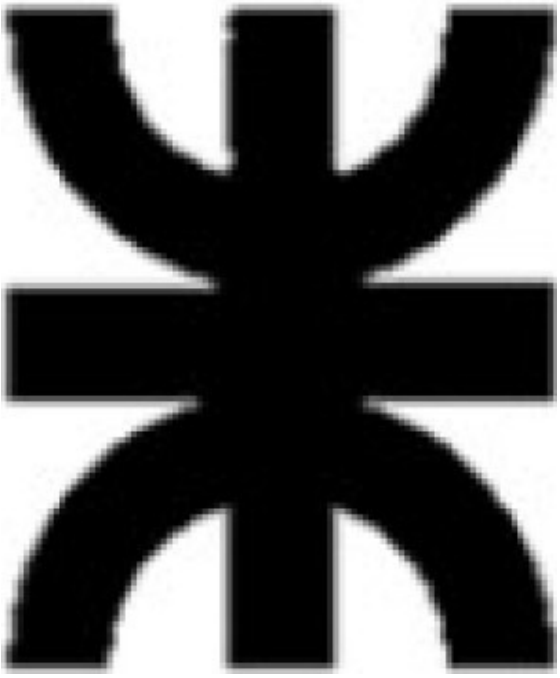


UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional Reconquista

Programación en Computación

Ciclo Lectivo: 2020

Trabajo Práctico N.º 9

9 – MATRICES

GRUPO N.º: 12
INTEGRANTES: Fulano, Mengano, Zutano

Guía Unidad 9: MATRICES

SITUACIONES PROBLEMÁTICAS:

1. Diseñar un algoritmo que imprima cuantos de los números almacenados son ceros, cuantos son positivos y cuantos son negativos en una matriz de m x n.
2. Dada una matriz M, imprimir el promedio general y el porcentaje de nulos sobre la cantidad de elementos de la matriz.
3. Hacer un algoritmo que determine la posición del número mayor almacenado en la matriz. Los números son diferentes.
4. Hallar e imprimir el máximo, mínimo y promedio de una matriz general de n filas y m columnas.
5. Realizar un algoritmo que imprima el porcentaje de números mayores a 10, el promedio de los elementos impares y la suma total de una matriz general.
6. Realizar un algoritmo que determine si en una matriz hay más pares o impares.
7. Imprimir cantidad de elementos nulos en cada fila y el menor número de una matriz.
8. Dada una matriz de n filas y m columnas, hallar e imprimir el máximo elemento de cada una de sus columnas.
9. Imprimir el mayor de cada fila, el promedio general y el porcentaje de números positivos de cada fila de una matriz general n filas y m columnas.
10. Hacer un algoritmo que halle la suma de cada columna e imprima que columna tuvo la máxima suma y la suma de esa columna.
11. Generar una matriz de m x n elementos. Imprimir promedio general, cantidad de elementos negativos, porcentaje de nulos, suma de los positivos y el menor de cada fila.
12. **Diseñar un algoritmo que cargue una Matriz m x n y almacene la suma de las filas en un vector. Imprimir el vector resultante.**

ACTIVIDADES:

Resolver las situaciones problemáticas anteriores, debidamente comentadas y comenzando con:

1. **ANALIZAR** el Problema, *Datos de Entrada, Salida y Auxiliares, y ¿Que ocurre sí?...*
2. **Diagrama de Flujo** en RAPTOR,
3. Codificación en **VSC# Ventana**. (Windows Form)

CONOCIMIENTOS NECESARIOS:

U_2) Algoritmos.

U_3) Tipos de Datos.

U_4) Diagramación lógica. Diagramas de Flujo.

U_5) Estructuras Secuenciales. Uso de Asignaciones, Entradas y Salidas de Datos.

U_6) Estructuras de selección o condicionales, estructuras de selección múltiple. Uso de condicionales para la formulación de algoritmos.

U_7) Estructuras repetitivas: HACER-PARA, HACER-MIENTRAS, REPETIR-HASTA.

U_8) Vectores. Utilización del tipo de dato vector.

U_9) Matrices. Índices. Recorridos por filas y columnas. Búsquedas de mayores y menores por filas y columnas de la matriz. Casos de utilización de matrices.

OPERATORIA:

Varia con cada situación problemática presentada, pero se resuelve en base a los conocimientos necesarios considerados.

RECOMENDACIÓN:

Para proceder a la realización de los TPs de esta Guía, es recomendable releer la siguiente bibliografía para un repaso de los conceptos teóricos involucrados en resolución de estos problemas:

Diseño_de_Algoritmos.pdf, En especial Capítulo VI y VII.

Problema 12:

Diseñar un algoritmo que cargue una Matriz m x n y almacene la suma de las filas en un vector. Imprimir el vector resultante.

1) Análisis del Problema:

El problema nos pide como Datos iniciales cargar una Matriz m x n; en consecuencia, como DATOS de Entrada necesitamos pedir al Usuario que cargue las Dimensiones de la Matiz, y validarlas para que sean Datos válidos (Números Naturales), y con ellos generar una Matriz y cargar sus Elementos:

DATOS de ENTRADA:

Números Enteros mayores a cero: M y N (Filas y Columnas de la Matriz), y Números Reales: los elementos de Matriz, M [i, j]

Una vez obtenida la Matriz, nos pide realizar la suma de los elementos de cada Fila y asignarlos a un Vector.

DATOS de Proceso o Cálculo:

Números Reales: Sumadores, Vector V [i] = $\sum_{j=1}^{n} M[i,j]$ para todo i.

Finalmente nos solicita Mostrar los Elementos del Vector Resultante.

DATOS de SALIDA:

Números Reales: V [i], para todo i.

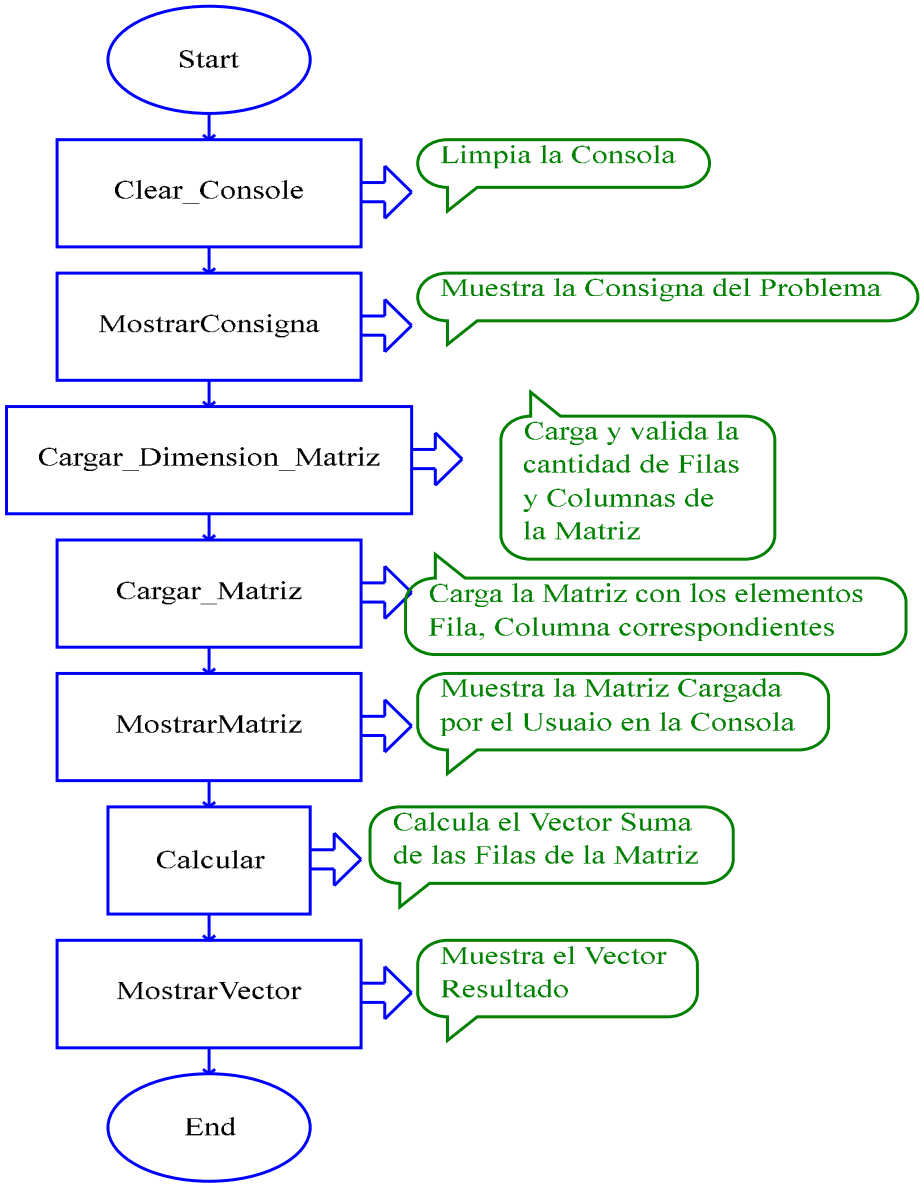
DATOS Auxiliares:

