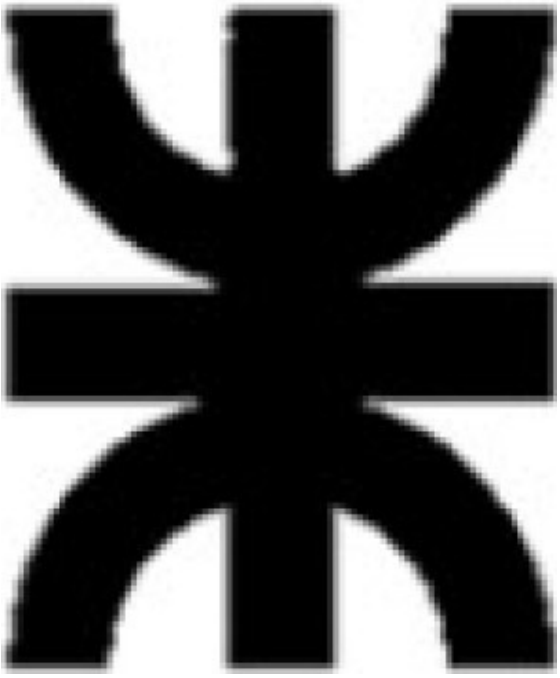


UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional Reconquista

Programación en Computación

Ciclo Lectivo: 2020

Trabajo Práctico N.º 8

8 – VECTORES

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

Guía Unidad 8: VECTORES

SITUACIONES PROBLEMÁTICAS:

1) Generar un vector de 10 elementos X(i). Realizar un programa que imprima la suma y el promedio de los cinco primeros elementos y de los cinco últimos elementos.
2) Almacenar N números en un vector, imprimir cuantos son ceros, cuantos negativos, cuantos positivos. Imprimir además la suma de los negativos y de los positivos.
3) Llenar un vector de 20 elementos. Imprimir el promedio general, el promedio de los 10 primeros elementos y el promedio de los 10 últimos.
4) Calcular el promedio de N valores almacenados en un vector. Imprimir el promedio, el número de datos mayores que el promedio y una lista de valores mayores que el promedio. Determinar además cuantos son menores que el promedio.
5) Llenar un vector de N elementos. Imprimir el porcentaje de nulos y el promedio de impares.
6) Generar un vector de A(i) de N elementos numéricos. Ingresando un valor K, calcular el promedio de todos los elementos de A(i) que sean menores que K. imprimir el resultado.
7) Llenar un vector de N elementos, imprimir la posición y el valor del elemento mayor almacenado en el vector. Suponga que todos los elementos del vector son diferentes.
8) Se tienen almacenados en la memoria dos vectores M(i) y N(i) de N elementos cada uno. Hacer un algoritmo que escriba la palabra “Iguales” si ambos vectores son iguales y “Diferentes” si no lo son. Serán iguales cuando en la misma posición de ambos vectores se tenga el mismo valor para todos los elementos.
9) Se tiene el vector A(i) con N elementos almacenados. Diseñe un algoritmo que escriba “SI” si el vector esta ordenado de forma ascendente o “NO” si el vector no está ordenado.
10) Diseñe un algoritmo que lea un numero cualquiera y lo busque en el vector X(i), el cual tiene almacenados N elementos. Escribir la posición donde se encuentra almacenado el número en el vector o el mensaje “NO” si no lo encuentra. Búsqueda secuencial.
11) Almacenar N números en un vector, almacenarlos en otro vector en orden inverso al vector original e imprimir el vector resultante.
12) Crear un algoritmo que reciba como entrada un valor N y que arroje como salida un Vector con los valores: 0!, 1!, 2!, 3!, 4!, ….., N!

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. Uso de condicionales para la formulación de algoritmos. U 7) Estructuras Repetitivas o Cíclicas: HACER-PARA, HACER-MIENTRAS, REPETIR-HASTA. Resolución de problemas usando todos los tipos de estructuras para la formulación de sus algoritmos. U_8) Vectores. Utilización del tipo de dato vector. Índice de un vector. Mayor y menor de un vector. Promedio de un vector. Porcentajes en un vector. Algoritmos con más de un vector.

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:

Crear un algoritmo que reciba como entrada un valor N y que arroje como salida un Vector con los valores: 0!, 1!, 2!, 3!, 4!, ….., N!.

1) Análisis del Problema:

El problema nos pide que, dado un número obtener un Vector que contenga los factoriales desde 0! Y hasta el número dado N !

Como DATOS necesitamos:
Numero_N, como número de partida, y del cual se pueda obtener el Factorial, o sea, un número Natural.
Calculador de Factorial de un Número:

¿Qué es factorial de un número?

Definición de factorial

El factorial de un número n es el producto de todos los naturales menores o igual a n. El factorial de 0 es, por definición, igual a 1. Para enteros negativos, los factores no se definen. El factorial puede ser visto como el resultado de la multiplicación de una secuencia de números naturales descendentes (como 3 × 2 × 1). El símbolo del factorial es el signo de exclamación "!".

La fórmula Factorial

Si n es un número natural mayor o igual a 1, entonces
n! = n x (n - 1) x (n - 2) x (n - 3) ... 3 x 2 x 1

En nuestro problema, por facilidad de cálculo invertiremos la fórmula y comenzaremos con 0
n! = 1 * 1 * 2 * 3 … (n-3) * (n-2) * (n-1) * n

