**FORMULARIO PRINCIPAL🡪 frm\_principal**

**using** System;  
**using** System.Drawing;  
**using** System.Windows.Forms;  
**using** System.IO;  
  
**namespace** matriz  
{  
    **public** partial class frm\_principal : Form  
    {  
        funciones fun = **new** **funciones**();  
          
        **public** **frm\_principal**()  
        {  
            **InitializeComponent**();  
        }  
                  
        void **Btn\_sumayrestaClick**(object sender, EventArgs e)  
        {  
            **abrir\_en\_panel**(**new** **frm\_sumayresta**());  
        }  
          
        void **Btn\_productoescalarClick**(object sender, EventArgs e)  
        {  
            **abrir\_en\_panel**(**new** **frm\_producto\_escalar**());  
        }  
          
        void **Btn\_productoClick**(object sender, EventArgs e)  
        {  
            **abrir\_en\_panel**(**new** **frm\_multiplicacion**());  
        }  
          
        void **Btn\_tiposClick**(object sender, EventArgs e)  
        {  
            **abrir\_en\_panel**(**new** **frm\_tipomatriz**());  
        }  
          
        void **Btn\_crearClick**(object sender, EventArgs e)  
        {  
            **abrir\_en\_panel**(**new** **frm\_muestra**());  
        }  
          
        **public** void **abrir\_en\_panel**(object formulario)  
        {  
               pnl\_body.Controls.**Clear**();  
            Form form = formulario **as** Form;  
            form.TopLevel = **false**;  
            pnl\_body.Controls.**Add**(form);  
            form.**Show**();  
        }  
          
        void **Pcbx\_linkClick**(object sender, EventArgs e)  
        {  
            System.Diagnostics.Process.**Start**("http://www.genesisvargasj.tk");  
        }  
          
        void **Pcbx\_manualClick**(object sender, EventArgs e)  
        {  
            string pdfPath = Path.**Combine**(Application.StartupPath, "manual.pdf");  
            System.Diagnostics.Process.**Start**(pdfPath);  
        }  
    }  
}

**FORMULARIO SUMA Y RESTA🡪 frm\_sumayresta**

**using** System;  
**using** System.Drawing;  
**using** System.Windows.Forms;  
  
**namespace** matriz  
{  
    **public** partial class frm\_sumayresta : Form  
    {  
        **int** m, n;  
        funciones fun = **new** **funciones**();  
  
        **public** **frm\_sumayresta**()  
        {  
            **InitializeComponent**();  
        }  
          
        void **Cmbx\_filasSelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_columnas.Text == "")  
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz1);  
                fun.**crearmatriz**(m, n, dgv\_matriz2);      
            }  
            **else**  
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                n = Convert.**ToInt32**(cmbx\_columnas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz1);      
                fun.**crearmatriz**(m, n, dgv\_matriz2);      
            }  
        }  
          
        void **Cmbx\_columnasSelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_columnas.Text == cmbx\_filas.Text && cmbx\_filas.Text != "")   
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                n = Convert.**ToInt32**(cmbx\_columnas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz1);  
                fun.**crearmatriz**(m, n, dgv\_matriz2);  
                fun.**error**(error, dgv\_matriz1,"");  
            }  
            **else**  
            {  
                fun.**error**(error, dgv\_matriz1,"EL NÚMERO DE FILAS DEBEN SER IGUALES QUE EL NÚMERO DE LAS COLUMNAS");  
            }  
        }  
          
        void **btn\_calcular\_Click**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_columnas.Text == cmbx\_filas.Text && cmbx\_filas.Text != "")   
            {  
                fun.**error**(error, dgv\_matriz1, "");  
                fun.**crearmatriz**(m, n, dgv\_suma);  
                fun.**crearmatriz**(m, n, dgv\_resta);  
                **if** (dgv\_matriz1.Rows.Count != 0 && dgv\_matriz2.Rows.Count != 0 && !fun.**celdas\_vacias**(dgv\_matriz1) && !fun.**celdas\_vacias**(dgv\_matriz2))  
                {  
                    fun.**error**(error, dgv\_suma,"");  
  fun.**sumayresta**(m, n, dgv\_matriz1, dgv\_matriz2, dgv\_suma, dgv\_resta);    
                }  
                **else**  
                {  
                    fun.**error**(error, dgv\_suma, "NO SE HAN CONSTRUIDO LAS MATRICES, HAY CELDAS VACIAS, O ESTAS DIGITANDO MAL LOS NUMEROS NEGATIVOS");  
                }  
            }  
            **else**   
            {  
                fun.**error**(error, dgv\_matriz1, "EL NÚMERO DE FILAS DEBEN SER IGUALES QUE EL NÚMERO DE LAS COLUMNAS");  
            }                  
        }  
     
