***.NET Framework*** proporciona un entorno de ejecución de aplicaciones Common Language Runtime o ***CLR***. Este entorno permite ejecutar las aplicaciones .NET e interactuar con el sistema operativo, ofreciendo sus servicios y recursos a estas aplicaciones .NET. Este entorno es común a todos los lenguajes .NET
El .NET Framework proporciona 3 elementos principales, el Common Language Runtime o motor en tiempo de ejecución común para todos los lenguajes .NET, El .Net Framework Class Library o biblioteca de clases base del .NET Framework y una Colección de Frameworks de desarrollo.

Plantillas de Visual Studio
Proporcionan el código inicial para construir y crear rápidamente aplicaciones.

- *Console Application*: Para desarrollar una aplicación que se ejecute en una interfaz de línea de comandos.
- *Windows Forms Application*: código inicial para desarrollar una aplicación gráfica Windows Form.
- *WPF Application*: código inicial para desarrollar una aplicación Windows rica en interfaz de usuario.
- *Blank App* (Universal Windows): código inicial para desarrollar una aplicación de la Plataforma Universal de Windows.
- *Blank App* (Universal Windows 8.1): para desarrollar una aplicación Universal para Windows y Windows Phone
- *Class Library*: código inicial para desarrollar una biblioteca de clases .dll. uerer invocar desde otra aplicación.
- *ASP.NET Web:* Application (.NET Framework): desarrollar aplicaciones ASP.NET como Web Forms, MVC o Web API.

XAML:
Extensible Application Markup Language: utiliza elementos y atributos para definir controles en sintaxis XML compatibles con aplicaciones .NET, plataforma Universal de Windows (UWP) o para desarrollar aplicaciones para iOS y Android con Xamarin.

Modo consola: Ordenes de entradas y salidas desde consola (i/o)

Mostrar mensajes en pantalla:

- En C#: utilizamos el objeto "**Console**": ***Console.Write****("Ingrese Horas trabajadas por el operario:");*
- En C: utilizamos **printf**("entre comillas fijo"); sin comillas variable
- En C++: podemos utilizar la función cin de la librería iostream: **cout** << "Hola " << endl;

Entrada de datos por teclado:

- En C#: Debemos definir una variable de tipo string que la llamaremos linea: string linea;
  Luego cada vez que necesitemos ingresar por teclado un conjunto de caracteres utilizaremos la función ***ReadLine*** del objeto Console con la siguiente sintaxis: linea = ***Console.ReadLine();***
  Luego poner el contenido de la variable linea en una variable de tipo int: horasTrabajadas = int.Parse(linea);
- En C: Usar: **scanf**("%d",&horasTrabajadas);
- En C++: podemos utilizar la función cin de la librería iostream: **cin**>>opcion;

Creación de un proyecto en C# (C sharp) desde consola

Pedir horas y coste/hora y mostrar el sueldo

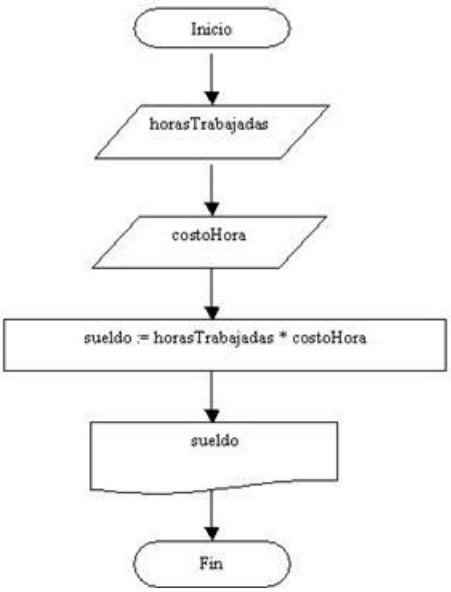
Pasos para la creación de un proyecto en C# en Microsoft Visual Studio Express:

1. Entramos en "Microsoft Visual C# 2013 Express".
2. Para la creación del proyecto. Escogemos desde el menú la opción "Archivo" -> "***Nuevo proyecto...***"
   Aparece un diálogo donde debemos indicar el nombre del proyecto y seleccionar el tipo de proyecto (elegiremos "***Aplicación de consola***" y pondremos como nombre al proyecto "CalculoSueldo".
   Podemos ver que el entorno nos genera automáticamente el esqueleto del programa.
   Para probar el funcionamiento del programa debemos presionar el ícono con un triángulo verde

Primero vamos a definir tres variables: (horasTrabajadas, costoHora, sueldo). La cantidad de horas normalmente será un valor entero (integer) , pero el costo de la hora es muy común que sea un valor decimal (coma flotante o float) y como el sueldo resulta de multiplicar las horas trabajadas por el costo por hora el mismo deberá ser decimal.
La definición de las variables la hacemos en la Main:

int horasTrabajadas; float costoHora, sueldo;

las palabras clave en minúsculas y el nombre de la variable, por ejemplo: horasTrabajadas (se propone que el nombre de la variable comience con minúscula y en caso de estar constituida por dos palabras o más palabras poner en mayúsculas el primer carácter (un nombre de variable no puede tener espacios en blanco, empezar con un número, ni tampoco utilizar caracteres especiales)
Utilizar nombres de variables “amigables” que indiquen lo que representan.