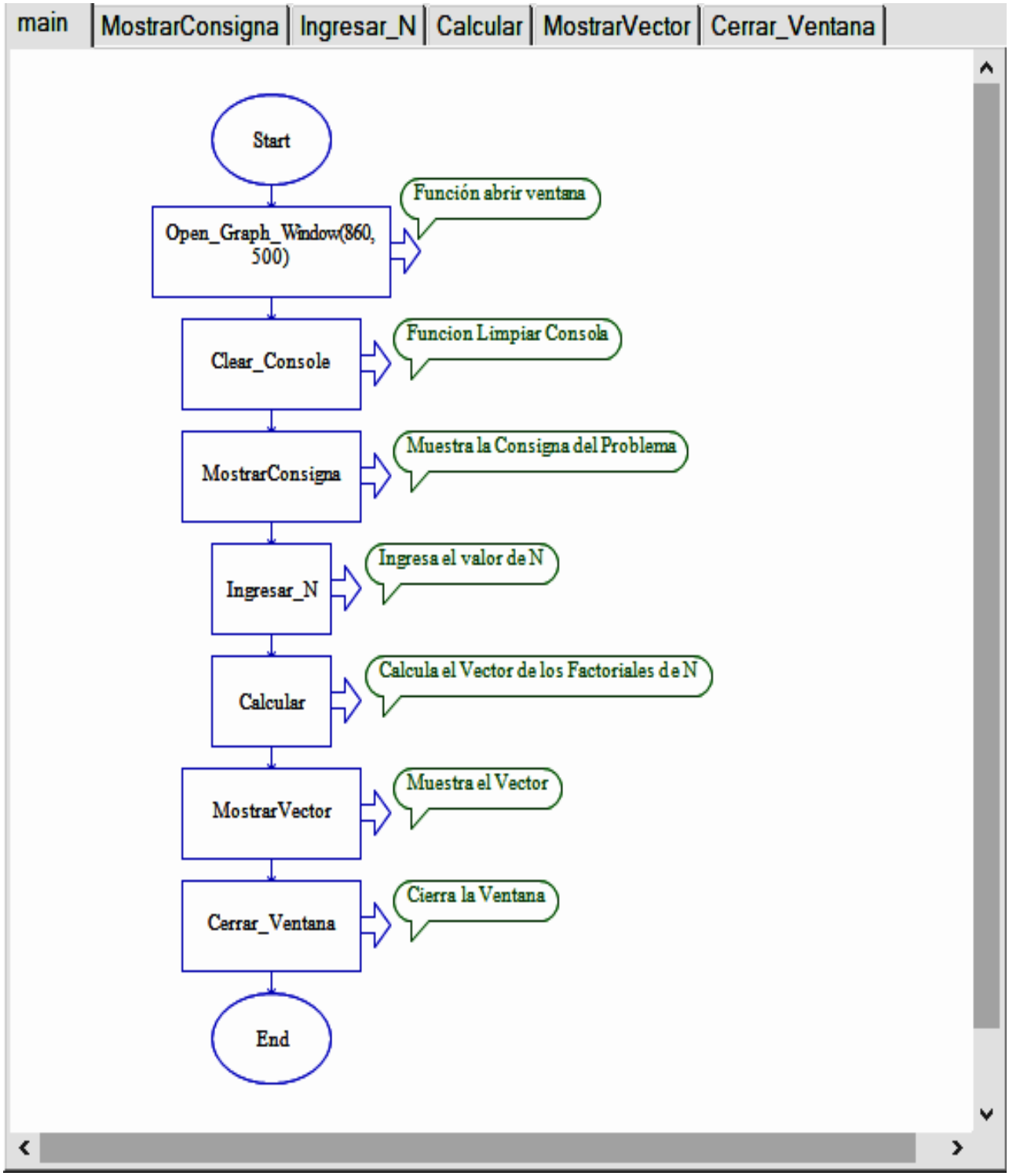
n	n!
0	1 (por definición)
1	0! * 1 = 1 * 1 = 1
2	1! * 2 = 1 * 2 = 2
3	2! * 3 = 2 * 3 = 6
4	3! * 4 = 6 + 4 = 24
5	4! * 5 = 24 * 5 = 120
6	5! * 6 = 120 * 6 = 720
7	6! * 7 = 720 * 7 = 5040
8	7! * 8 = 5040 * 8 = 40320
9	8! * 9 = 40320 * 9 = 362880
10	9! * 10 = 362880 * 10 = 3628800

……

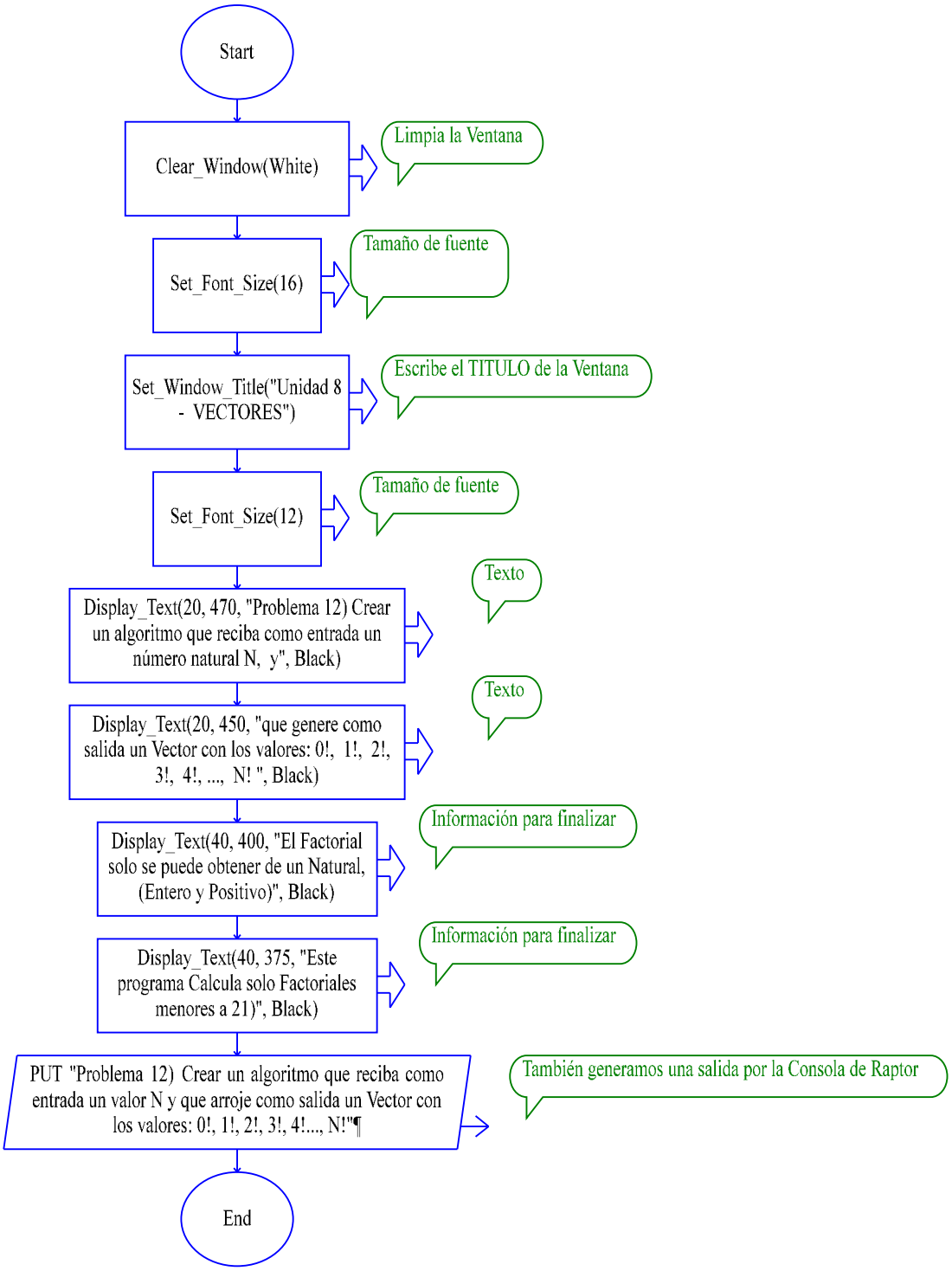
VectorFactorial[ ], formado por todos los factoriales hasta el factorial de n, o sea, será un vector de n+1 elementos.