De Texto: Mensajes para el Usuario,

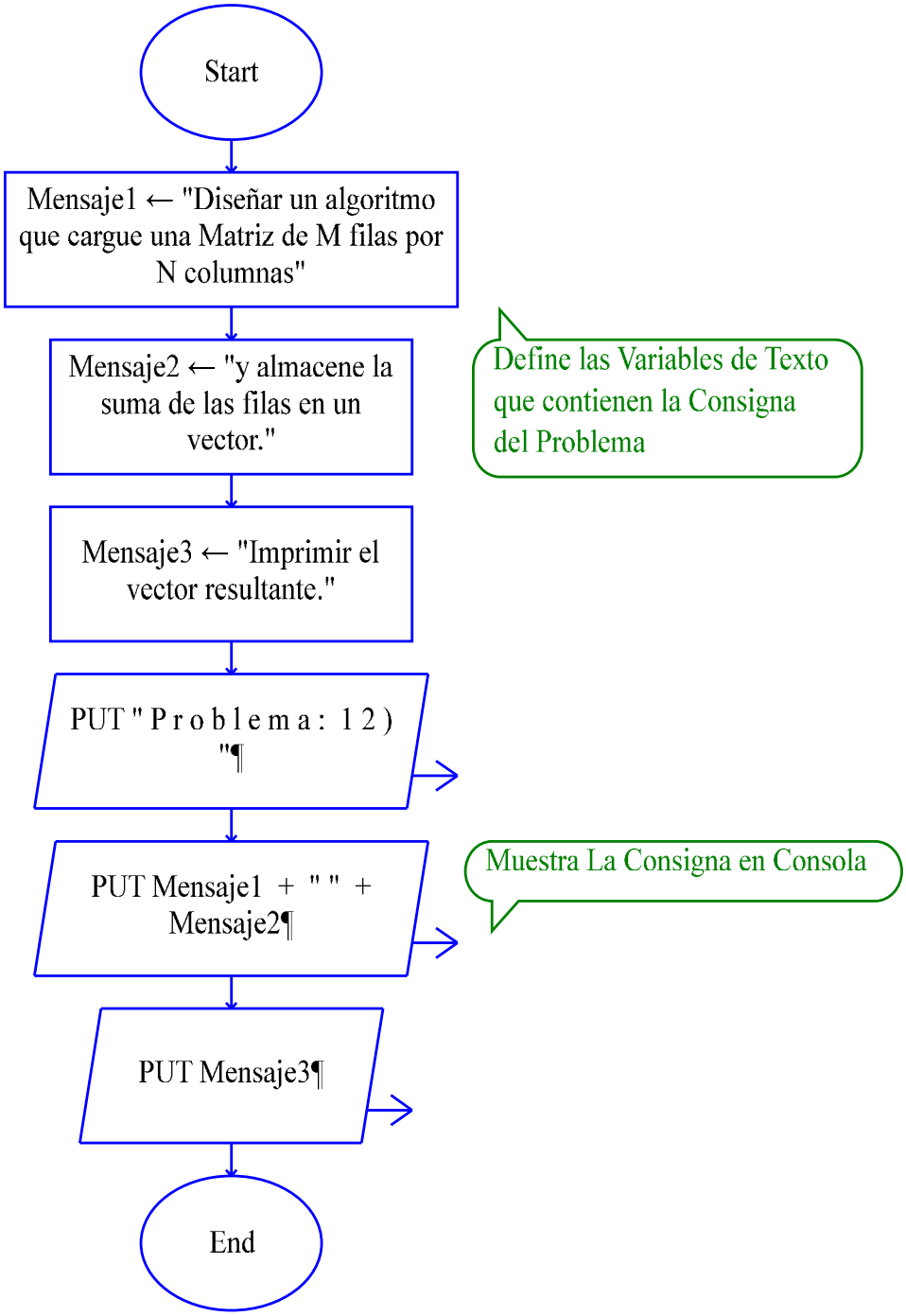
Números Naturales: Fila, Columna, m, n, i, j, Contadores.

2) Diagrama de Flujo en RAPTOR,

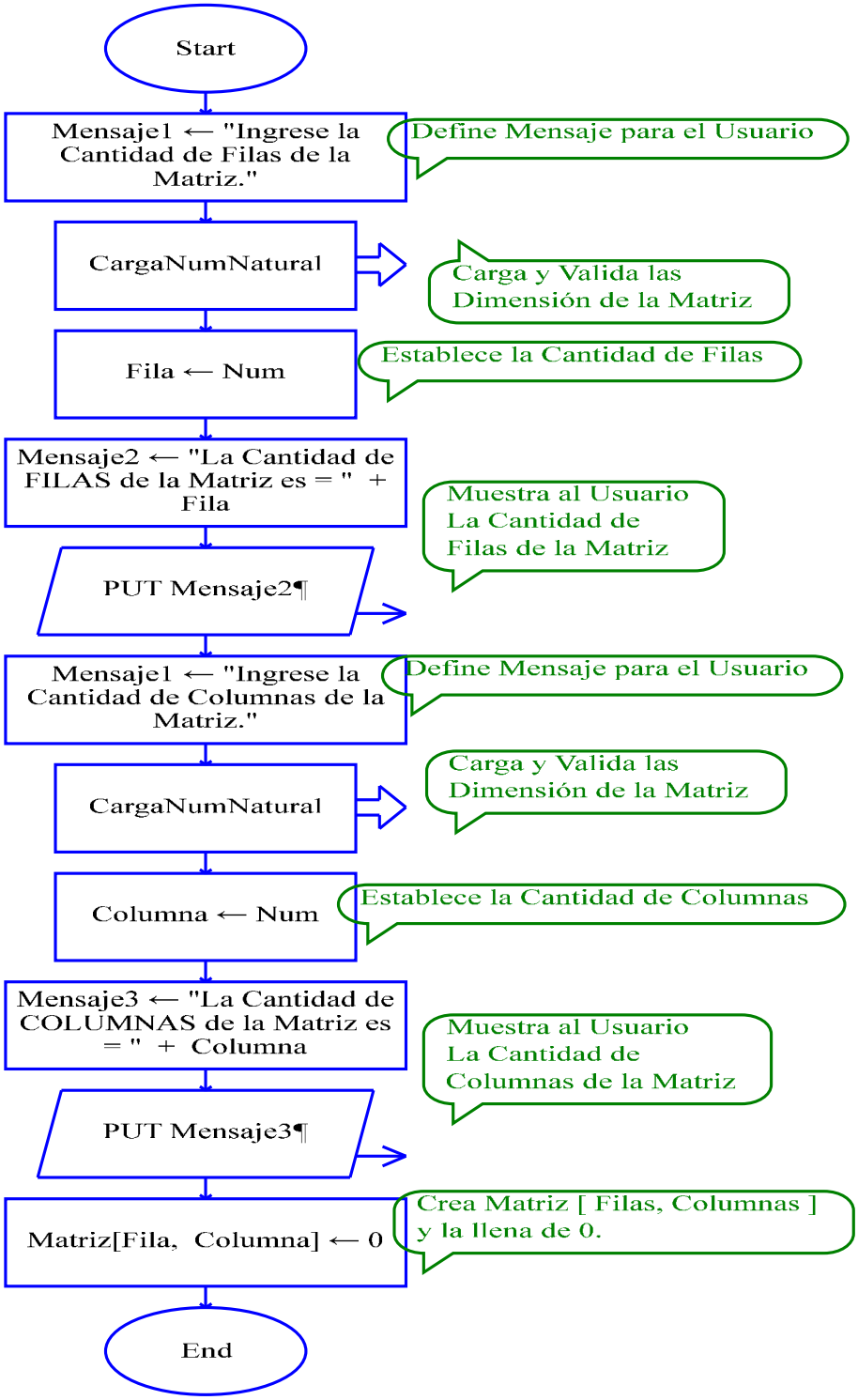
Al *Diagrama de Flujo en RAPTOR*, lo construiremos en base a los lineamientos surgidos del ANALISIS del problema. Para concentrarnos en el Manejo de Matriz y Vector, hemos dejado de lado la presentación en Ventana Gráfica y solo usamos la salida de Consola. Emplearemos la técnica de Refinamiento Progresivo con el uso de Módulos. Los Diagramas están auto explicados a través de sus comentarios. Programa "**main**"