        void **Dgv\_matriz1KeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**solo\_numeros**(dgv\_matriz1, e);  
        }  
          
        void **Dgv\_matriz2KeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**solo\_numeros**(dgv\_matriz2, e);  
        }  
          
        void **Cmbx\_filasKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Cmbx\_columnasKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
  
        void **Dgv\_matriz1EditingControlShowing**(object sender, DataGridViewEditingControlShowingEventArgs e)  
        {  
            TextBox celda = e.Control **as** TextBox;  
            celda.KeyPress -= **new** **KeyPressEventHandler**(Dgv\_matriz1KeyPress);  
            celda.KeyPress += **new** **KeyPressEventHandler**(Dgv\_matriz1KeyPress);              
        }  
           
        void **Dgv\_matriz2EditingControlShowing**(object sender, DataGridViewEditingControlShowingEventArgs e)  
        {  
            TextBox celda = e.Control **as** TextBox;  
            celda.KeyPress -= **new** **KeyPressEventHandler**(Dgv\_matriz2KeyPress);  
            celda.KeyPress += **new** **KeyPressEventHandler**(Dgv\_matriz2KeyPress);   
        }  
    }  
}

**FORMULARIO PRODUCTO POR UN ESCALAR🡪 frm\_producto\_escalar**

**using** System;  
**using** System.Drawing;  
**using** System.Windows.Forms;  
  
**namespace** matriz  
{  
    **public** partial class frm\_producto\_escalar : Form  
    {  
        **int** esc, m, n;  
        funciones fun = **new** **funciones**();  
  
        **public** **frm\_producto\_escalar**()  
        {  
            **InitializeComponent**();  
        }  
          
        void **Cmbx\_filasSelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_columnas.Text == "")   
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz);  
            }  
            **else**  
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                n = Convert.**ToInt32**(cmbx\_columnas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz);      
                fun.**crearmatriz**(m, n, dgv\_resultado);  
            }  
        }  
          
        void **Cmbx\_columnasSelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_filas.Text != "")   
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                n = Convert.**ToInt32**(cmbx\_columnas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz);  
                fun.**crearmatriz**(m, n, dgv\_resultado);  
                fun.**error**(error, dgv\_matriz,"");  
            }  
            **else**  
            {  
                fun.**error**(error, dgv\_matriz,"NO HA ESCOGIDO EL NUMERO DE FILAS");  
            }  
        }  
          
        void **Btn\_calcularClick**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_columnas.Text != "")   
            {  
                fun.**error**(error, dgv\_matriz, "");  
                **if** (txt\_escalar.Text != "" && dgv\_matriz.Rows.Count != 0 && !fun.**celdas\_vacias**(dgv\_matriz))  
                {  
                    fun.**error**(error, dgv\_matriz,"");  
                    esc = Convert.**ToInt32**(txt\_escalar.Text);  
                    fun.**producto\_escalar**(m, n, dgv\_matriz, dgv\_resultado, esc);   
                }  
                **else**  
                {  
                    fun.**error**(error, dgv\_matriz,"NO HA DIGITADO EL NUMERO ESCALAR, NO SE HAN CONSTRUIDO LAS MATRICES, HAY CELDAS VACIAS, O ESTAS DIGITANDO MAL LOS NUMEROS NEGATIVOS");  
                }  
            }      
            **else**  
            {  
                fun.**error**(error, dgv\_matriz,"AÚN NO HAS ESCOGIDO LAS COLUMNAS");  
            }  
        }  
          
        void **Txt\_escalarKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**solo\_numeros**(dgv\_matriz, e);  
        }  
          
        void **Dgv\_matrizKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**solo\_numeros**(dgv\_matriz, e);  
        }  
          
        void **Cmbx\_filasKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Cmbx\_columnasKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Dgv\_matrizEditingControlShowing**(object sender, DataGridViewEditingControlShowingEventArgs e)  
        {  
            TextBox celda = e.Control **as** TextBox;  
            celda.KeyPress -= **new** **KeyPressEventHandler**(Dgv\_matrizKeyPress);  
            celda.KeyPress += **new** **KeyPressEventHandler**(Dgv\_matrizKeyPress);  
        }    
    }  
}