2) Diagramas de Flujo en RAPTOR.

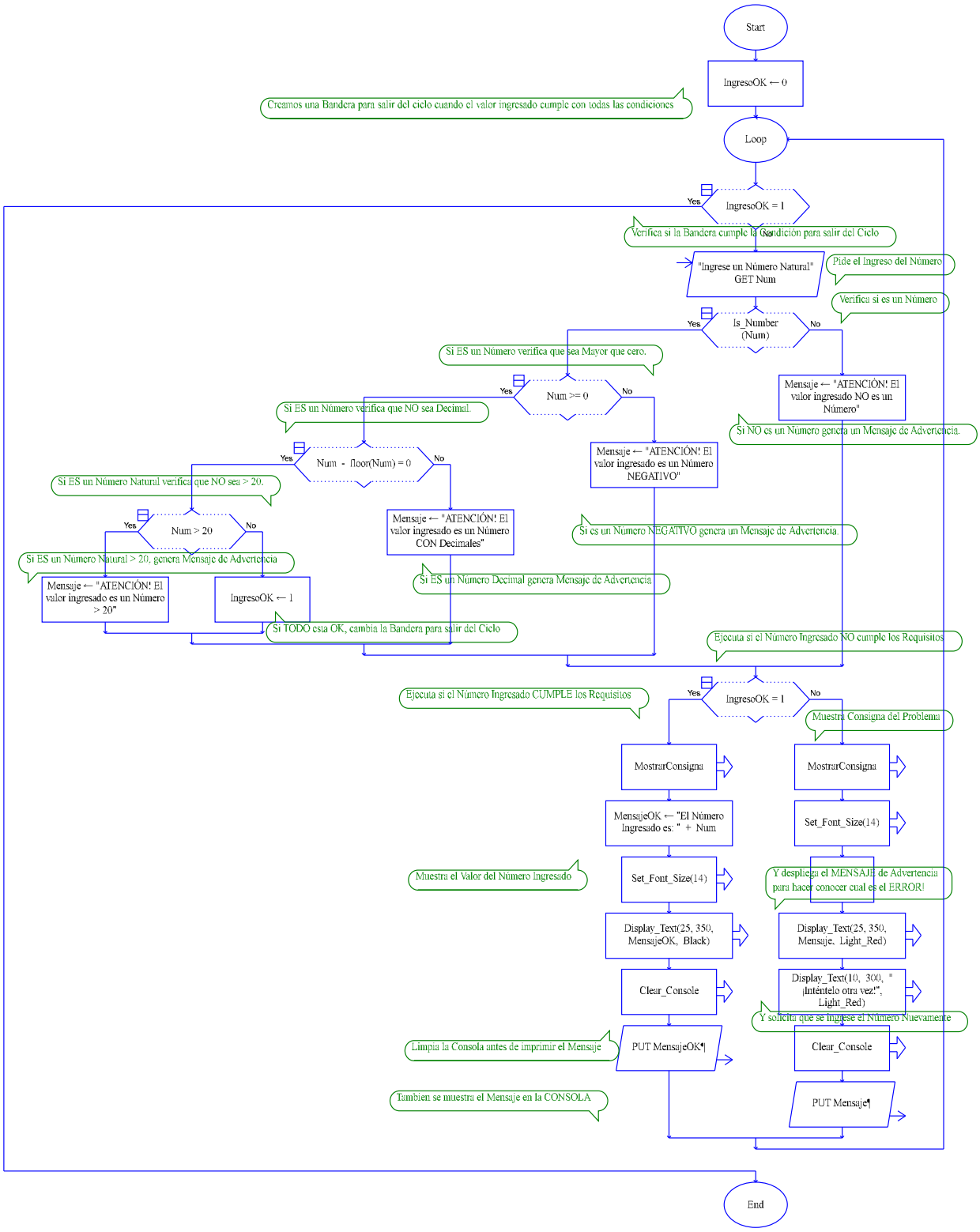
Programa **“main”**



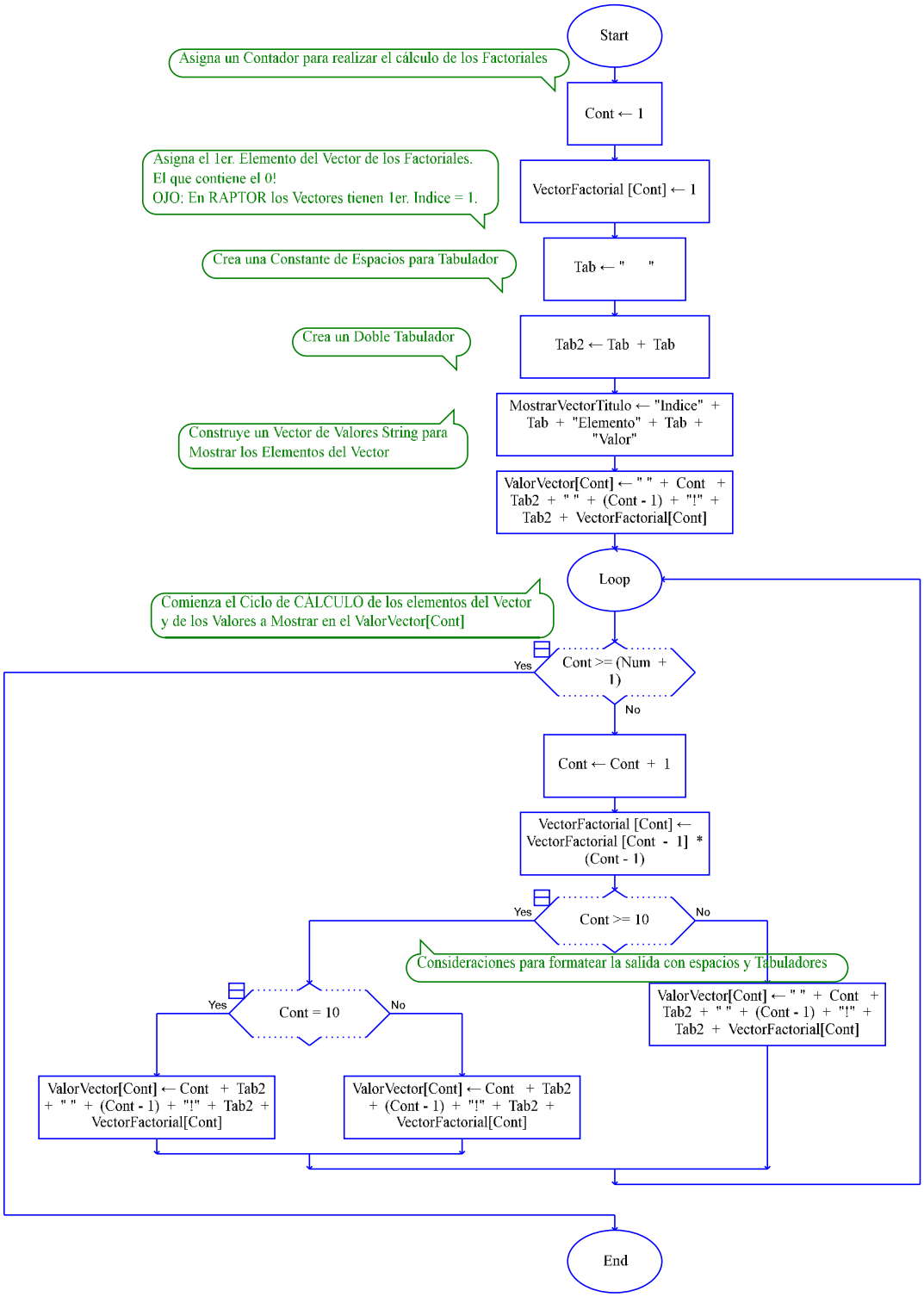
Subrutina **“MostrarConsigna”**