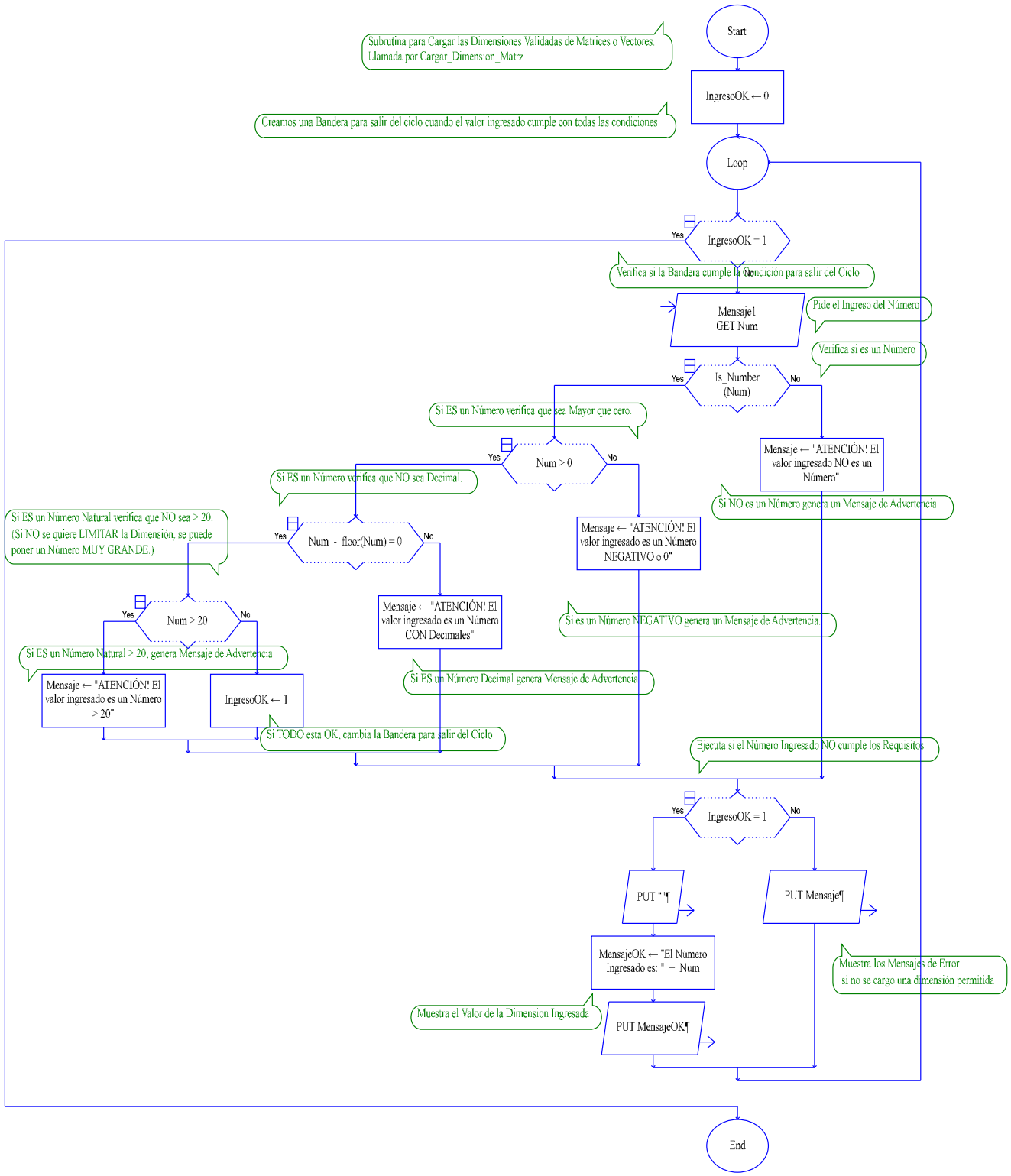
Sub Programa **“MostrarConsigna”**



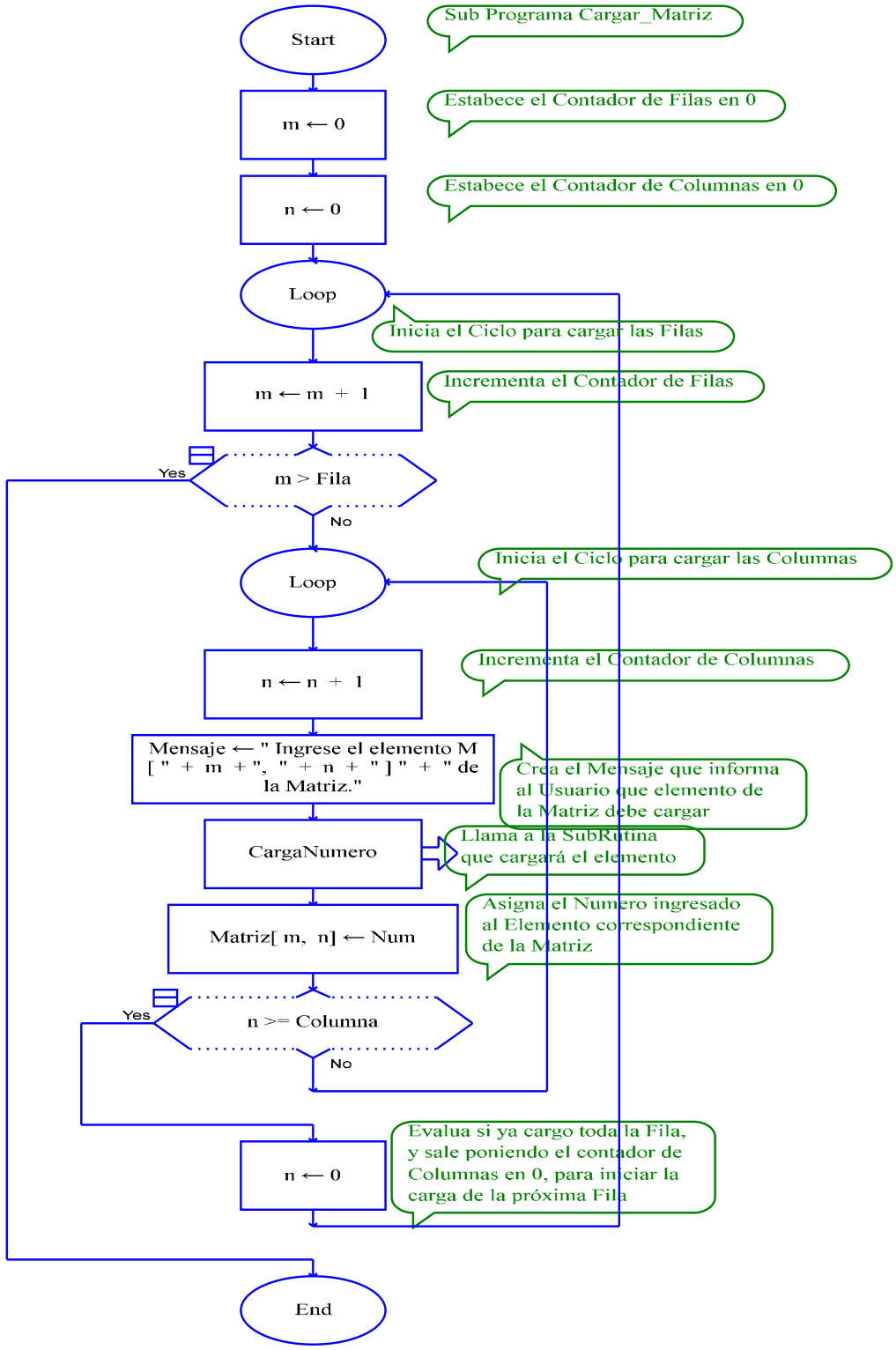
Sub Programa **“Cargar_Dimension_Matriz”**



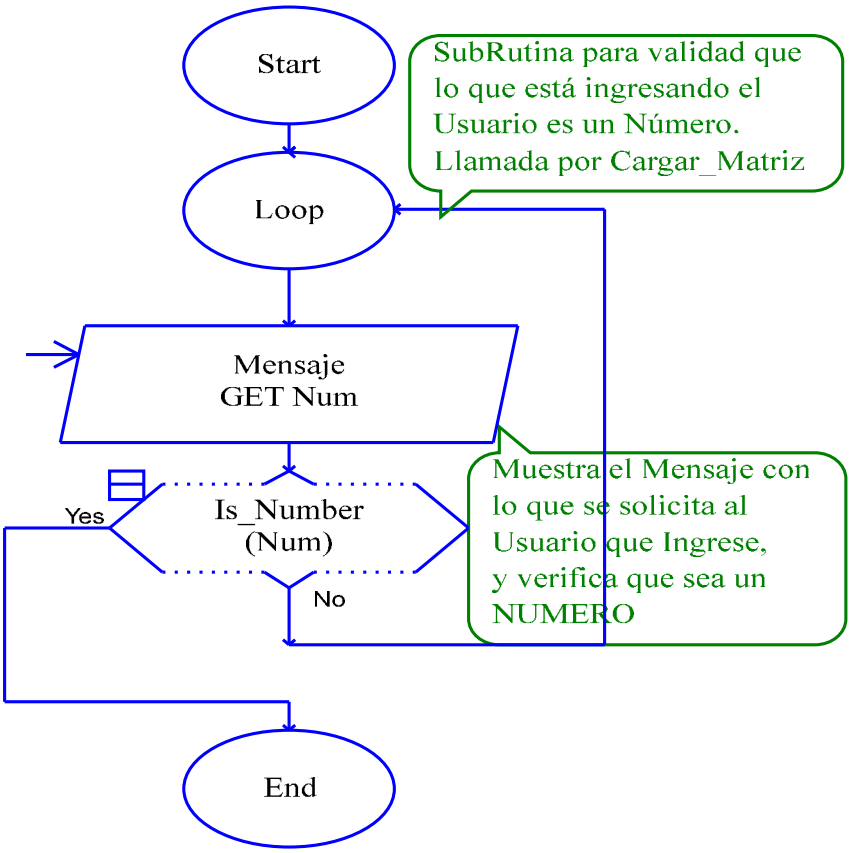
El Sub Programa anterior llama a un Sub Sub Programa **"CargaNumNatural"**



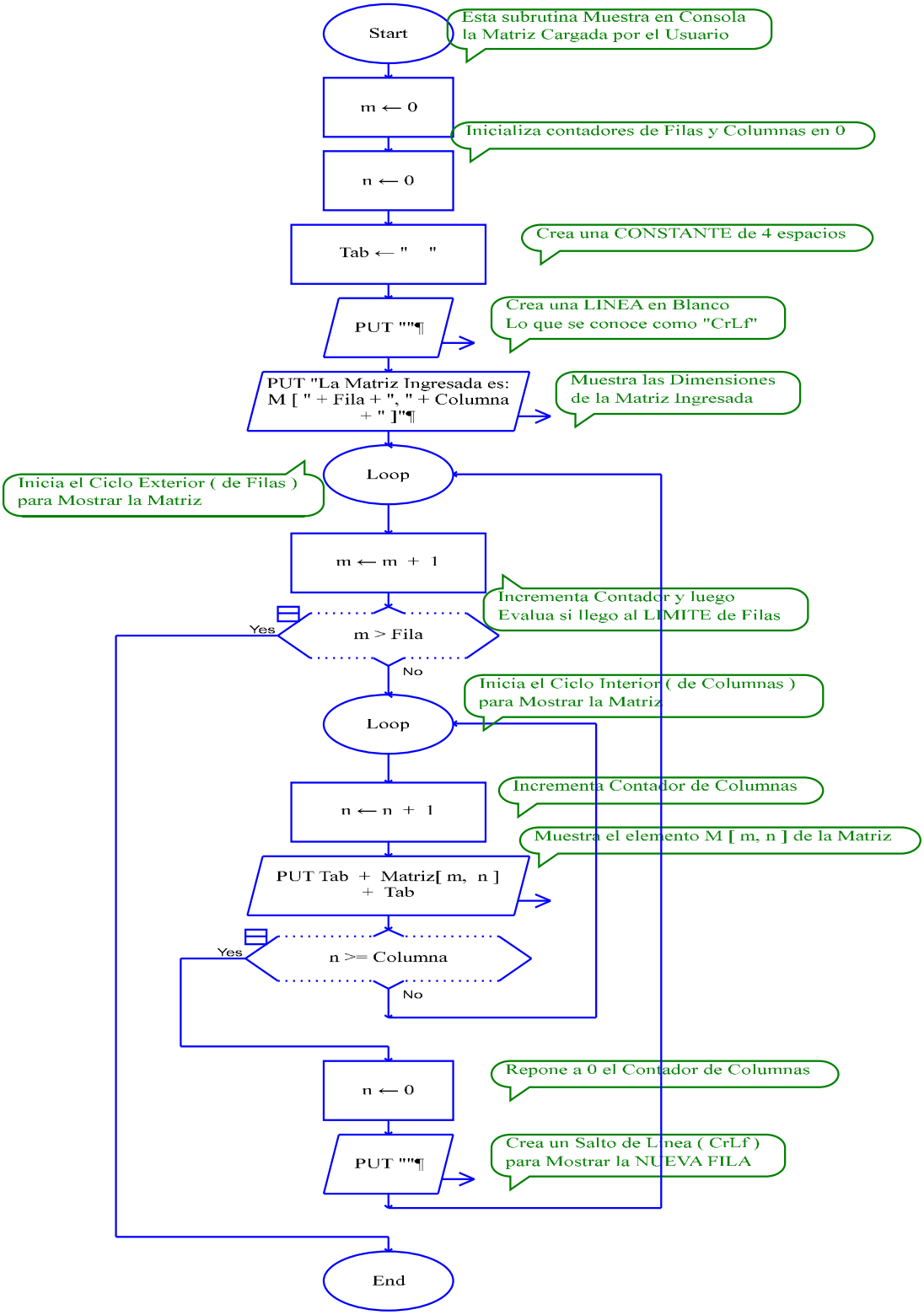
Sub Programa "**Cargar_Matriz**"