**FORMULARIO PRODUCTO🡪 frm\_multiplicacion**

**using** System;  
**using** System.Drawing;  
**using** System.Windows.Forms;  
  
**namespace** matriz  
{  
    **public** partial class frm\_multiplicacion : Form  
    {  
        **int** m, n, m2, n2;  
        funciones fun = **new** **funciones**();  
              
        **public** **frm\_multiplicacion**()  
        {  
            **InitializeComponent**();  
        }  
          
        void **Cmbx\_filasSelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_columnas.Text == "")  
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz1);  
            }  
            **else**  
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                n = Convert.**ToInt32**(cmbx\_columnas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz1);      
            }  
        }  
          
        void **Cmbx\_columnasSelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_filas.Text != "")   
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                n = Convert.**ToInt32**(cmbx\_columnas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz1);  
                fun.**error**(error, dgv\_matriz1,"");  
            }  
            **else**  
            {  
                fun.**error**(error, dgv\_matriz1,"NO HA ESCOGIDO EL NUMERO DE FILAS DE LA MATRIZ");  
            }  
        }  
          
        void **Cmbx\_filas2SelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_columnas2.Text == "")  
            {  
                m2 = Convert.**ToInt32**(cmbx\_filas2.Text);  
                fun.**crearmatriz**(m2, n2, dgv\_matriz2);      
            }  
            **else**  
            {  
                m2 = Convert.**ToInt32**(cmbx\_filas2.Text);  
                n2 = Convert.**ToInt32**(cmbx\_columnas2.Text);  
                fun.**crearmatriz**(m2, n2, dgv\_matriz2);      
            }  
        }  
          
        void **Cmbx\_columnas2SelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_filas2.Text != "")   
            {  
                m2 = Convert.**ToInt32**(cmbx\_filas2.Text);  
                n2 = Convert.**ToInt32**(cmbx\_columnas2.Text);  
                fun.**crearmatriz**(m2, n2, dgv\_matriz2);  
                fun.**error**(error, dgv\_matriz2,"");  
            }  
            **else**  
            {  
                fun.**error**(error, dgv\_matriz2,"NO HA ESCOGIDO EL NUMERO DE FILAS DE LA MATRIZ");  
            }  
        }  
          
        void **Btn\_calcularClick**(object sender, EventArgs e)  
        {  
            **if** (dgv\_matriz1.Rows.Count != 0 && dgv\_matriz2.Rows.Count != 0 && cmbx\_columnas.Text == cmbx\_filas2.Text && !fun.**celdas\_vacias**(dgv\_matriz1) && !fun.**celdas\_vacias**(dgv\_matriz2))  
            {  
                fun.**error**(error, dgv\_resultado,"");  
                fun.**crearmatriz**(m, n2, dgv\_resultado);  
                fun.**multiplicacion**(m, n, n2, dgv\_matriz1, dgv\_matriz2, dgv\_resultado);  
            }  
            **else**  
            {  
                fun.**error**(error, dgv\_resultado,"LAS COLUMNAS DE LA PRIMERA MATRIZ DEBEN SER IGUALES A LAS FILAS DE LA SEGUNDA MATRIZ, NO SE HAN CONSTRUIDO LAS MATRICES, HAY CELDAS VACIAS, O ESTAS DIGITANDO MAL LOS NUMEROS NEGATIVOS");  
            }  
        }  
          
        void **Dgv\_matriz1KeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**solo\_numeros**(dgv\_matriz1, e);  
        }  
          
        void **Dgv\_matriz2KeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**solo\_numeros**(dgv\_matriz2, e);  
        }  
          
        void **Cmbx\_filasKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Cmbx\_columnasKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Cmbx\_filas2KeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Cmbx\_columnas2KeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Dgv\_matriz1EditingControlShowing**(object sender, DataGridViewEditingControlShowingEventArgs e)  
        {  
            TextBox celda = e.Control **as** TextBox;  
            celda.KeyPress -= **new** **KeyPressEventHandler**(Dgv\_matriz1KeyPress);  
            celda.KeyPress += **new** **KeyPressEventHandler**(Dgv\_matriz1KeyPress);   
        }  
          
        void **Dgv\_matriz2EditingControlShowing**(object sender, DataGridViewEditingControlShowingEventArgs e)  
        {  
            TextBox celda = e.Control **as** TextBox;  
            celda.KeyPress -= **new** **KeyPressEventHandler**(Dgv\_matriz2KeyPress);  
            celda.KeyPress += **new** **KeyPressEventHandler**(Dgv\_matriz2KeyPress);   
        }  
    }  
}