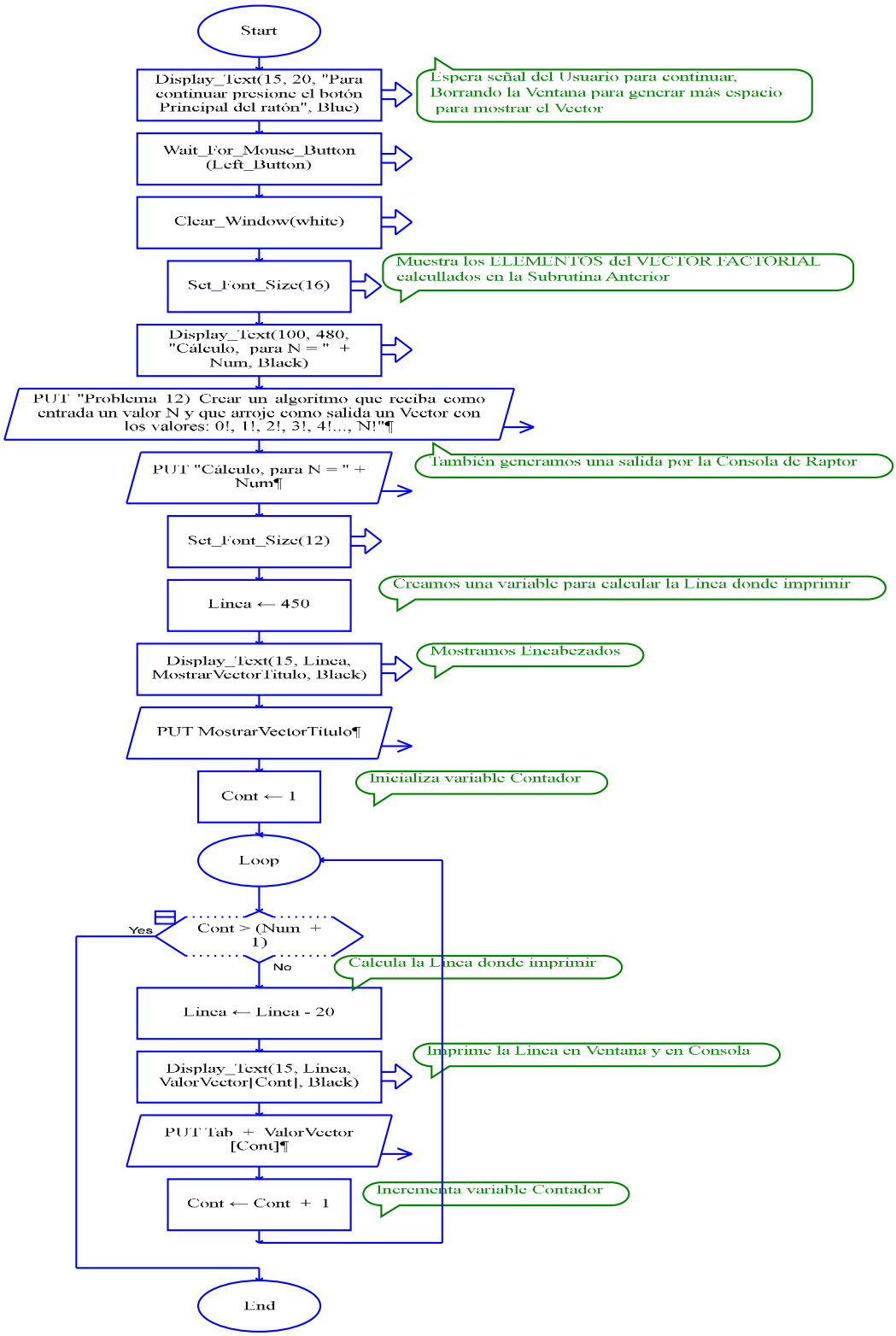
Subrutina **“Ingresar_N”**



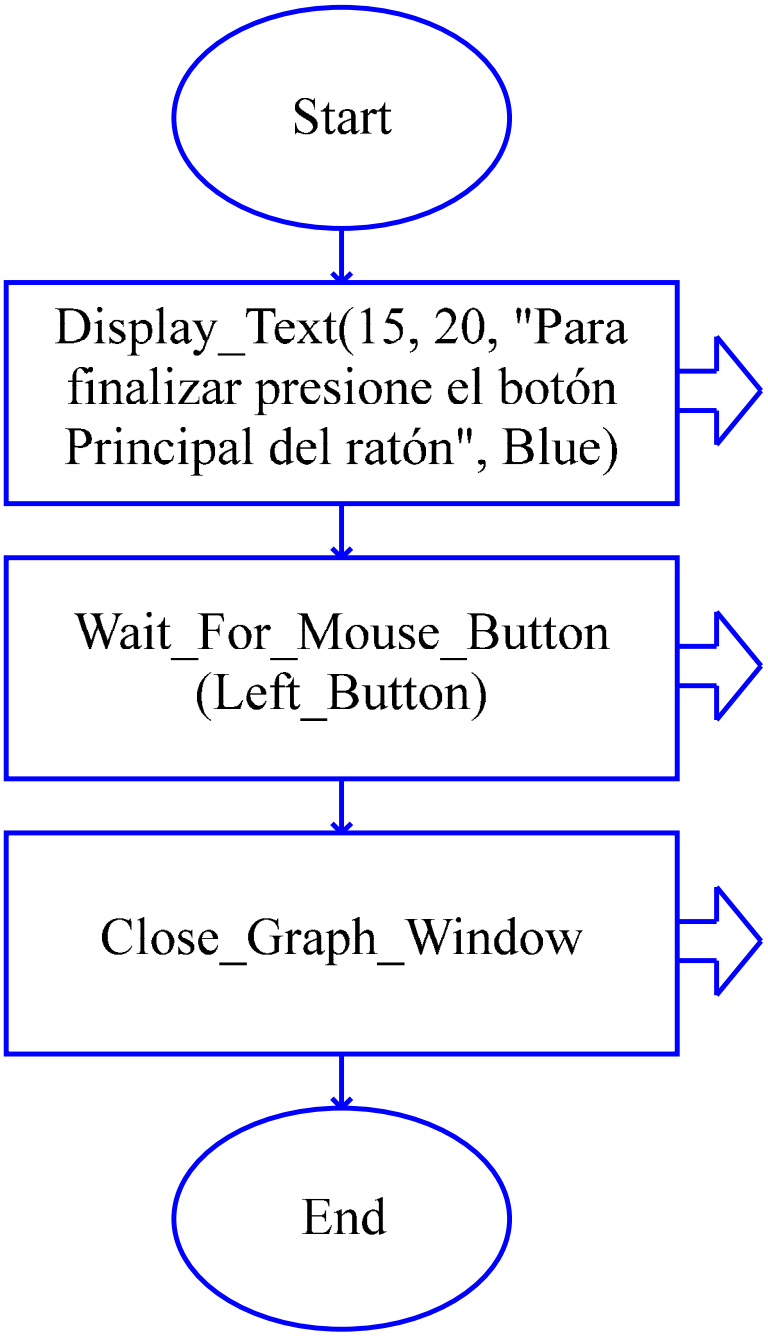
Subrutina **“Calcular”**



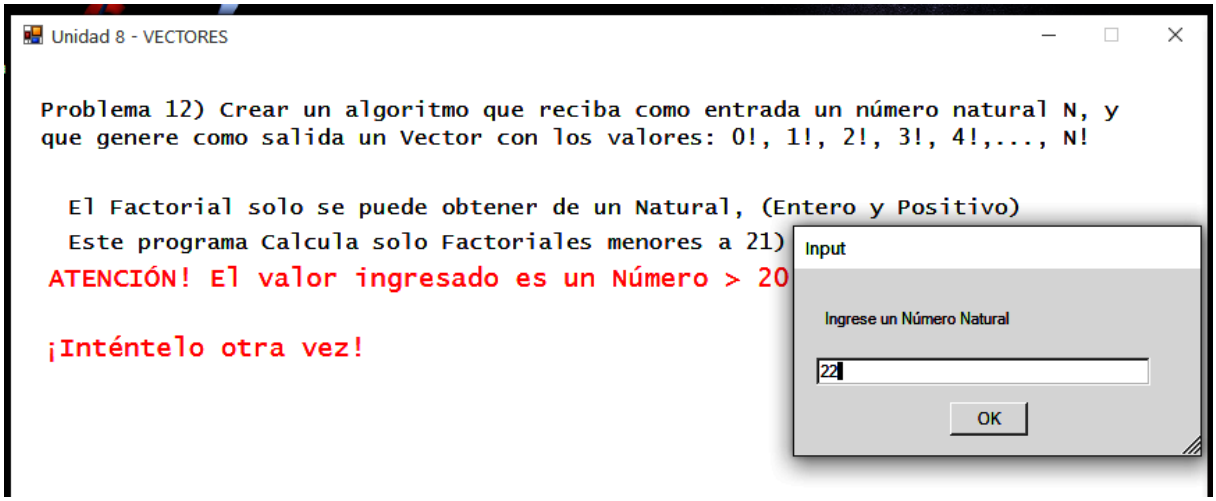