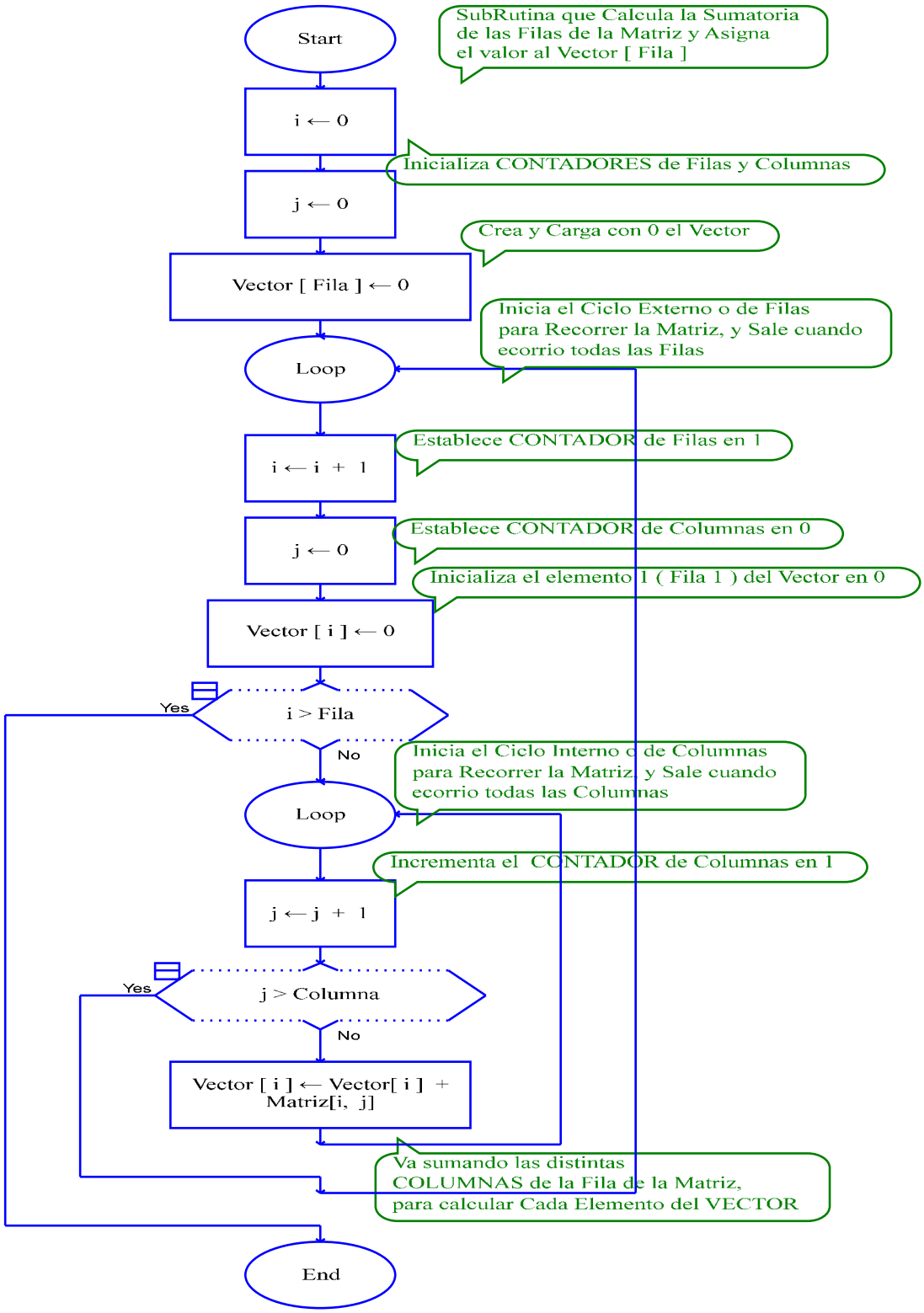
El Sub Programa anterior llama a un Sub Sub Programa **"CargaNumero"**



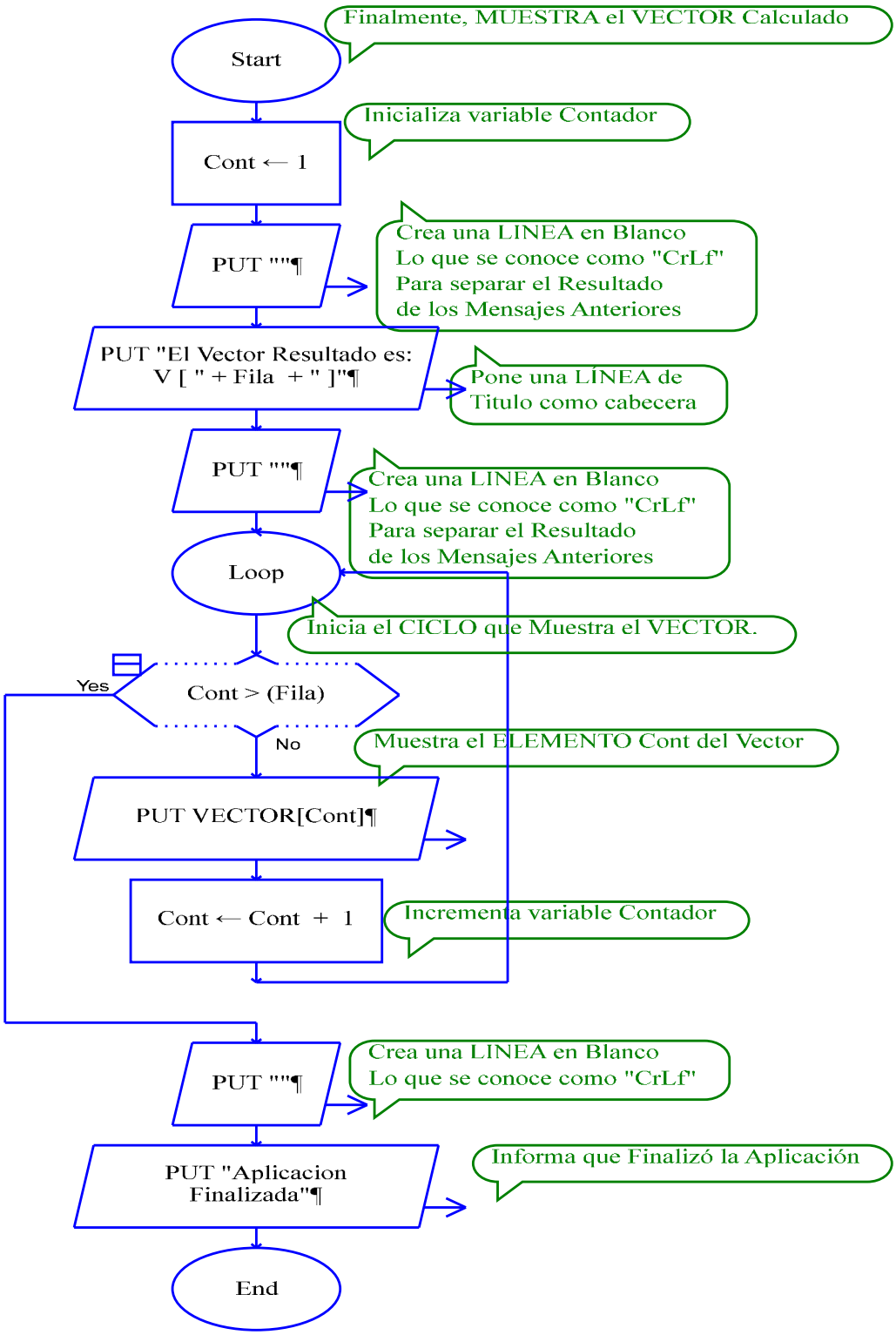
Sub Programa "**MostrarMatriz**"



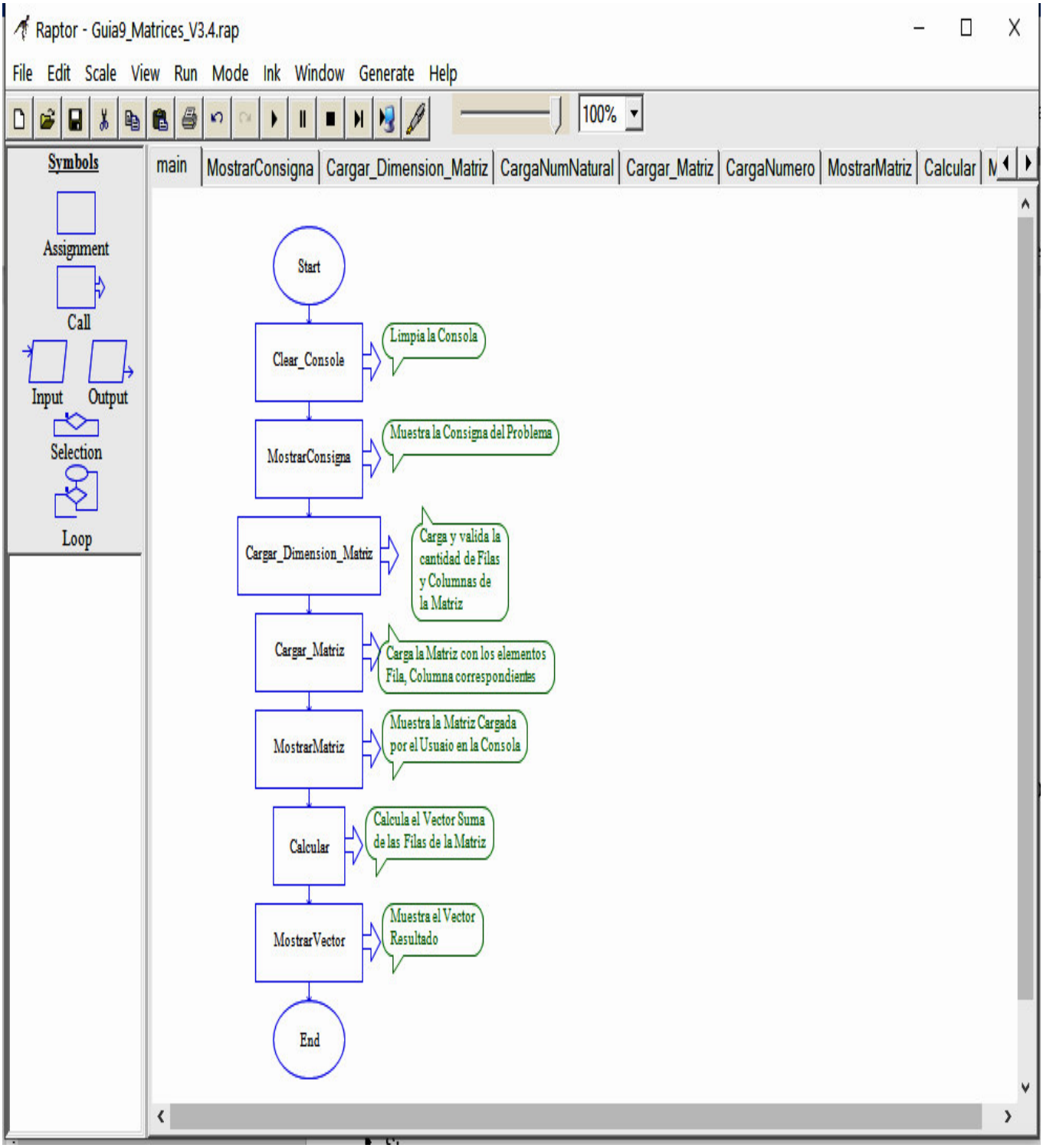
Sub Programa "**Calcular**"