**FORMULARIO TIPO DE MATRIZ🡪 frm\_tipomatriz**

**using** System;  
**using** System.Drawing;  
**using** System.Windows.Forms;  
  
**namespace** matriz  
{  
    **public** partial class frm\_tipomatriz : Form  
    {  
        **int** m, n;  
        funciones fun = **new** **funciones**();  
          
        **public** **frm\_tipomatriz**()  
        {  
            **InitializeComponent**();  
        }  
          
        void **Cmbx\_filasSelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_columnas.Text == "")   
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz);  
            }  
            **else**  
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                n = Convert.**ToInt32**(cmbx\_columnas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz);      
            }  
        }  
          
        void **Cmbx\_columnasSelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_filas.Text != "")   
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                n = Convert.**ToInt32**(cmbx\_columnas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz);  
                fun.**error**(error, dgv\_matriz,"");  
            }  
            **else**  
            {  
                fun.**error**(error, dgv\_matriz,"NO HA ESCOGIDO EL NUMERO DE FILAS DE LA MATRIZ");  
            }  
        }  
          
        void **Btn\_calcularClick**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_columnas.Text != "")   
            {  
                fun.**error**(error, dgv\_matriz, "");  
                fun.**error**(error, dgv\_matriz,"");  
                **if** (!fun.**hay\_trampa**(dgv\_matriz))  
                {  
                    lstbx\_resultados.Items.**Clear**();  
                    **if** (!fun.**es\_cuadrada**(m, n))  
                    {      
                        lstbx\_resultados.Items.**Add**("LA MATRIZ ES RECTANGULAR");  
                    }  
                    **else**  
                    {  
                        lstbx\_resultados.Items.**Add**("LA MATRIZ ES CUADRADA");  
                    }  
                    **if** (fun.**es\_nula**(m, n, dgv\_matriz))  
                    {  
                        lstbx\_resultados.Items.**Add**("LA MATRIZ ES NULA");  
                    }  
                    **if** (fun.**es\_identidad**(m, n, dgv\_matriz))  
                    {  
                        lstbx\_resultados.Items.**Add**("LA MATRIZ ES DE IDENTIDAD");  
                    }  
                    **if** (fun.**es\_diagonal**(m, n, dgv\_matriz))  
                    {  
                        lstbx\_resultados.Items.**Add**("LA MATRIZ ES DIAGONAL");  
                    }  
                    **if** (fun.**es\_simetrica**(m, n, dgv\_matriz))  
                    {  
                        lstbx\_resultados.Items.**Add**("LA MATRIZ ES SIMETRICA");  
                    }  
                    **if** (fun.**es\_antisimetrica**(m, n, dgv\_matriz))  
                    {  
                        lstbx\_resultados.Items.**Add**("LA MATRIZ ES ANTISIMETRICA");  
                    }  
                    **if** (fun.**es\_escalar**(m, n, dgv\_matriz))  
                    {  
                        lstbx\_resultados.Items.**Add**("LA MATRIZ ES ESCALAR");  
                    }  
                }      
                **else**  
                {  
                    fun.**error**(error, dgv\_matriz,"NO SE HAN CONSTRUIDO LAS MATRICES, HAY CELDAS VACIAS, O ESTAS DIGITANDO MAL LOS NUMEROS NEGATIVOS");  
                }  
            }  
            **else**   
            {  
                fun.**error**(error, dgv\_matriz, "AÚN NO HAS ESCOGIDO LAS COLUMNAS");  
            }                      
        }  
          
        void **Dgv\_matrizKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**solo\_numeros**(dgv\_matriz, e);  
        }  
          
        void **Cmbx\_filasKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Cmbx\_columnasKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Dgv\_matrizEditingControlShowing**(object sender, DataGridViewEditingControlShowingEventArgs e)  
        {  
            TextBox celda = e.Control **as** TextBox;  
            celda.KeyPress -= **new** **KeyPressEventHandler**(Dgv\_matrizKeyPress);  
            celda.KeyPress += **new** **KeyPressEventHandler**(Dgv\_matrizKeyPress);   
        }  
    }  
}