Programación en C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CalculoSuedo
{
 class Program
 {
   static void Main(string[] args)
    {
     int horasTrabajadas;
     float costoHora;
     float sueldo;
     string linea;
     Console.Write("Horas trabajadas:");
     linea = Console.ReadLine();
     horasTrabajadas = int.Parse(linea);
     Console.Write("Cote por hora:");
     linea = Console.ReadLine();
     costoHora = float.Parse(linea);
     sueldo = horasTrabajadas * costoHora;
     Console.Write("El sueldo total del operario:");
     Console.Write(sueldo);
     Console.ReadKey();
    }
  }
}
```

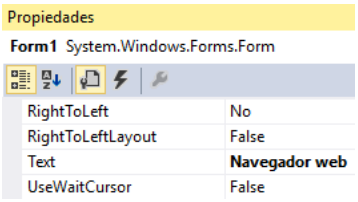
Programación en C o C++

```
#include <stdio.h>                    //ok edu
main ()
{
   int horasTrabajadas;    //se declaran las variables
   float costoHora, sueldo;  // 2 variables en misma linea
   costoHora, sueldo=0; //se inicializan las variables
   printf("Horas trabajadas por el operario:");
   scanf("%d",&horasTrabajadas);
   printf("Coste por hora:");
   scanf("%f",&costoHora);
   sueldo = horasTrabajadas * costoHora;
   printf("El sueldo total del operario es: %f",
   sueldo);
   printf("\n");
   printf ("Pulsa RETURN para terminar.");
   scanf("%d");
}

/*%c: formato caracter %d: formato entero %f:
formato decimal flotante. (introducir los decimales
con el punto decimal no la coma) */
```

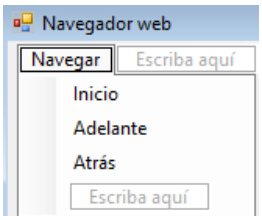
Crear una aplicación de formularios Windows Forms en C#

- En el menú Archivo, haga clic en ***Nuevo proyecto***.
- Aparecerá el cuadro de diálogo *Nuevo proyecto* con diferentes tipos de aplicaciones que puede crear.
- Seleccione ***Aplicación de Windows Forms*** como tipo de proyecto.
- Active la casilla: ✓ *Crear directorio para la solución* y cambie el nombre de la aplicación a *Navegador.* Aceptar.
- Se mostrará en la vista Diseñador un formulario o ventana de Windows vacía, titulada *Form1*.
- En la ***vista Diseño***, puede arrastrar diversos controles desde el *Cuadro de herramientas* hasta el formulario. Estos controles no están realmente "activos", Visual C# en segundo plano, crea el código para que el control real ocupe la posición correcta cuando se ejecute el programa. Este código fuente de diseño se encuentra en el archivo Form1.designer.cs.



Cambiar el título del formulario Windows.

- Haga clic en el formulario para seleccionarlo.
- Active la ventana *Propiedades* desde el menú: *Ver – Ventana de propiedades*
- En la ventana *Propiedades*, desplácese hacia abajo hasta *Text*, seleccione el texto "Form1" y escriba *Navegador web.*
- Active el *Cuadro de herramientas*. Desplácese hacia abajo por la lista de controles y expanda *Menús y barras de herramientas* hasta que vea *MenuStrip*. Arrastre este control a cualquier lugar del formulario Windows.
  Este control crea un menú predeterminado en la parte superior del formulario.
- En el cuadro que dice *Escriba aquí*, escriba ***Navegar***. Cuando presione ENTRAR, aparecerán nuevos cuadros vacíos para crear más elementos de menú. En el cuadro inferior, escriba ***Inicio***. Presione *ENTRAR* y aparecerán más cuadros. Escriba ***Adelante***. Presione ENTRAR y escriba ***Atrás***

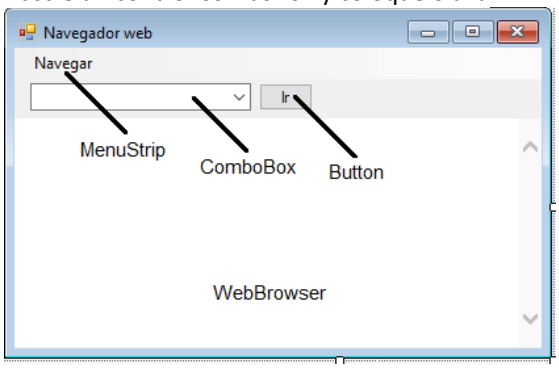


Agregue un botón.

- En el *Cuadro de herramientas*, en la categoría *Controles comunes*, arrastre un control ***Button*** hasta aproximadamente la mitad del formulario, justo debajo de la barra de menús. En sus *Propiedades*, cambie la propiedad *Text* a *Ir* en lugar de *button1*, y cambie el nombre del diseño, que se muestra como (Nombre), de button1 a ***BotonIr***.

Agregue un control ComboBox.

- En el *Cuadro de herramientas*, en la categoría *Controles comunes,* arrastre un control *ComboBox* y colóquelo a la *izquierda* del botón. Arrastre los bordes y las esquinas para cambiar el tamaño hasta que quede alineado con el botón. El control *ComboBox* va a contener una lista de los sitios web favoritos. Para crear la lista de sitios, seleccione el control ComboBox y vea sus *propiedades*.
  Seleccione la propiedad ***Items*** Agregue tantas direcciones URL del sitio Web como desee, como por ejemplo:
  http://www.ofimega.es
  http://www.google.com
  presionando RETORNO después de cada una.



Agregue el control WebBrowser.

- En el *Cuadro de herramientas*, en la categoría *Controles comunes*, desplácese hacia abajo hasta llegar al control *WebBrowser*. Arrastre el control hasta el formulario Windows Forms. Cambie el tamaño del control WebBrowser para ajustarlo al formulario Windows sin ocultar los controles ComboBox y Button. Al establecer la configuración de Anchor en Superior, Inferior, Izquierda, Derecha, el control WebBrowser cambiará su tamaño correctamente cuando se cambie el tamaño de la ventana de la aplicación. El control WebBrowser realiza la representación de páginas Web.

Agregue un controlador de eventos para el control Button.

- Un controlador de eventos es un método que se ejecuta cuando el usuario interactúa con el control.
- Haga doble clic en el botón y verá aparecer el *Editor de código* para el proyecto. También verá que se ha creado el controlador para el evento Click, Agregue código al método del controlador de eventos de modo similar al siguiente código:

```
private void goButton_Click(object sender, System.EventArgs e)
{
    webBrowser1.Navigate(new Uri(comboBox1.SelectedItem.ToString()));
}
```

Agregue controladores de eventos para las opciones de MenuStrip.

- Vuelva a la ventana Diseñador y haga doble clic en los subelementos del menú de uno en uno. Visual C# Express creará métodos de control de eventos para cada uno. Edite estos métodos, de modo que se asemejen al código siguiente.

```
private void homeToolStripMenuItem_Click(object sender, System.EventArgs e)
{
    webBrowser1.GoHome();
}
private void goForwardToolStripMenuItem_Click(object sender, System.EventArgs e)
{
    webBrowser1.GoForward();
}
private void goBackToolStripMenuItem_Click(object sender, System.EventArgs e)
{
    webBrowser1.GoBack();
}
```

Agregue código de inicialización a Form1 en el método Form1_Load.

- Haga clic en la ficha Form1.cs [Diseño] en la parte superior del editor de código para regresar al formulario Windows. Seleccione el formulario y, en la ventana *Propiedades*, haga clic en el botón *Eventos* (el que tiene un icono de rayo) y, a continuación, haga doble clic en Cargar. Esto agregará un método de control de eventos y colocará el cursor en el método en la vista Código.
- En la vista Código, agregue :

```
private void Form1_Load(object sender, EventArgs e)
{
    comboBox1.SelectedIndex = 0;
    webBrowser1.GoHome();
}
```

Genere y ejecute el programa.

- Presione F5 para generar y ejecutar el explorador web. Se mostrará en pantalla el formulario Windows Forms creado y, a continuación, aparecerá la página principal predeterminada del equipo. Puede utilizar el control ComboBox para seleccionar un sitio web, y hacer clic en Ir para navegar al mismo.

Para finalizar pulse en *Archivo – Guardar todo.*

Un poco más de teoría:

Tipos de datos y operadores.

Tipos de Datos en C#

Tipo	Descripción
int / long	Números enteros.
float / double	Números de coma flotante
decimal	Valores de moneda.
char	Un simple carácter Unicode.
bool	Valor booleano.
DateTime	Momentos en el tiempo
string	Texto o cadena de caracteres

Operadores

Aritméticos	+, -, *, /, %
Incremento, decremento	++, --
Concatenación	+
Operaciones lógicas	&, |, ^, !, ~, &&, ||
Indizado	[ ]
Asignación	=, +=, -=, *=, /=, %=, ^=, <<=
Tipo de datos	sizeof, typeof
Apuntadores	*, ->, [ ], &

TIPOS DE ARCHIVOS EN C# CON WINDOWS FORMS:

Un archivo de solución (*.sln) puede contener a uno o varios proyectos. Un proyecto puede contener varias clases o tipos. Un proyecto de Windows Forms contiene la clase o tipo Form1. Al guardar una solución, se guardan, a su vez, los siguientes archivos, por ejemplo:

Form1.**cs**: Código fuente en C#; Form1.**resx**: archivo de recursos en XML;
Ejemplo.**csproj**: Archivo del proyecto; Ejemplo.**projdata**; archivo oculto interno de VStudio
ClassExample.**sln**: Archivo que contiene los elementos de la solución.
ClassExample.**csproj.user**: guarda las opciones personalizadas del proyecto.

EJERCICIO 1

- Realizar una interfaz gráfica que permita al usuario introducir dos números. El programa calculará y desplegará la suma de ambos.
- Se requieren 3 etiquetas, 3 cuadros de texto y dos botones. Cambiar sus propiedades como indica la 2ª figura

Notar las "Líneas guía" que el editor muestra al acomodar los controles para alinearlos facilmente.

- **Añadir el código:**

En el botón sumar button1_Click():