Subrutina **“MostrarVector”**

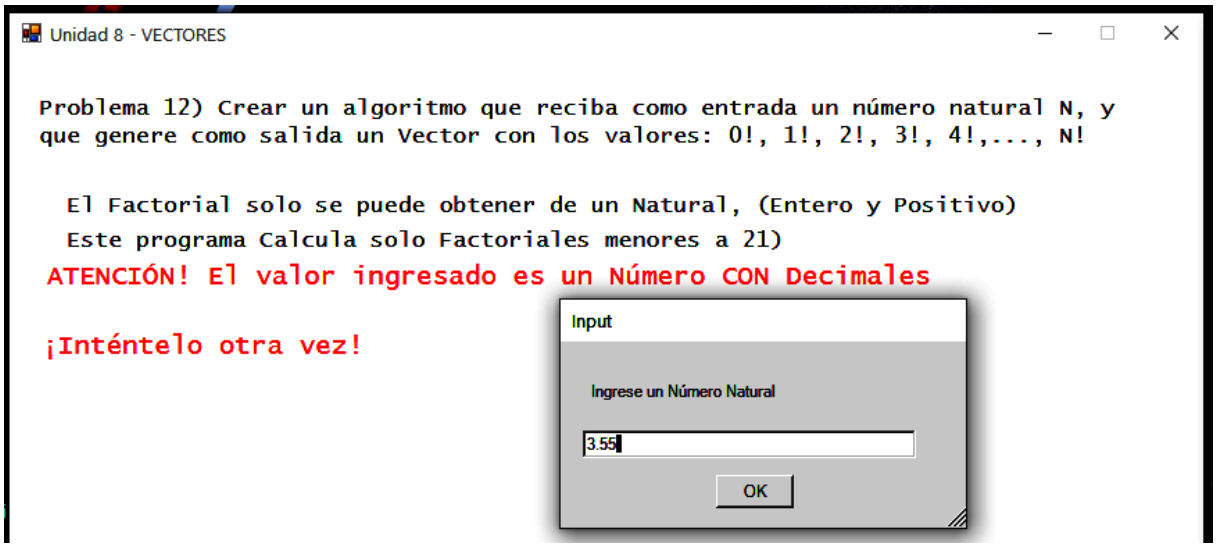


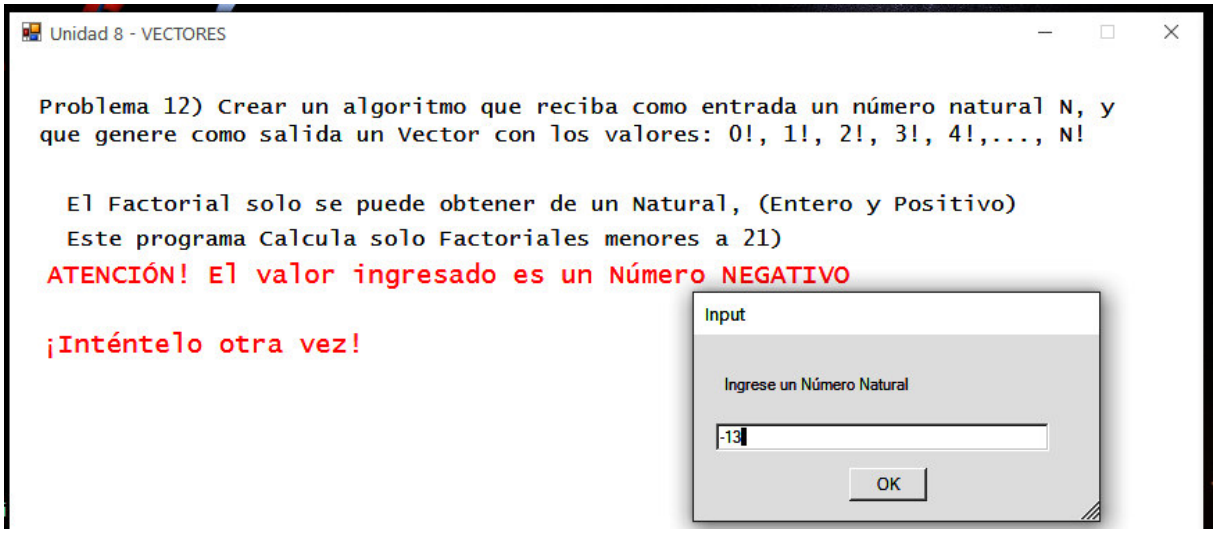
Subrutina **“Cerrar_Ventana”**