**FORMULARIO MUESTRA🡪 frm\_muestra**

**using** System;  
**using** System.Drawing;  
**using** System.Windows.Forms;  
  
**namespace** matriz  
{  
    **public** partial class frm\_muestra : Form  
    {  
        **int** m, n;  
        funciones fun = **new** **funciones**();  
          
        **public** **frm\_muestra**()  
        {  
            **InitializeComponent**();  
        }  
          
        void **Cmbx\_filasSelectedIndexChanged**(object sender, EventArgs e)  
        {    **if** (cmbx\_columnas.Text == "")   
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz);  
            }  
            **else**  
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                n = Convert.**ToInt32**(cmbx\_columnas.Text);  
                fun.**crearmatriz**(m, n, dgv\_matriz);  
                fun.**crearmatriz**(m, n, dgv\_opuesta);  
                fun.**crearmatriz**(n, m, dgv\_traspuesta);      
            }  
        }  
          
        void **Cmbx\_columnasSelectedIndexChanged**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_filas.Text != "")   
            {  
                m = Convert.**ToInt32**(cmbx\_filas.Text);  
                n = Convert.**ToInt32**(cmbx\_columnas.Text);  
                fun.**crearmatriz**(m, n,dgv\_matriz);  
                fun.**crearmatriz**(m, n, dgv\_opuesta);  
                fun.**crearmatriz**(n, m, dgv\_traspuesta);  
                fun.**error**(error, dgv\_matriz,"");  
            }  
            **else**  
            {  
                fun.**error**(error, dgv\_matriz,"NO HA ESCOGIDO EL NUMERO DE FILAS DE LA MATRIZ");  
            }  
        }  
          
        void **Btn\_calcularClick**(object sender, EventArgs e)  
        {  
            **if** (cmbx\_columnas.Text != "")   
            {  
                fun.**error**(error, dgv\_matriz, "");  
                **if** (dgv\_matriz.Rows.Count != 0 && !fun.**celdas\_vacias**(dgv\_matriz))  
                {  
                    fun.**error**(error, dgv\_matriz,"");  
                    fun.**opuesta**(m, n, dgv\_matriz, dgv\_opuesta);  
                    fun.**traspuesta**(m, n, dgv\_matriz, dgv\_traspuesta);  
                }  
                **else**  
                {  
                    fun.**error**(error, dgv\_matriz,"NO SE HAN CONSTRUIDO LAS MATRICES, HAY CELDAS VACIAS, O ESTAS DIGITANDO MAL LOS NUMEROS NEGATIVOS");  
                }  
            }  
            **else**   
            {  
                fun.**error**(error, dgv\_matriz, "AÚN NO HAS ESCOGIDO LAS COLUMNAS");  
            }           
        }  
          
        void **Dgv\_matrizKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**solo\_numeros**(dgv\_matriz, e);  
        }  
          
        void **Cmbx\_filasKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Cmbx\_columnasKeyPress**(object sender, KeyPressEventArgs e)  
        {  
            fun.**ninguna\_tecla**(e);  
        }  
          
        void **Dgv\_matrizEditingControlShowing**(object sender, DataGridViewEditingControlShowingEventArgs e)  
        {  
            TextBox celda = e.Control **as** TextBox;  
            celda.KeyPress -= **new** **KeyPressEventHandler**(Dgv\_matrizKeyPress);              
            celda.KeyPress += **new** **KeyPressEventHandler**(Dgv\_matrizKeyPress);              
        }  
    }  
}

**CLASE (FUNCIONES)🡪 public class funciones**

**using** System;  
**using** System.Collections.Generic;  
**using** System.Linq;  
**using** System.Text;  
**using** System.Windows.Forms;  
  