```
int n1, n2, suma;
n1 = int.Parse(txtNumero1.Text);
n2 = int.Parse(txtNumero2.Text);
suma = n1 + n2;
txtResultado.Text = suma.ToString();
```

En el botón salir: `Application.Exit();`

Ejercicio propuesto: Agregar un botón más al formulario para "Limpiar" el contenido de los cuadros de texto

EJERCICIO 2:

Realizar una pantalla que pida "Login" y "Password" a un usuario. Mostrar un mensaje de "Bienvenida" si los datos son correctos, o un mensaje de "Rechazo" si no lo son.
Datos correctos: – Login: "ofimega" – Password: "danone"

Formulario:

Código:

```
private void button1_Click(object sender, EventArgs e)
{
  string Login, Password; //variables de texto
  Login = txtLogin.Text.Trim();  //quita espacios
  Password = txtPassword.Text.TrimEnd(); //quita espacios
  if (Login=="Ofimega" && Password=="danone")
  {MessageBox.Show("Bienvenido al sistema"); }
   else
  { MessageBox.Show("Acceso denegado"); }
}
```

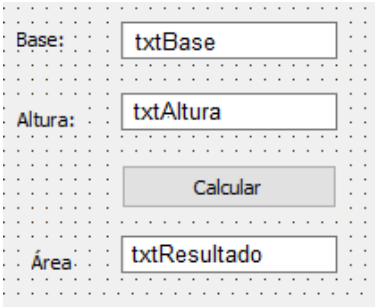
- Ocultar los caracteres tecleados en el *password* en la propiedad *PasswordChar*
- El método ***TrimEnd***( ) elimina los espacios en blanco hasta el final
- ***MessageBox***.Show ("Mensaje") muestra una ventana con un mensaje para el usuario

Comparación de cadenas:

- mediante el comparador ==
- mediante ***Equals***
- mediante ***CompareTo***

EJERCICIO. AREA DEL TRIANGULO:

- El programa calculará el área del triángulo a partir de la base y la altura con la fórmula: Area = Base*Altura/2
- Se requieren 3 etiquetas, 3 cuadros de texto y un botón.
- Distribuir y cambiar sus propiedades como indica la figura



Código en C ++ (Builder) :

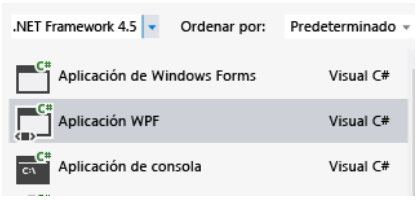
```
void __fastcall TForm5::Button1Click(TObject *Sender) {

   txtResultado->Text=FormatFloat("###.##",txtBase ->
   Text.ToDouble()* txtAltura ->Text.ToDouble()/2);
}
```

Código en C # :

```
private void button1_Click(object sender, EventArgs e) {
    float b, a;
    b=float.Parse(txtBase.Text);
    a=float.Parse(txtAltura.Text);
    txtResultado.Text = (a * b / 2).ToString();
}
```

Crear una aplicación en modo WPF:



Una aplicación en modo **WPF** (Windows Presentation Foundation) permite utilizar el diseño del formulario en formato hipertexto extendido XAML, sus controles están basados en formato vectorial, código de programación separado del diseño gráfico, permite la posibilidad de trabajo en conjunto para diseñadores y programadores

- Elije del menú Crear un Nuevo proyecto del tipo: *Aplicación **WPF*** →
- Dale el nombre: ***HolaMundoWPF*** y la ubicación que quieras para tu aplicación.
- Aconsejable tener marcada la casilla de verificación: ☑ *Crear directorio para la solución*.

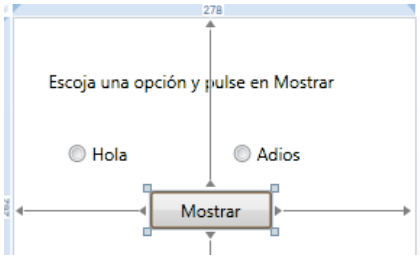
Cambiar el nombre de MainWindow.xaml:

- En el Explorador de soluciones, selecciona MainWindow.xaml.
- En la ventana Propiedades, (Si no se ve: Ver -> Ventana de propiedades). Cambia la propiedad Nombre de archivo (*File Name*) : a Hola.xaml. Este archivo de código está anidado bajo el nodo del archivo .xaml para mostrar su relación.

Agregar controles:

- En el Cuadro de herramientas (Si no se ve: Ver -> Ventana de propiedades), busca el control: TextBlock. (Bloque de texto) y arrástralo a la ventana.
- Para cambiar el texto del recuadro en las propiedades: Con el textBlock seleccionado, busca la propiedad Text y añade el texto: “Escoja una opción y pulse en Mostrar”
- Para cambiar el texto del recuadro en modo Xaml, busca la línea Xaml:

```
<TextBlock Margin="30,58,21,0" Name="textBlock1" Text="Escoja
una opción y pulse en Mostrar" Height="42"
VerticalAlignment="Top" />
```



- Eligiendo el elemento RadioButton y arrástralo a la dos veces para tener dos controles RadioButton.
- En la superficie de diseño, selecciona RadioButton1 y en sus propiedades añade a la propiedad Content el texto: Hola.
- En la superficie de diseño, selecciona RadioButton2 y en sus propiedades añade a la propiedad Content el texto: Adiós.
- En el Cuadro de herramientas, busca el control *Botón* (Button) y, después, agrégalo a la superficie de diseño.
- Cambia la propiedad del botón Content por: Mostrar.

Agregar código al botón Mostrar

En la versión 2018 pulsa doble clic sobre el botón para abrir el evento: button1_Click
o en la versión 2013 pulsa en el rayo junto a las propiedades.

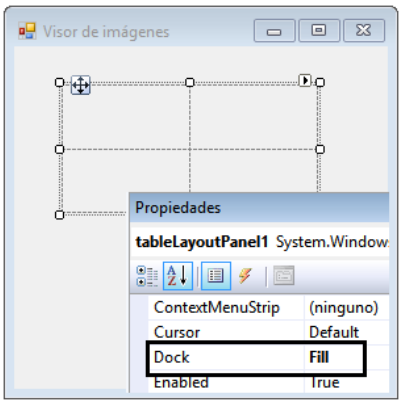
```
private void button1_Click(object sender,
RoutedEventArgs e)
    {
        if (radioButton1.IsChecked==true)
        {
            MessageBox.Show("Hola");
        }
        else
        {
            radioButton2.IsChecked = true;
            MessageBox.Show("Adiós");
        }
    }
```

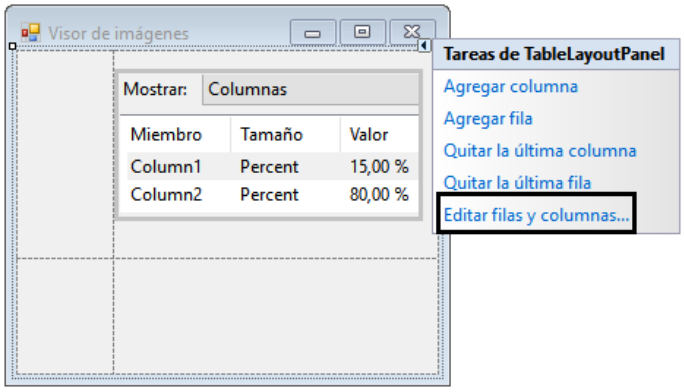
Depurar y probar:

Para buscar y corregir errores, inicia el depurador seleccionando *Depurar -> Iniciar depuración*.
Aparece un mensaje de error: No se encuentra el recurso ‘mainwindow.xaml’.
Falta especificar Hola.xaml como el URI de inicio: abre el archivo App.xaml y cambia StartupUri="MainWindow.xaml" a StartupUri="Hola.xaml" y después guarda los cambios con Ctrl-s.
Para más información véase el sitio de Microsoft: https://msdn.microsoft.com/es-es/library/jj153219.aspx

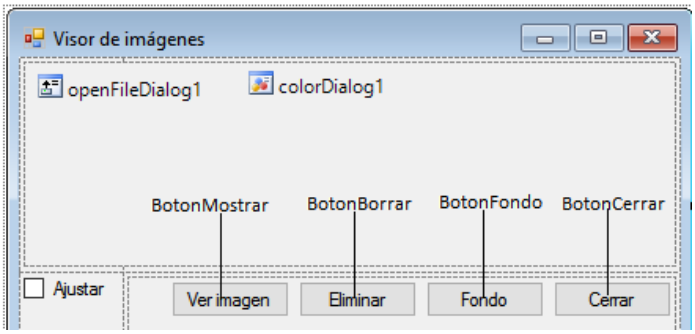
Ejercicio C# Windows Forms: Visor de imágenes.

- Crea un *Nuevo Proyecto* – Aplicación de Windows Forms.
- Pon el nombre del nuevo formulario: ***Visor***, y pulsa Aceptar.
- Arrastra la esquina de la ventana-formulario para ampliar su tamaño.
- En el panel *Propiedades*, escribe en ***Texto: Visor de imágenes***. Si no ves la ventana de propiedades pulsa en .
- Arrastra de la ***Caja de herramientas*** un control: ***TableLayoutPanel*** sobre el formulario.
- Selecciónalo y cambia la *propiedad*: ***Doc*** a ***Fill*** para ajustarlo a todo el formulario.
- TableLayoutPanel tiene dos filas y dos columnas de igual tamaño. Para cambiar el tamaño de la fila superior y la columna derecha, pulsa en un pequeño triángulo negro de la esquina superior derecha. Selecciona *Editar filas y columnas*. Pon **15** en el porcentaje de la columna 1. Seleccione Filas (rows) y pon a **90** % la fila 1 y a 10 % la fila 2.





- Añade, del cuadro de herramientas, un ***PictureBox*** (cuadro de imagen) al formulario.
- Cambia su propiedad ***Dock*** por Fill (llenar).
- Establecer su ***ColumnSpan*** propiedad a ***2*** . Además, cuando el PictureBox está vacío, quiere mostrar un marco vacío. Establece su propiedad BorderStyle a Fixed3D.
- Añade un control ***CheckBox*** al formulario.
- Agregar un control ***FlowLayoutPanel*** a la última celda (abajo a la derecha) y cambia su propiedad ***Dock*** por ***Fill*** (llenar).
- De la caja de herramientas, añade *tres* botones ***Button*** al FlowLayoutPanel.
- En el primer botón y establece su propiedad texto a: *Ver imagen*. A continuación, establece las propiedades de texto de los tres botones: *Eliminar, Color de fondo* y *Cerrar*.
- Cambiar en el FlowLayoutPanel su propiedad ***FlowDirection*** en ***RightToLeft*** . Los botones deben alinearse a la derecha de la celda, e invertir su orden.
- Selecciona todos los botones a la vez, (tecla CTRL) y cambia la propiedad ***AutoSize*** = True.



- Cambiar el nombre de los botones: BotonBorrar, BotonCerrar, BotonFondo y BotonMostrar.
- Añade un control ***ColorDialog*** al formulario desde el cuadro de herramientas (Tool box)
- Añade también un control ***OpenFileDialog***. Cambia su propiedad ***Filter***: JPEG (*.jpg)|*.jpg|PNG (*.png)|*.png|BMP (*.bmp)|*.bmp|Todos (*.*)|*.* y su propiedad ***Title*** : "Escoja una imagen".

Código en C#:

```
private void BotonMostrar_Click(object sender,
EventArgs e)
{
  if (openFileDialog1.ShowDialog() ==
DialogResult.OK)
  {
pictureBox1.Load(openFileDialog1.FileName);
} }

private void BotonBorrar_Click(object sender,
EventArgs e)
{
pictureBox1.Image = null;   // Borra la imagen.
}

private void BotonFondo_Click(object sender,
EventArgs e)
{
    if (colorDialog1.ShowDialog() ==
DialogResult.OK)  // abre la caja de color
pictureBox1.BackColor = colorDialog1.Color;
}
```

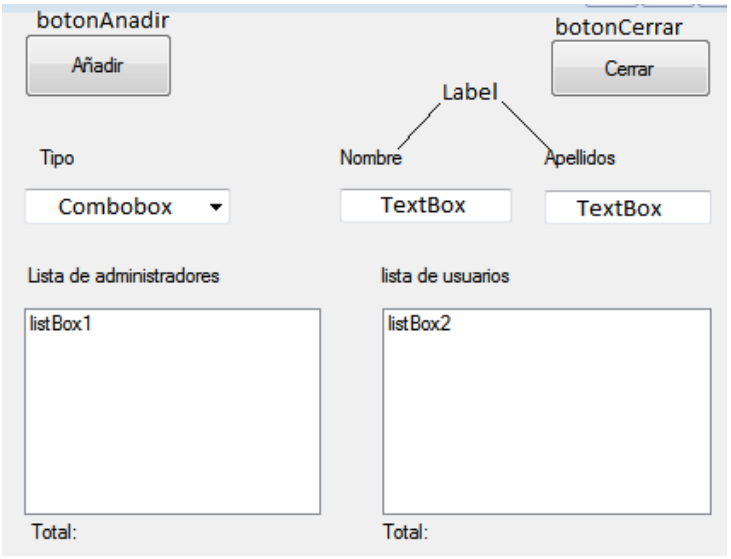
```
private void BotonCerrar_Click(object sender,
EventArgs e)
{
   this.Close();    // cierra la ventana
}

private void checkBox1_CheckedChanged(object
sender, EventArgs e)
{
  if (checkBox1.Checked)   // cambia el ajuste
  pictureBox1.SizeMode =
PictureBoxSizeMode.StretchImage;
  else
  pictureBox1.SizeMode =
PictureBoxSizeMode.Normal;
}
```

Ejercicio C# Windows Forms: Listas

En este ejercicio utilizaremos variables de cadena de texto (strings) para traspasar texto de unos objetos a otros.

- Crea un nuevo proyecto:
  Archivo – Nuevo proyecto (File – New Project) - Aplicación de Windows Forms: **Listas**
- Añade al formulario los objetos de la figura. →



Ingredientes:

- 7 Labels
- 2 Buttons
- 2 TextBox
- 1 ComboBox
  Items: Administrador / Usuario
  Text: Usuario
- 2 ListBox

- Añade el código a los eventos ⚡:
- Añadir el código al evento Click del botonCerrar:

```
private void botonCerrar_Click(object sender, EventArgs e) {
   this.Close();}   //-> añadir este código
```

(en realidad este código es delegado al método:
this. botonCerrar.Click += new system.EventHandler(this. botonCerrar Click);)

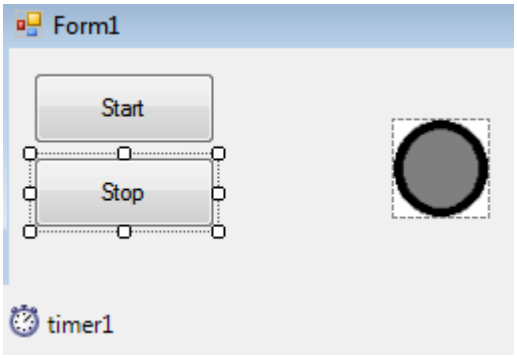
- Añadir el código al evento Click del botonAnadir:

```
private void botonAnadir_Click(object sender, EventArgs e) {
    string tipo = comboBox1.Text;
    string nombre = textBox1.Text;
    string apellidos = textBox2.Text;
    string nombreCompleto = nombre + " " +apellidos;
    int total1, total2;
    if (nombreCompleto != " ")   {  // probar con: if (comboBox1.SelectedIndex==0)
        if (tipo == "Administrador")
            listBox1.Items.Add(nombreCompleto);
        else listBox2.Items.Add(nombreCompleto);
    total1 = comboBox1.Items.Count;
    labelTotal1.Text = "Total: " + total1.ToString();
    }
    else  MessageBox.Show("mal");
       }    }
```

Ejercicio C# Windows Forms: Pong

En este ejercicio utilizaremos las propiedades *Left* y *Top* y el objeto *Timer* para desplazar una imagen por la ventana.

- Crea un nuevo proyecto:
  Archivo – Nuevo proyecto (File – New Project)
  Aplicación de Windows Forms: **Pong**
- Añade al formulario los objetos de la figura. →



Ingredientes:

- 2 Buttons: Text: Start y Stop
- 1 Timer: Interval: 20.
  Image: Importar pelota.gif
- 1 Picture box. Name: bola

- Añade el código:

Primero crearemos dos variables públicas antes de public Form():

```
int vel=5;       //variable numérica entera para la velocidad
int direc = 1;   // variable numérica entera para la dirección
   public Form1()
```

- Añadir el código al evento Click del botón Start: `timer1.Enabled = true;`
- Añadir el código al evento Click del botón Stop: `timer1.Enabled = false;`
- Añadir el código al evento Tic del Timer:

```
private void timer1_Tick(object sender, EventArgs e)
{
if (direc == 1) //abajo der
{
 bola.Left = bola.Left + vel;
 bola.Top = bola.Top + vel;
 if ((bola.Top+bola.Height) >= this.Height) direc
 = 2;
 if ((bola.Left+ bola.Width) >= this.Width) direc
 = 3;
}
if (direc == 2) //arriba der
{
 bola.Left = bola.Left + vel;
 bola.Top = bola.Top - vel;
 if ((bola.Top) <=0) direc = 1;
 if ((bola.Left + bola.Width) >= this.Width)
 direc = 4;
}
if (direc == 3) //abajo iz
{
 bola.Left = bola.Left - vel;
 bola.Top = bola.Top + vel;
 if ((bola.Top) >= this.Height) direc = 4;
 if ((bola.Left)<=0 ) direc = 1;
}
if (direc == 4) //arriba iz
{
 bola.Left = bola.Left - vel;
 bola.Top = bola.Top - vel;
 if ((bola.Top) <=0) direc = 3;
 if (bola.Left <= 0) direc = 2;
}
}
```

Ejercicio propuesto:

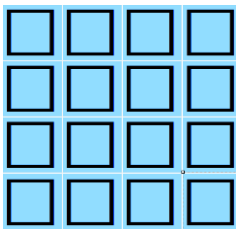
Al hacer clic sobre la pelota, aumentará la velocidad y la puntuación.

a) `Incrementando la variable vel: vel++;`
b) `Reduciendo el intervalo del timer: timer1.interval--;`

Ejercicio C# Windows Forms: Juego de parejas. (Extracto tutorial MSDN)

En el Juego de buscar parejas entre iconos ocultos.

- Crear el proyecto: Archivo – Nuevo proyecto (File – New Project) - Aplicación de Windows Forms: **Parejas**
- Cambia la propiedad Tamaño del Form (Size): en 550; 550
- Agrega un control TableLayoutPanel
  Propiedades: BackColor: tipo web: CornflowerBlue - Dock: Fill - CellBorderStyle : Inset (Insertado)
  Pulsa en el triángulo del menú contextual: Editar Filas y Columnas: 4 filas x 4 columnas. 25%



- Con el control TableLayoutPanel seleccionado, agrega un control Label a la celda superior izquierda del TableLayoutPanel.
  Propiedades para el label: BackColor: tipo web: CornflowerBlue - AutoSize: False – Dock: Fill – TextAlign: MiddleCenter – Font: Webdings, Estilo de fuente en Negrita y Tamaño en 70 – Text: c.
- Copia el Label y pega en cada cuadro.
- En: Ver- Código, agregar el código:

```
public partial class Form1 : Form
{
    Random random = new Random(); // aleatorio
    List<string> icons = new List<string>() //nuevo objeto del tipo lista de strings para los iconos
    {
        "!", "!", "N", "N", ",", ",", "k", "k",
        "b", "b", "v", "v", "w", "w", "z", "z"
    };
    Label firstClicked = null;
    Label secondClicked = null;
    private void AssignIconsToSquares() //asignaremos a cada label un valor aleatorio
    {
        foreach (Control control in tableLayoutPanel1.Controls) //repite para cada control
        {
            Label iconLabel = control as Label;
            if (iconLabel != null)
            {
                int randomNumber = random.Next(icons.Count);
                iconLabel.Text = icons[randomNumber];
                icons.RemoveAt(randomNumber);
            }
        }
    }
    public Form1()
    {
        InitializeComponent();
        AssignIconsToSquares(); //--> Llamamos al método
    }
}
```

Una vez comprobado, añade el código: `iconLabel.ForeColor = iconLabel.BackColor; para ocultar iconos`

- Añade el código al evento ⚡Click del Label:

```
Label clickedLabel = sender as Label;
if (clickedLabel != null)  {
    if (clickedLabel.ForeColor == Color.Black)
        return;
    if (firstClicked == null) {
        firstClicked = clickedLabel;
        firstClicked.ForeColor = Color.Black;
        return;
    }
}
```

- Agrega un control Timer. Interval: 750

```
private void timer1_Tick(object sender, EventArgs e)
    {
        timer1.Stop();
        firstClicked.ForeColor = firstClicked.BackColor;
        secondClicked.ForeColor = secondClicked.BackColor;
        firstClicked = null;
        secondClicked = null;
    }
```

Código completo:

```
namespace WindowsFormsApplication{
    public partial class Form1 : Form{
     Random random = new Random(); // aleatorio
     List<string> icons = new List<string>() //iconos
     {
      "!", "!", "N", "N", ",", ",", "k", "k",
      "b", "b", "v", "v", "w", "w", "z", "z"
     };
    Label firstClicked = null;
    Label secondClicked = null;
    private void AssignIconsToSquares() {
      foreach (Control control in tableLayoutPanel1.Controls) //repite para cada control
      {
      Label iconLabel = control as Label;
    if (iconLabel != null) {
          int randomNumber = random.Next(icons.Count);
          iconLabel.Text = icons[randomNumber];
          icons.RemoveAt(randomNumber);
          iconLabel.ForeColor = iconLabel.BackColor;
     }
            }  }
       public Form1()
       {
         InitializeComponent();
         AssignIconsToSquares();
       }

    private void label1_Click(object sender, EventArgs e)
    {
      if (timer1.Enabled == true)
          return;
      Label clickedLabel = sender as Label;
      if (clickedLabel != null) {
          if (clickedLabel.ForeColor == Color.Black)
              return;
          CheckForWinner();
          if (firstClicked == null) {
              firstClicked = clickedLabel;
              firstClicked.ForeColor = Color.Black;
              return;
           }
            secondClicked = clickedLabel;
            secondClicked.ForeColor = Color.Black;
            secondClicked = clickedLabel;
            secondClicked.ForeColor = Color.Black;
            if (firstClicked.Text == secondClicked.Text) {
               firstClicked = null;
               secondClicked = null;
               return;
            }
               timer1.Start();
           } }
    private void timer1_Tick(object sender, EventArgs e) {
          timer1.Stop();// Stop the timer
          firstClicked.ForeColor = firstClicked.BackColor;// Hide both icons
          secondClicked.ForeColor = secondClicked.BackColor;
          firstClicked = null;
          secondClicked = null;
        }
    private void CheckForWinner(){
    foreach (Control control in tableLayoutPanel1.Controls)
    {
       Label iconLabel = control as Label;
       if (iconLabel != null) {
       if (iconLabel.ForeColor == iconLabel.BackColor)
       return;
    }}
       MessageBox.Show("Terminado. Felicidades!");
       Close();     }  }   }
```