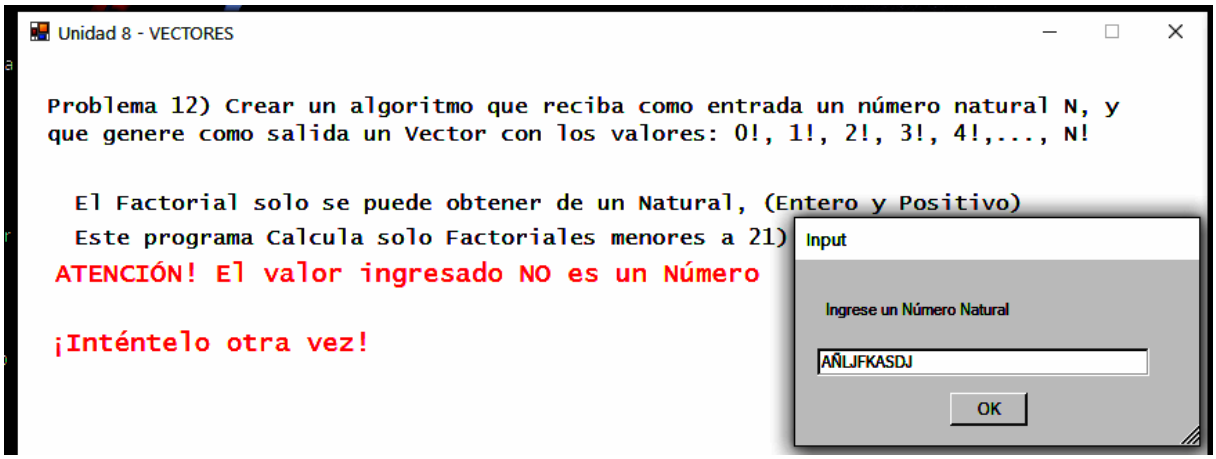


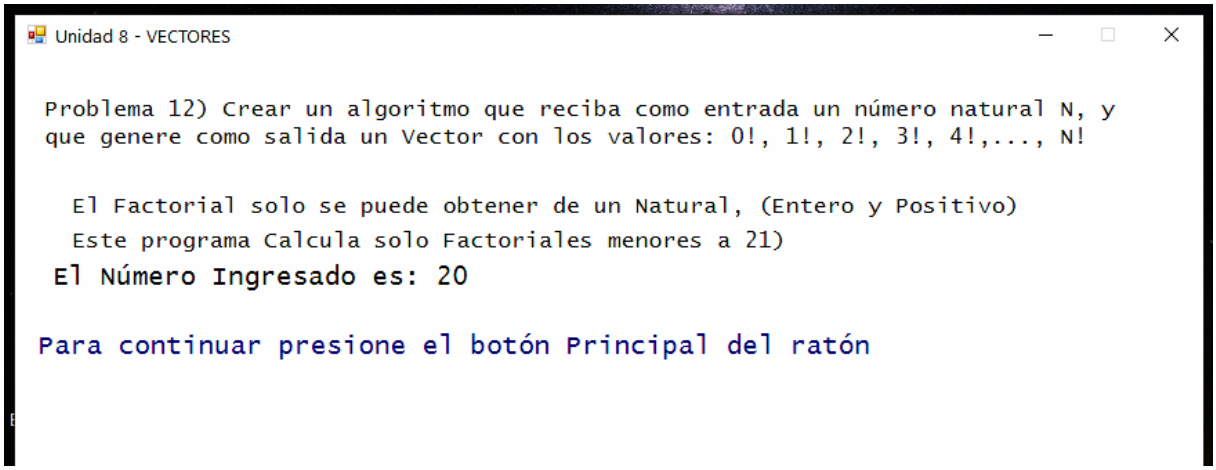
Al DF en RAPTOR, lo ejecutamos y nos muestra sus Salidas en Ventana