**namespace** matriz  
{  
    **public** class funciones  
    {  
        **int** i, j, k;  
          
        //funcion par permitir solo números positivos y negativos y la tecla borrar  
        **public** KeyPressEventArgs **solo\_numeros**(DataGridView m, KeyPressEventArgs e)  
        {  
            //comparo si el caracter es numero, control o negativo y no lo controlo para que se puedan aceptar  
               **if** (**char**.**IsNumber**(e.KeyChar) || **char**.**IsControl**(e.KeyChar) || e.KeyChar == Convert.**ToChar**("-"))  
               {  
                   e.Handled = **false**;  
               }  
            **else**  
            {  
                e.Handled = **false**;                  
               }  
            return e;  
        }  
             
           //funcion para no permitir teclas  
           **public** KeyPressEventArgs **ninguna\_tecla**(KeyPressEventArgs e)   
        {  
               //controlo el evento keypresseventargs para todas las teclas  
               e.Handled = **true**;  
            return e;  
        }  
             
           //metodo para mostrar errores  
           **public** void **error**(ErrorProvider error, DataGridView m, string mensaje)  
           {  
               //establezco en el error: primero, el control asociado y segundo el mensaje como tipo string  
               error.**SetError**(m, mensaje);  
           }  
             
           //metodo para crear matrices dependiendo si hay filas y columnas seleccionadas los en datagridviews  
           **public** void **crearmatriz**(**int** f,**int** c, DataGridView m1)   
           {  
               //verifico si las columnas estan vacias  
               **if**(c != 1 && c != 2 && c != 3 && c != 4 && c != 5 && c!= 6 && c != 7 && c!= 8 && c!= 9 && c != 10)  
               {  
                   //borro las columnas del datagridview  
                   m1.Columns.**Clear**();  
                    **for** (i = 0; i < 1; i++)  
                {  
                       // Creo una columna  
                       DataGridViewTextBoxColumn col1 = **new** **DataGridViewTextBoxColumn**();  
                       col1.Width = 30;  
                       col1.Name = "columna";  
                       col1.MaxInputLength = 2;  
                       col1.Resizable = DataGridViewTriState.False;  
                       ContextMenuStrip menu = **new** **ContextMenuStrip**();  
                       col1.ContextMenuStrip = menu;  
                       m1.Columns.**Add**(col1);  
                }  
                m1.Rows.**Add**(f);  
               }  
               **else**   
               {  
                   //borro las columnas del datagridview  
                   m1.Columns.**Clear**();  
                **for** (i = 0; i < c; i++)  
                {  
                       //creo una columna  
                       DataGridViewTextBoxColumn col1 = **new** **DataGridViewTextBoxColumn**();  
                       col1.Width = 30;  
                       col1.Name = "columna";  
                       col1.MaxInputLength = 2;  
                       col1.Resizable = DataGridViewTriState.False;  
                       ContextMenuStrip menu = **new** **ContextMenuStrip**();  
                       menu.IsAccessible = **false**;  
                       col1.ContextMenuStrip = menu;  
                       m1.Columns.**Add**(col1);  
                }  
                m1.Rows.**Add**(f);  
               }  
           }  
             
           //funcion para ver si hay celdas vacias o hay numeros mal digitados  
        /\*utilizo el for each porque con el for me queda mas dificil saber si la celda es vacía  
        ya que en c# no se puede hacer lo que se me ocurrio primero: if (m[i,j].value="")... ademas  
           un  for each hace el recorrido de manera completa sin importar el numero de filas o columnas\*/  
           **public** **bool** **celdas\_vacias**(DataGridView m)  
        {  
            **bool** vacia;  
            vacia = **false**;  
            **foreach** (DataGridViewRow fila **in** m.Rows)   
            {  
                //recorro las celdas de cada fila  
                **foreach** (DataGridViewCell celda **in** fila.Cells)   
                {  
                    // declaro un objeto y le asigno el valor de cada celda  
                    object valor = celda.Value;  
                    //comparo si ese valor es nulo, si el valor es de tipo dbnull, negativo sin numero o numero negativo mal digitado  
                    **if** ((valor == **null**) || (object.**ReferenceEquals**(valor, DBNull.Value)) || valor.**ToString**() == "-"  || valor.**ToString**() == "--" || valor.**ToString**() == "1-"  || valor.**ToString**() == "2-" || valor.**ToString**() == "3-"  || valor.**ToString**() == "4-" || valor.**ToString**() == "5-"  || valor.**ToString**() == "6-" || valor.**ToString**() == "7-"  || valor.**ToString**() == "8-" || valor.**ToString**() == "9-"  || valor.**ToString**() == "0-")  
                    {  
                        vacia = **true**;  
                    }  
                }  
            }  
            return vacia;  
        }  
                        