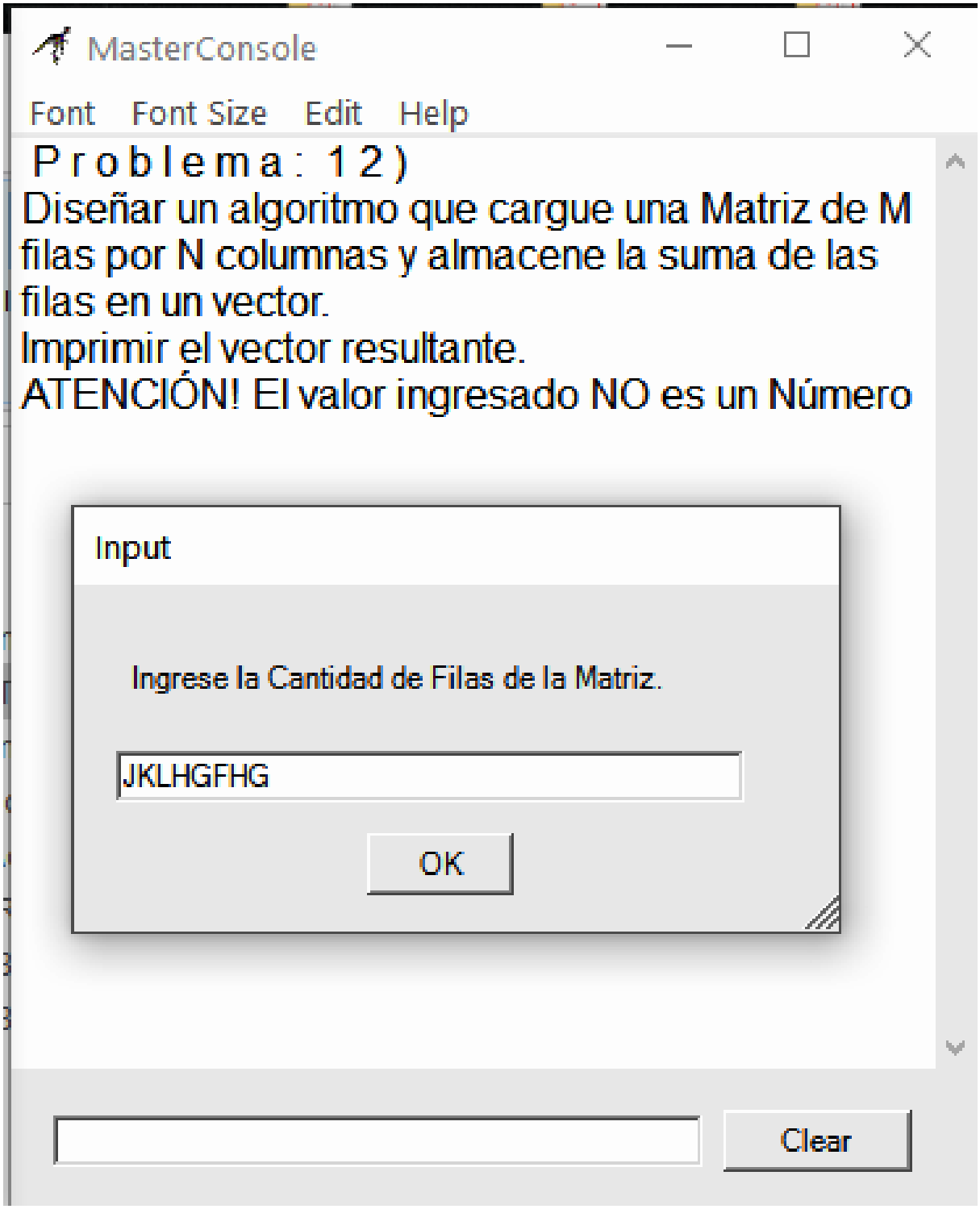
Sub Programa "**MostrarVector**"



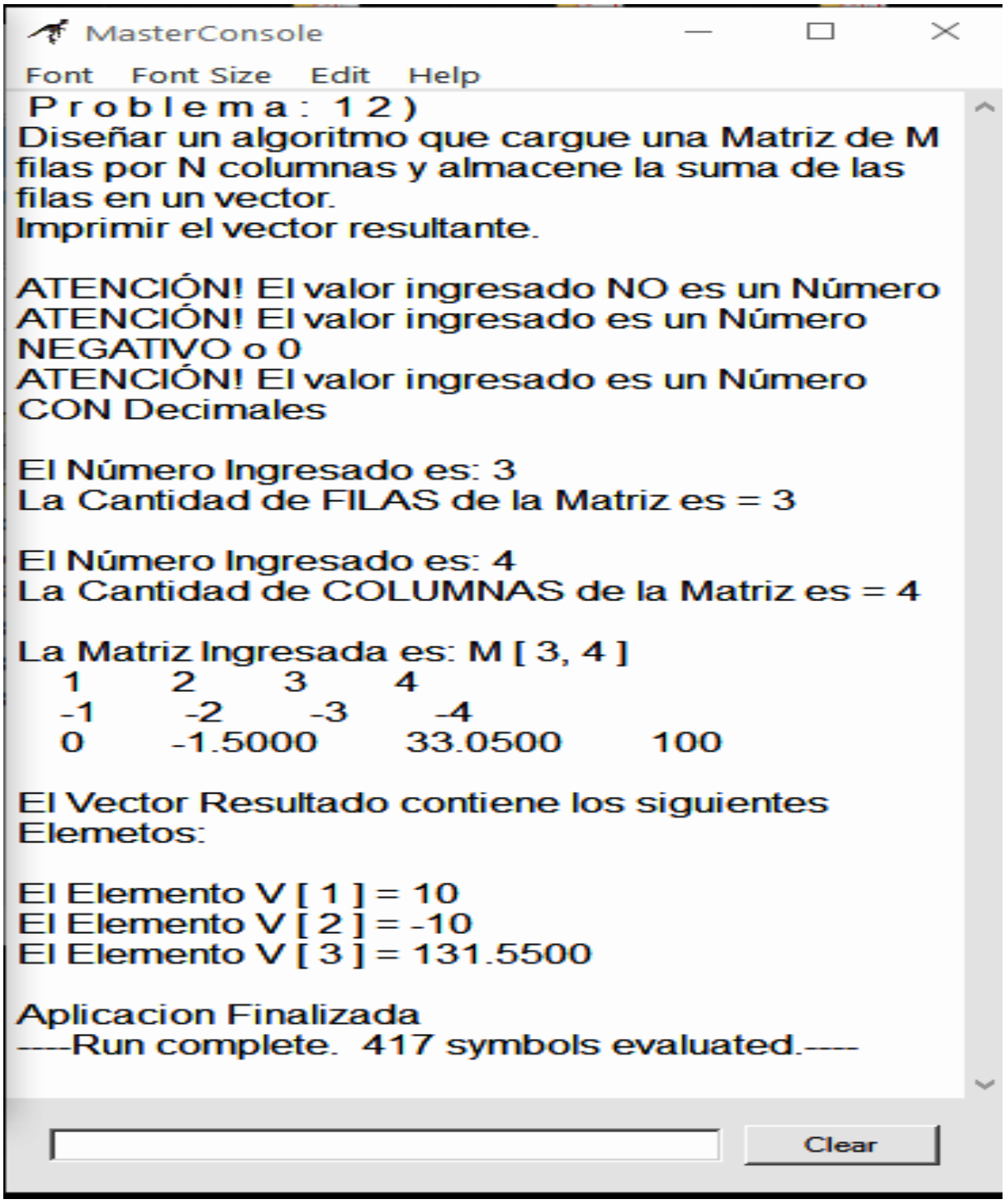
El aspecto del IDE de RAPTOR, al finalizar el *Diagrama de Flujo* es el Siguiente:



Al *D.F. de RAPTOR*, lo ejecutamos y nos muestra sus Salidas por Consola:



Salida General luego de Cargar las Dimensiones de la Matriz y sus elementos y realizar el Cálculo Correspondiente y Mostrar el Vector Resultado:

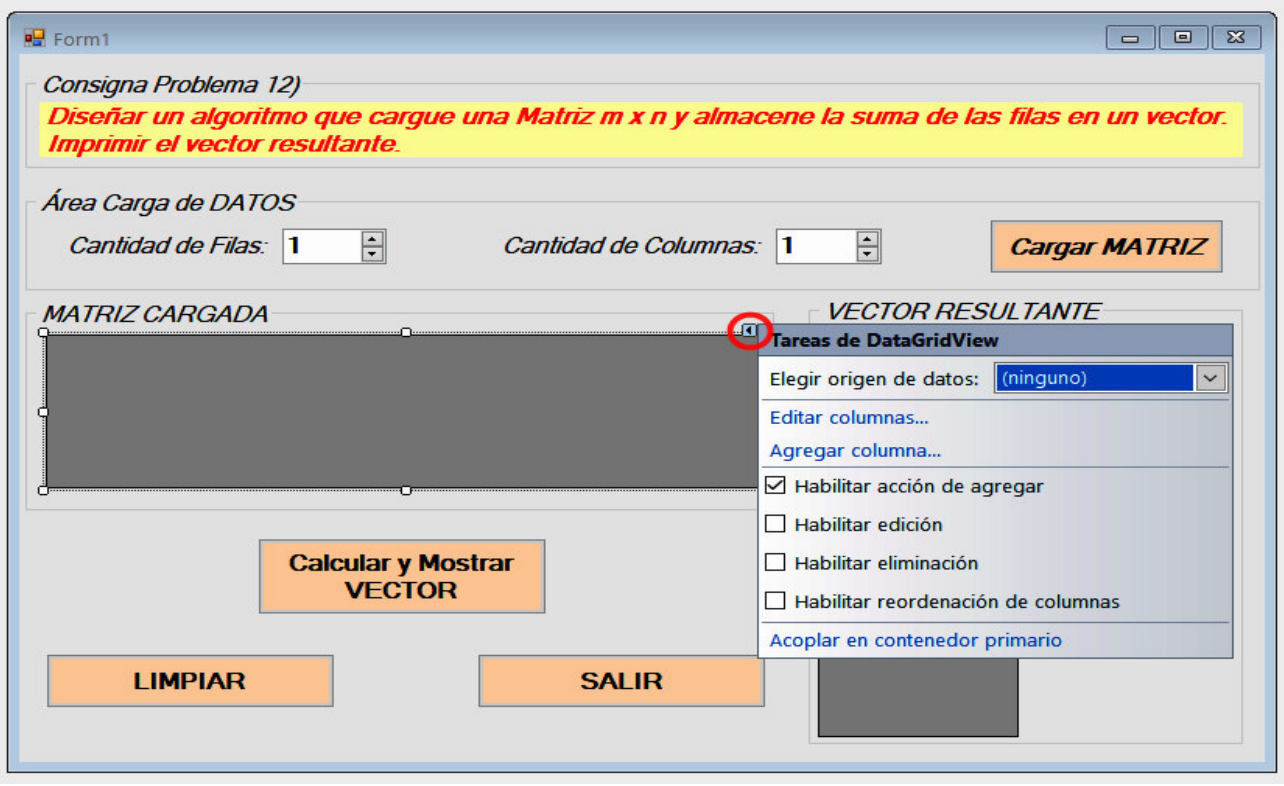


3) Codificación en C# (Windows From).