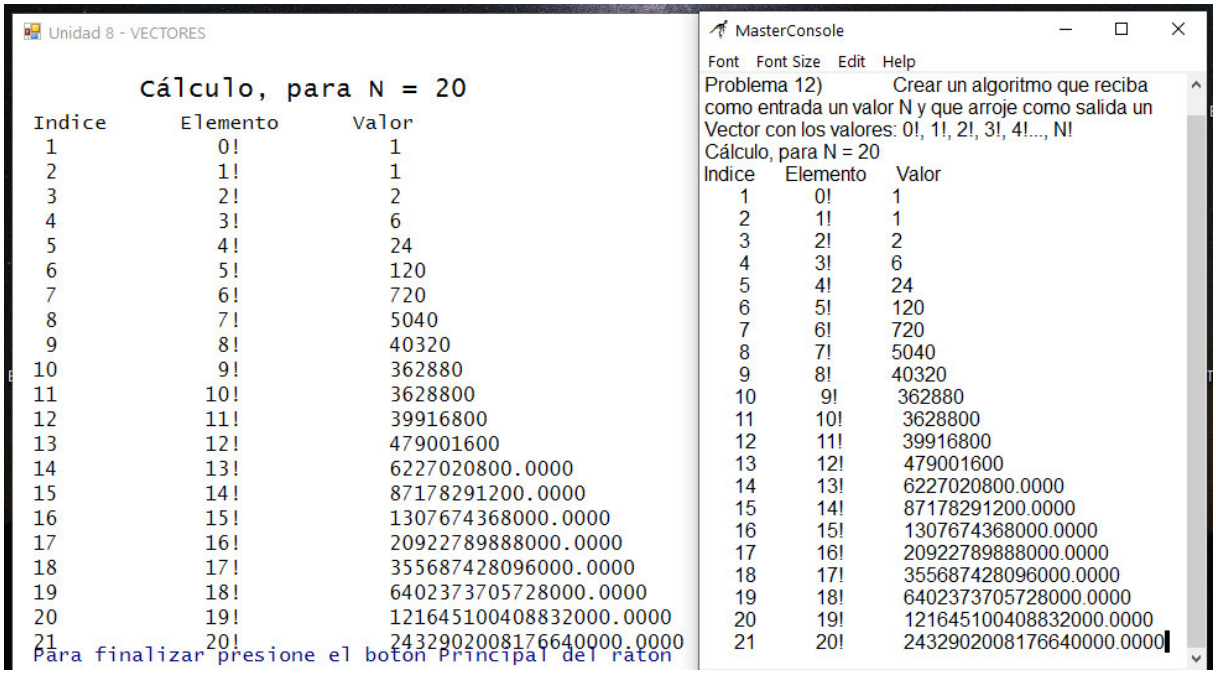








Salidas en Ventana Y en CONSOLA



3) Codificación en C# (Windows From).

A partir de ahora, la codificación en C# Consola NO se realizará más.