           //metodo para sumar matrices  
           **public** void **sumayresta**(**int** f, **int** c, DataGridView m1, DataGridView m2, DataGridView sum, DataGridView res)   
           {  
               **for** (i = 0; i < f; i++)  
            {  
                **for** (j = 0; j < c; j++)  
                {  
                    sum[j,i].Value = Convert.**ToInt32**(m1[j,i].Value) + Convert.**ToInt32**(m2[j,i].Value);  
                    res[j,i].Value = Convert.**ToInt32**(m1[j,i].Value) - Convert.**ToInt32**(m2[j,i].Value);  
                }  
               }  
           }  
             
           //metodo para multiplicar matriz por un escalar  
           **public** void **producto\_escalar**(**int** f, **int** c, DataGridView m1, DataGridView res, **int** num)   
           {  
               **for** (i = 0; i < f; i++)  
            {  
                **for** (j = 0; j < c; j++)  
                {  
                    res[j,i].Value = Convert.**ToInt32**(m1[j,i].Value) \* num;  
                }  
               }  
           }  
             
           //funcion para multiplicar matrices  
           **public** DataGridView **multiplicacion** (**int** f, **int** c, **int** c2, DataGridView m1, DataGridView m2, DataGridView res)   
           {  
               **for** (i = 0; i < f; i++)  
            {  
                **for** (j = 0; j < c2; j++)  
                {  
                    res[j,i].Value = 0;  
                    **for** (k = 0; k < c; k++)  
                    {  
                        res[j,i].Value = Convert.**ToInt32**(res[j,i].Value) + Convert.**ToInt32**(m1[k,i].Value) \* Convert.**ToInt32**(m2[j,k].Value);  
                    }  
                }  
               }  
               return res;  
           }  
             
           //funcion para verificar que una matriz es cuadrada  
           **public** **bool** **es\_cuadrada** (**int** f, **int** c)   
           {  
               **bool** cuad;  
               **if** (f != c)  
               {  
                   cuad = **false**;   
               }  
               **else**  
               {  
                   cuad = **true**;  
               }  
               return cuad;  
           }  
             
           //funcion para verificar que una matriz es nula  
           **public** **bool** **es\_nula** (**int** f, **int** c, DataGridView m)   
           {  
               **int** nul=0 , valm = 0;  
               **bool** nulo;  
               **for**(i=0; i<f; i++)  
               {  
                   **for**(j=0; j<c; j++)  
                   {  
                       valm = Convert.**ToInt32**(m[j,i].Value);  
                       **if**(valm == 0)  
                       {  
                           nul += 1;  
                       }  
                   }  
               }  
               **if** (nul == f \* c)  
               {  
                   nulo = **true**;   
               }  
               **else**  
               {  
                   nulo = **false**;  
               }  
               return nulo;  
           }  
  
           //funcion para verificar que una matriz es de identidad  
           **public** **bool** **es\_identidad** (**int** f, **int** c, DataGridView m)   
           {  
               **int** ceros=0 , unos = 0, valm = 0;  
               **bool** identi;  
               **if** (f == c)  
               {  
                   **for**(i=0; i<f; i++)  
                   {  
                       **for**(j=0; j<c; j++)  
                       {  
                           valm = Convert.**ToInt32**(m[j,i].Value );  
                           **if**(i == j && valm == 1)  
                           {  
                               unos += 1;  
                           }  
                           **else** **if**(i != j && valm == 0)  
                           {  
                               ceros += 1;  
                           }  
                       }  
                   }  
                   **if** (ceros == f \* c - f && unos == f)  
                   {  
                       identi = **true**;   
                   }  
                   **else**  
                   {  
                       identi = **false**;  
                   }                 
               }  
            **else**   
            {  
                identi = **false**;  
            }  
            return identi;  
           }  
             