Realizaremos el programa en C# Ventana como una aplicación nueva y totalmente independiente.

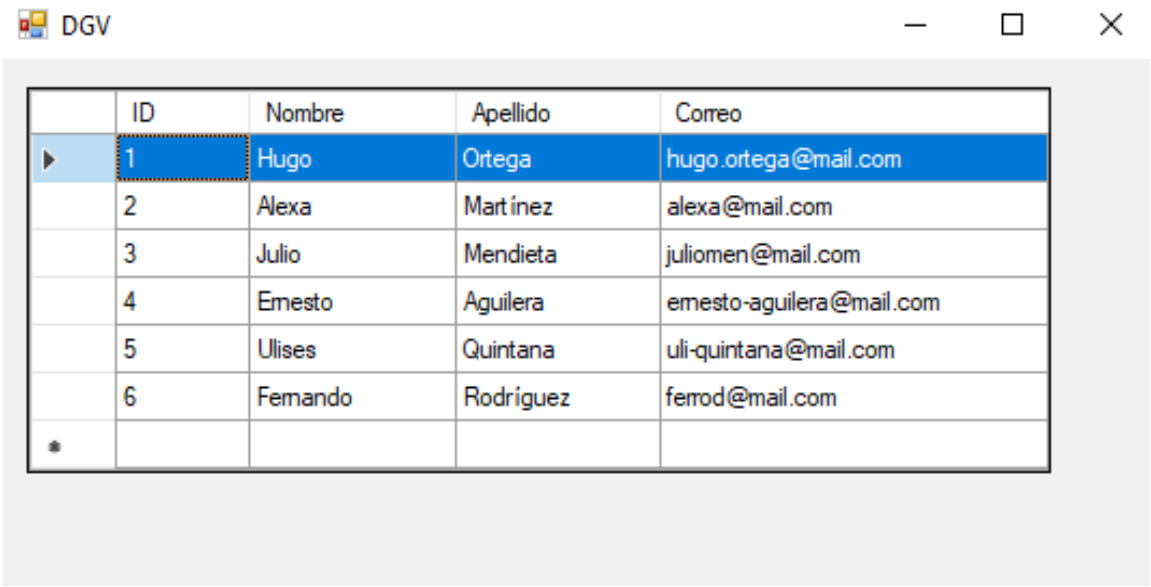
Como siempre que debemos programa con Interfaz Gráfica de Usuario (GUI), es conveniente comenzar diseñando ésta.

1. Esta GUI es muy simple, comienza con un GroupBox que contiene una Label con la Consigna del Problema que resuelve.
2. Continua con un Área para la Carga de Datos formada por dos NumericUpDown que No permiten decimales, Ni valores negativos, para cargar las Dimensiones (Filas y Columnas) de la Matriz y continua con un Botón para cargar y Validar los Elementos de la Matriz.
3. Continua con otro GroupBox que contiene un DataGridView que utilizaremos para Mostrar los Elementos de la Matriz cargada.
4. Mas abajo tenemos el Área de Procesos con los Botones: Calcular Vector, Limpiar y Salir.
5. Finaliza con otro GroupBox llamado VECTOR RESULTANTE, que contiene un DataGridView y un ComboBox, para Mostrar en dos formatos distintos el Vector Calculado. El Texto que nos muestra el ComboBox al inicio, indica que el mismo está Vacío y, en consecuencia, el botón Limpiar NO está habilitado. Lo nuevo de esta GUI es el uso del Control DataGridView, que necesitaremos aprender a utilizar para este Práctico y para el Examen Final.



¿Qué es y para qué sirve un Datagridview?

Un Datagridview es un control en Windows Forms el cual te permite mostrar información al usuario en forma de una tabla.



	ID	Nombre	Apellido	Correo
▶	1	Hugo	Ortega	hugo.ortega@mail.com
	2	Alexa	Martínez	alexa@mail.com
	3	Julio	Mendieta	juliomen@mail.com
	4	Ernesto	Aguilera	ernesto-aguilera@mail.com
	5	Ulises	Quintana	uli-quintana@mail.com
	6	Fernando	Rodríguez	ferrod@mail.com
*				

Un Datagridview te brinda la posibilidad de obtener la información que deseas presentar al usuario desde una base de datos o incluso desde un servicio web.

Por otro lado, también te permite agregar manualmente los datos que deseas sin necesidad de conectarte a una fuente de datos como puede ser una base de datos.

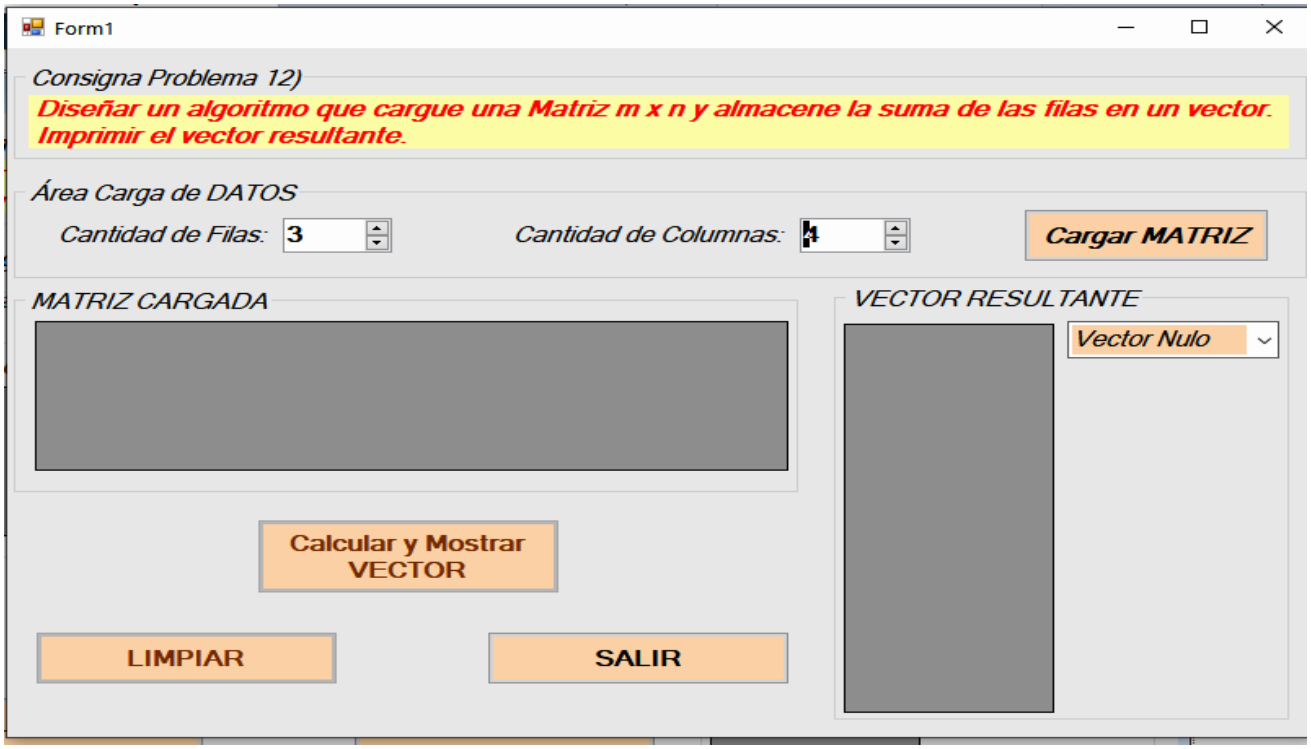
Una de las ventajas de este control es que además de mostrar información, también te da la posibilidad de agregar, modificar e incluso de eliminar datos directamente en la base de datos por lo que puede resultar muy útil y práctico manejar la información con este control.

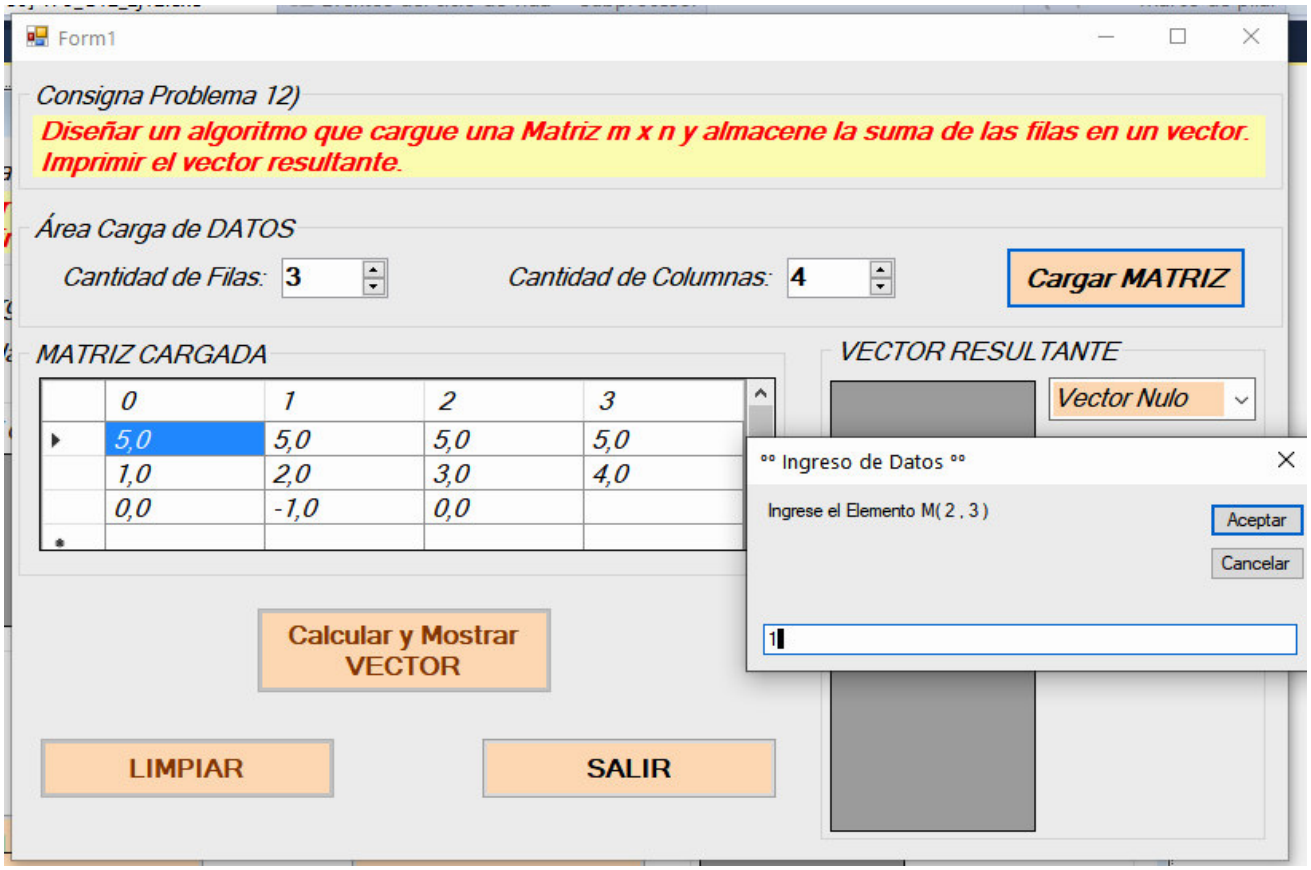
Otra de las ventajas es que puedes personalizar a tu gusto y/o necesidad la apariencia del control estableciendo, por ejemplo:

- Los colores de fila
- El color de fondo del control
- El alto y ancho de las filas y/o columnas
- La alineación del texto
- La fuente
- Entre otros

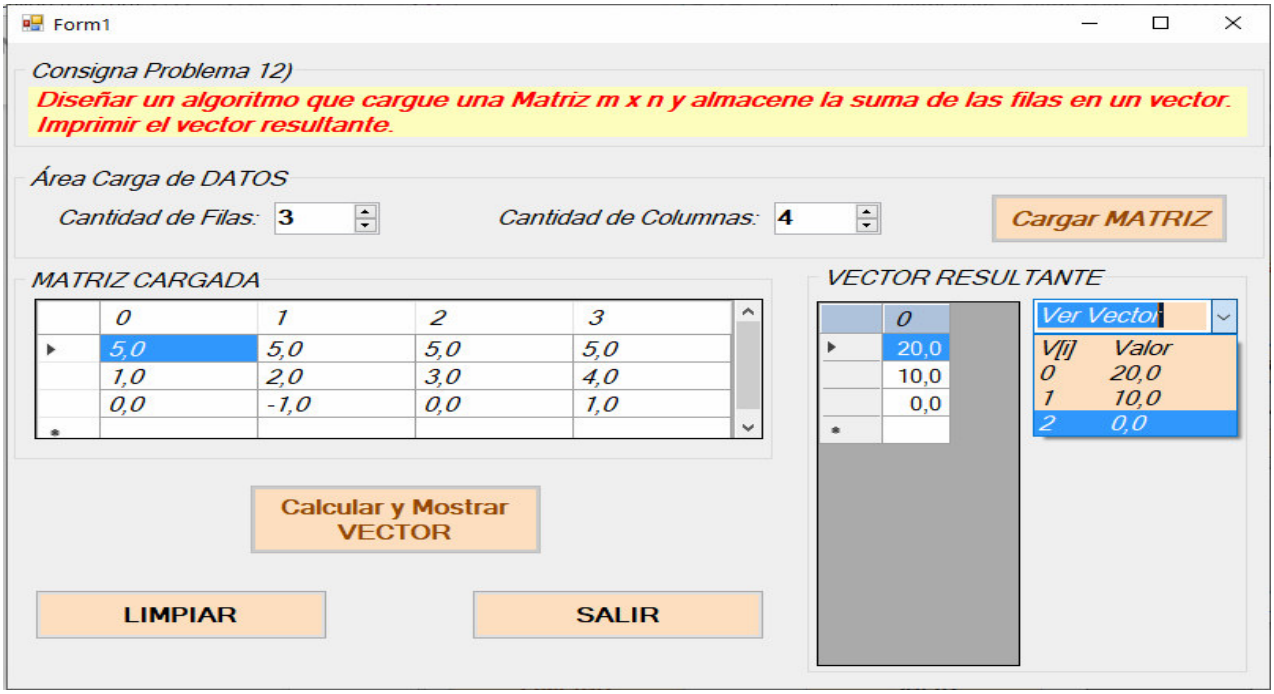
Fuente: https://envb.net/datagridview/

Al Iniciar el programa, establecemos las dimensiones de la Matriz y Ejecutamos un clic sobre el Botón Cargar Matriz, y procederemos a la carga de la misma a través de Ventanas de Input validadas.


Form1
Consigna Problema 12)
Diseñar un algoritmo que cargue una Matriz m x n y almacene la suma de las filas en un vector.
Imprimir el vector resultante.
Área Carga de DATOS
Cantidad de Filas: 3
Cantidad de Columnas: 4
Cargar MATRIZ
MATRIZ CARGADA
VECTOR RESULTANTE
Vector Nulo
Calcular y Mostrar VECTOR
LIMPIAR
SALIR

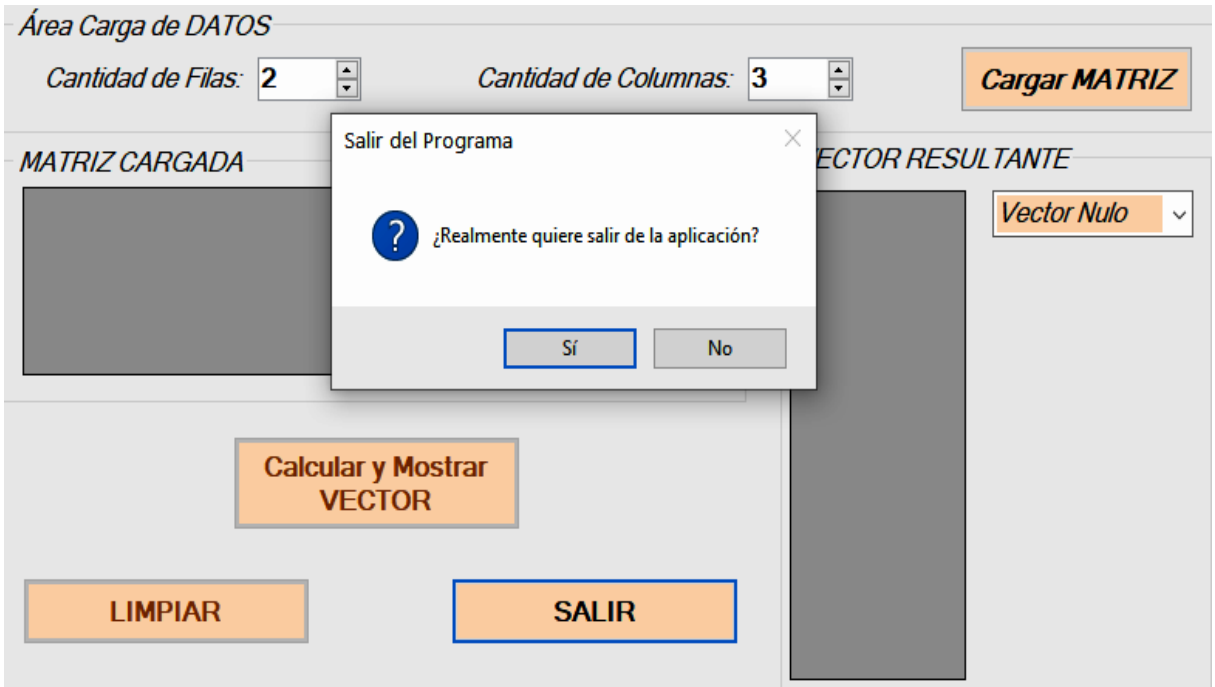

Form1
Consigna Problema 12)
Diseñar un algoritmo que cargue una Matriz m x n y almacene la suma de las filas en un vector.
Imprimir el vector resultante.
Área Carga de DATOS
Cantidad de Filas: 3
Cantidad de Columnas: 4
Cargar MATRIZ
MATRIZ CARGADA
VECTOR RESULTANTE
Vector Nulo
0 1 2 3
5,0 5,0 5,0 5,0
1,0 2,0 3,0 4,0
0,0 -1,0 0,0
°° Ingreso de Datos °°
Ingrese el Elemento M(2 , 3)
Aceptar
Cancelar
1
Calcular y Mostrar VECTOR
LIMPIAR
SALIR

Al finalizar la carga de la matriz se nos habilitará el botón calcular y mostrar matriz el cual nos ofrecerá la siguiente ventana. En este punto los únicos botones habilitados serán los de LIMPIAR para reiniciar el cálculo y la carga y SALIR para finalizar el Programa.

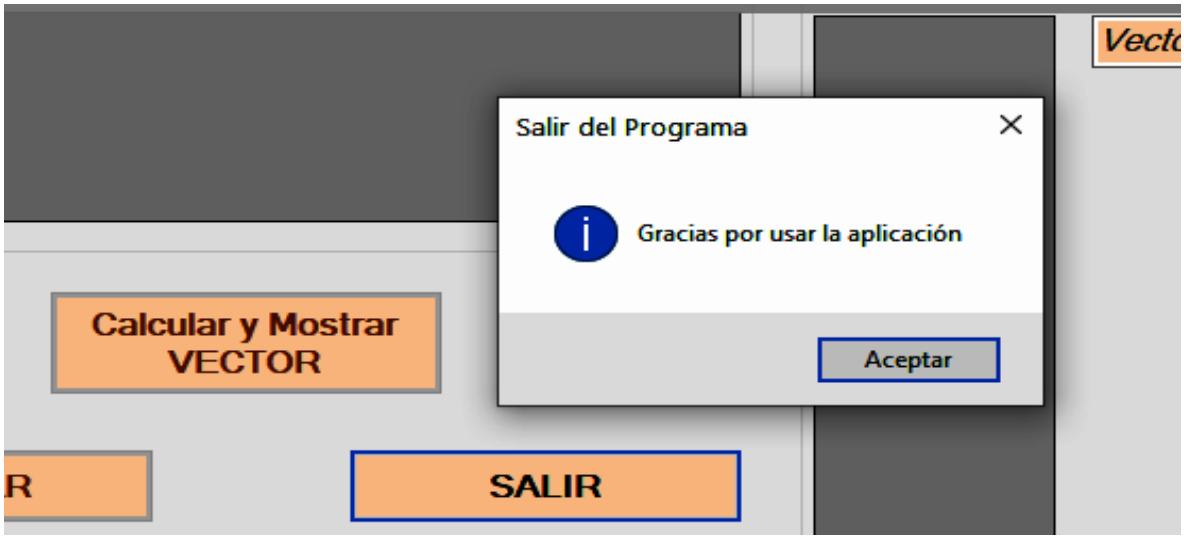


Si presionamos el botón LIMPIAR volveremos al estado inicial el programa, con solo los botones Cargar Matriz y SALIR habilitados.

Si presionamos el botón SALIR, aparece una ventana donde confirmar nuestra decisión.



Y si la confirmamos, aparece una ventana que nos agradece haber usado la Aplicación.



Veamos ahora el código de esta aplicación que maneja entrada de datos numéricos validados, así como la definición y carga de Matriz y Vector y la muestra de ellos en DataGridView.

Form1.cs [Diseño] Form1.cs

TP9_G12_Ej12.Form1 BtnLIMPIAR_Click(object sender, EventArgs e)

```
10
11  namespace TP9_G12_Ej12
12  {
13      public partial class Form1 : Form
14      {
15          public Form1()
16          {
17              InitializeComponent();
18          }
19
20          // Defino e inicializo varialbes globales (Tipo RAPTOR), que usaré en varios Módulos
21          // A la Matriz/Vector, los Predefino antes de Conocer sus Dimensiones:
22          Double[,] DblMatriz;    // Pre-defino la Matriz, antes de conocer sus dimensiones
23          Double[] DblVector;     // Pre-defino el Vector, antes de conocer su dimension
24          Double D_Num;           // Defino el Número que retornará la FunLeerNum()
25          Int16 M, N;             // Los Valores de Filas y Columnas de la Matriz (que uso en varios Módulos)
26          String StrMensaje;      // El Mensaje que paso a la FunLeerNum()
27
28          private Double FunLeerNum(string StrPrompt) // DEFINO la función Leer Número
29          {
30              string StrNum; bool BoolBandera = true; // Defino variables locales para usar en la Funcion
31
32              while (BoolBandera) // Inicio el Ciclo Mientras (Bandera sea verdadera)
33              {  //Llamo a 2 Funciones de MS.VisualBasic: La Fun InputBox() y la IsNumeric(). OJO: Hay que Cargar Referencia.
34                  StrNum = Microsoft.VisualBasic.Interaction.InputBox(StrPrompt, "°° Ingreso de Datos °°", "5");
35
36                  if (Microsoft.VisualBasic.Information.IsNumeric(StrNum))
37                  {
38                      D_Num = Convert.ToDouble(StrNum);   // Si el valor Retornado por InputBox() esNumeric(), lo convierto a Double
39                      BoolBandera = false;    // cambio el valor de la Bandera para salir del Ciclo Mientras
40                  }   // Fin del if
41              }       // Fin Ciclo Mientras
42              return D_Num;   // Retorno el Numero a quien llamó a la función
43          }   // Fin funcion LeerNum
44
```

Definición de Variables Globales

Definición de la Función para Validar la Carga de Elementos para la Matriz Numérica

Código del Botón SALIR

```
44
45      private void BtnSALIR_Click(object sender, EventArgs e)
46      {
47          DialogResult DrOpcion;  // Declaro la Variable del tipo Resultado de la Opción de un MensajeBox
48          DrOpcion = MessageBox.Show("¿Realmente quiere salir de la aplicación?", "Salir del Programa",
49                                     MessageBoxButtons.YesNo, MessageBoxIcon.Question);
50
51          if (DrOpcion == DialogResult.Yes) //Evaluo la Respuesta, Si es SI, muestro mensaje de Agradecimiento
52          {
53              MessageBox.Show("Gracias por usar la aplicación", "Salir del Programa",
54                              MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
55              Close();    // y salgo de la Aplicación
56          }
57      }
58
```

Código del Botón LIMPIAR

```
59      private void BtnLIMPIAR_Click(object sender, EventArgs e)
60      {
61          CbbVector.Items.Clear();     // Limpia el Contenido del ComboBox
62          CbbVector.Text = "Vector Nulo";     // Cambia el Título del ComboBox
63          DgvMatriz.Rows.Clear(); //Limpio las Filas del DGV de la Matriz
64          DgvMatriz.Columns.Clear(); //Limpio las Columnas del DGV de la Matriz
65          DgvVector.Rows.Clear(); //Limpio las Filas del DGV del Vector
66          DgvVector.Columns.Clear();  //Limpio las Columnas del DGV del Vector
67
68          Nud_M.Value = 2; Nud_N.Value = 3;   //Repongo valores iniciales para Dimensionar Matriz
69
70          // Además, controlo el estado de los Botones
71          BtnCargar_MATRIZ.Enabled = true;     // Re-habilita el botón Cargar Matriz
72          BtnCalcular_VECTOR.Enabled = false; // Despues de Limpiar Des-hablita el botón Calcular
73          BtnLIMPIAR.Enabled = false; // y DesHabilita el botón Limpiar
74      }
75
```

```
75
76      private void BtnCargar_MATRIZ_Click(object sender, EventArgs e)
77      {   // Inicio procedimiento Cargar_Matriz
78          M = Convert.ToInt16(Nud_M.Value);   // Obtenemos M (Filas) como n° NATURAL del NumUpDown
79          N = Convert.ToInt16(Nud_N.Value);   // Obtenemos N (Columnas) como n° NATURAL
80          // Ahora que conozco la cantidad de elementos del Vector y Matriz, los Definimos así xq ya estaban PreDefinidos
81          DblMatriz = new Double[M, N];   // Sino, la Defino ASI: Double[,] DblMatriz = new Double[M, N];
82          DblVector = new Double[M];    // Defino el Vector
83          String StrColumna, StrFila; // Defino los contadores de Filas y Columnas como Texto para usar en el DGV
84          for (int c = 0; c < N; c++)   {   // Creo las Columnas necesarias en el DGV
85              StrColumna = Convert.ToString(c);   // Convierto el Contador de Columnas a String
86              DgvMatriz.Columns.Add("", StrColumna);    }      // Agrego las Columnas al DGV
87          // Preguntamos al Usuario por los elementos de la Matriz para cargarla
88          for (int f = 0; f < M; f++) //Ciclo FOR para contar las Filas
89          {
90              DblVector[f] = 0;   // Asigno valor Nulo al 1er. elemento del Vector suma de Columnas
91              StrFila = Convert.ToString(f);  //Convierto Nro. Fila a String
92              DgvMatriz.Rows.Add();  //Agrego Fila a la DGV Matriz
93              for (int c = 0; c < N; c++) //Ciclo FOR para contar las Columnas
94              {
95                  StrColumna = Convert.ToString(c);   //Convierto Nro. Columna a String
96                  // Llamo a la función Leer Num para pedir el ingreso de un Número para M[f, c], al Usuario.
97                  StrMensaje = "Ingrese el Elemento M( " + StrFila + " , " + StrColumna + " )";  //Genero el Mensaje
98                  D_Num = FunLeerNum(StrMensaje); // Llamo a la función que me trae el Número de tipo double
99                  DblMatriz[f, c] = D_Num;    // Asigno el Numero Retornado por la Función al elemento corresondiente de la Matriz
100                 // Mostrar las Notas en el DataGridView
101                 DgvMatriz.Rows[f].Cells[c].Value = String.Format("{0:#0.0}", DblMatriz[f, c]);
102                 DblVector[f] = DblVector[f] + DblMatriz[f, c];  // Calculo la suma de columnas de c/Fila y la asigno al vector
103             }   // Fin del ciclo por Columnas
104         }       // Fin del Ciclo por Filas
105         // Además, controlo el estado de los Botones al Terminar el Procedimiento con la Matriz Cargada
106         BtnCargar_MATRIZ.Enabled = false;    // Des-habilita el botón Cargar Matriz
107         BtnCalcular_VECTOR.Enabled = true;  // Habilita el botón Calcular
108         BtnLIMPIAR.Enabled = true;  // Habilita el botón Limpiar
109     }
```

Punto de Interrupción para Depurar el Programa, colocado en la línea 98

```
110
111     private void BtnCalcular_VECTOR_Click(object sender, EventArgs e) // Solo Muestra el Vector, xq ya lo Calculó en Linea 102
112     {
113         String S_Fila, S_Columna;   // Defino las variables internas del Módulo
114                   //    OJO TODAS ESTA LÍNEAS SON  S O L O  PARA MODIFICAR EL ASPECTO DEL DGV. Si no van NO pasa NADA.
115          // Para cambiar el formato de la Cabecera del DGV, 1ro. el FALSE en HeadersVisualStyle
116          DgvVector.EnableHeadersVisualStyles = false;     // 1ro. el FALSE, y Ahora SI, cambiar las propiedades:
117          DgvVector.ColumnHeadersDefaultCellStyle.BackColor = Color.LightSteelBlue;  //Cambia Color de Cabecera de Columnas
118          DgvVector.ColumnHeadersDefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter; //Cambia Alineacion de Texto
119          DgvVector.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight; // Alineamiento del resto de las Celdas.
120          DgvVector.DefaultCellStyle.Font = new Font("Microsoft Sans Serif", 11);    // Establece tipo de Font y Tamaño
121          DgvVector.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.DisplayedCells; //Columnas se adapten al ancho de las Celdas
122          CbbVector.Items.Clear();    // Limpia el Contenido del ComboBox
123          CbbVector.Items.Add("V[i]".PadRight(10) + "Valor".PadRight(18)); // Mostramos Los Títulos de las Columnas en el ComboBox
124
125         for (int c = 0; c < 1; c++) {
126             S_Columna = Convert.ToString(c);   // Convierto el Contador de Columnas a String
127             DgvVector.Columns.Add("", S_Columna); }  // Agrego las Columnas al DGV
128         // Preguntamos al Usuario por los elementos de la Matriz para cargarla
129         for (int f = 0; f < M; f++)     {
130             Int16 N_c = 0;  // Establezco la Columna donde mostraré los Datos del Vector Columna
131             S_Fila = Convert.ToString(f);   // Convierto a String el Nro de Fila
132             DgvVector.Rows.Add();  // Agrego una Fila
133
134             // Mostrando los Valores de la Suma en el DataGridView
135             DgvVector.Rows[f].Cells[N_c].Value = String.Format("{0:#0.0}", DblVector[f]);
136             // Mostrando los Valores de la Suma en el ComboBox Agregando (Add) Items.
137             CbbVector.Items.Add(S_Fila.PadRight(10) + String.Format("{0:#0.0}", DblVector[f]));
138         }
139         CbbVector.Text = "Ver Vector";    // Cambio el Título del ComboBox
140         BtnCalcular_VECTOR.Enabled = false;  // Habilita el botón Calcular
141     }   //Fin BtnCalcular_VECTOR
142 }       // FIN public partial class Form1
143 }       // FIN namespace TP9_G12_Ej12
144
```

100 %

Fin del Documento, lo que resta es experimentar con el programa.