           //funcion para verificar que una matriz es diagonal  
           **public** **bool** **es\_diagonal** (**int** f, **int** c, DataGridView m)   
           {  
               **int** ceros=0, num=0, valm = 0;  
               **bool** diago;  
               **if** (f == c)  
               {  
                   **for**(i=0; i<f; i++)  
                   {  
                       **for**(j=0; j<c; j++)  
                       {  
                           valm = Convert.**ToInt32**(m[j,i].Value );  
                           **if**(i == j && valm != 0)  
                           {  
                               num += 1;  
                           }  
                           **else** **if**(i != j && valm == 0)  
                           {  
                               ceros += 1;  
                           }  
                       }  
                   }  
                   **if** (ceros == f \* c - f && num == f)  
                   {  
                       diago = **true**;   
                   }  
                   **else**  
                   {  
                       diago = **false**;  
                   }                 
               }  
            **else**   
            {  
                diago = **false**;  
            }  
            return diago;  
           }  
             
           //funcion para verificar que una matriz es escalar  
           **public** **bool** **es\_escalar** (**int** f, **int** c, DataGridView m)   
           {  
               **int** ceros=0, num, acum=0, valm = 0;  
               num = Convert.**ToInt32**(m[0,0].Value);  
               **bool** esc;  
               **if** (f == c)  
               {  
                   **for**(i=0; i<f; i++)  
                   {  
                       **for**(j=0; j<c; j++)  
                       {  
                           valm = Convert.**ToInt32**(m[j,i].Value );  
                           **if**(j == i && valm == num)  
                           {  
                               acum += 1;  
                           }  
                           **else** **if**(i != j && valm == 0)  
                           {  
                               ceros += 1;  
                           }  
                       }  
                   }  
                   **if** (ceros == f \* c - f && acum == f)  
                   {  
                       esc = **true**;   
                   }  
                   **else**  
                   {  
                       esc = **false**;  
                   }                 
               }  
            **else**   
            {  
                esc = **false**;  
            }  
            return esc;  
           }  
             
           //funcion para verificar que una matriz es simetrica  
           **public** **bool** **es\_simetrica**(**int** f, **int** c, DataGridView m)   
           {  
               **int** valm = 0, valmsim = 0;  
               **bool** sime = **true**;  
               **if** (f == c)  
               {  
                   **for**(i=0; i<f; i++)  
                   {  
                       **for**(j=0; j<c; j++)  
                       {  
                           valm = Convert.**ToInt32**(m[i,j].Value);  
                           valmsim = Convert.**ToInt32**(m[j,i].Value);  
                           **if**(valm != valmsim)  
                           {  
                               sime = **false**;  
                               i = f;  
                               j=c;  
                           }  
                       }  
                   }               
               }  
            **else**   
            {  
                sime = **false**;  
            }  
            return sime;  
           }  
             
           //funcion para verificar que una matriz es anti-simetrica  
           **public** **bool** **es\_antisimetrica**(**int** f, **int** c, DataGridView m)   
           {  
               **int** valm = 0, valmantisim = 0;  
               **bool** antisime = **true**;  
               **if** (f == c)  
               {  
                   **for**(i=0; i<f; i++)  
                   {  
                       **for**(j=0; j<c; j++)  
                       {  
                           valm = Convert.**ToInt32**(m[i,j].Value);  
                           valmantisim = Convert.**ToInt32**(m[j,i].Value);  
                           **if**(valmantisim != -valm)  
                           {  
                               antisime = **false**;  
                               i = f;  
                               j=c;  
                           }  
                       }  
                   }               
               }  
            **else**   
            {  
                antisime = **false**;  
            }  
            return antisime;  
           }  
             
           //funcion para mostrar una matriz opuesta  
           **public** DataGridView **opuesta** (**int** f, **int** c, DataGridView m, DataGridView res)   
           {  
               **for**(i=0; i<f; i++)  
               {  
                   **for**(j=0; j<c; j++)  
                   {  
                       res[j,i].Value = Convert.**ToInt32**(m[j,i].Value) \* -1 ;  
                   }  
               }  
               return res;  
           }  
             
           //funcion para mostrar una matriz traspuesta  
           **public** DataGridView **traspuesta** (**int** f, **int** c, DataGridView m, DataGridView res)   
           {  
               **for**(i=0; i<f; i++)  
               {  
                   **for**(j=0; j<c; j++)  
                   {  
                       res[i,j].Value = Convert.**ToInt32**(m[j,i].Value);  
                   }  
               }  
               return res;  
           }  
    }  
}