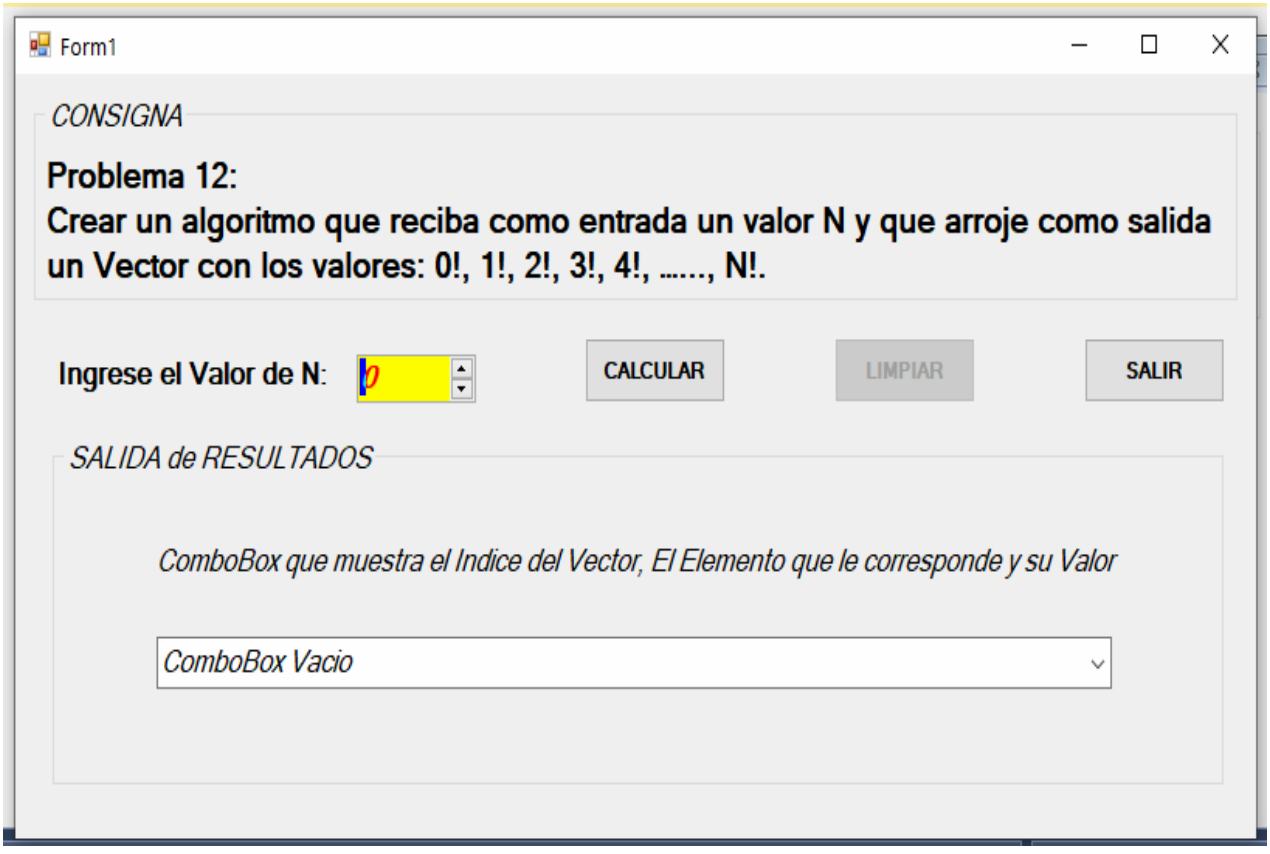
Tampoco usaremos el Código C# generado por Raptor; realizaremos el programa en C# Ventana como una aplicación nueva y totalmente independiente.

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

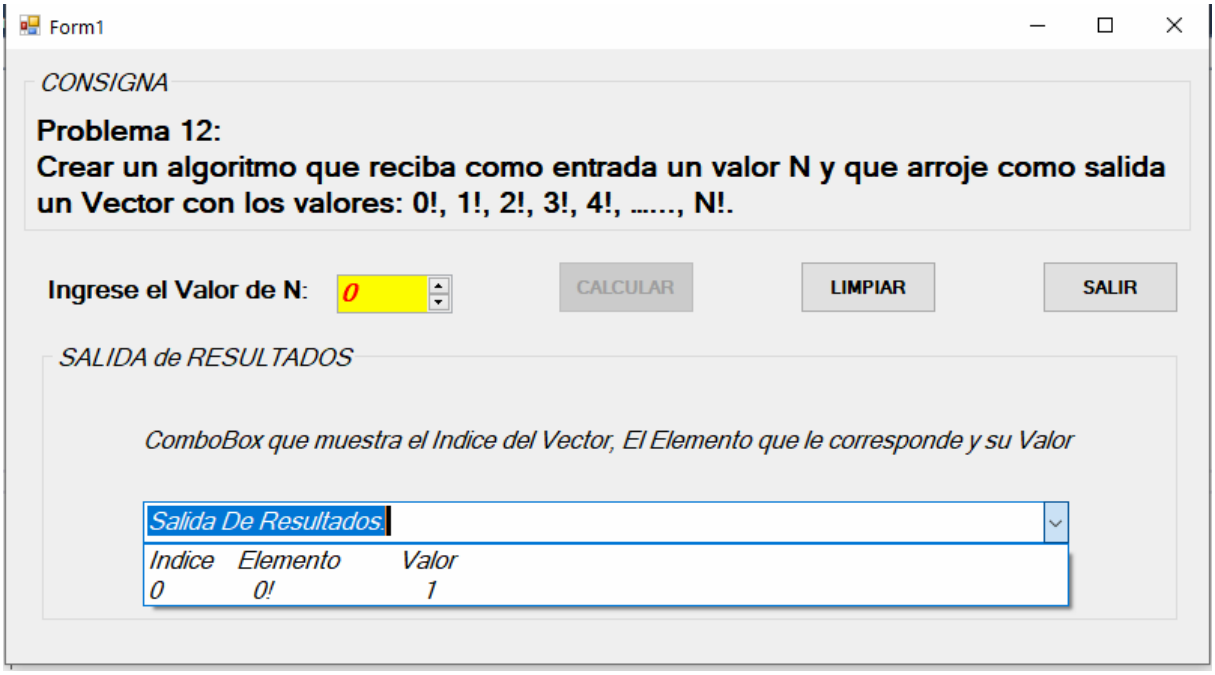
Esta GUI es muy simple, comienza con un GroupBox que contiene una Label con la Consigna del Problema que resuelve. Continua con un Área para el ingreso de Datos y los Botones de Procesos más habituales: Calcular, Limpiar y Salir. Finaliza con otro GroupBox que contiene una Label que nos indica lo que mostrará el ComboBox.

Para el ingreso del Dato usamos un NumericUpDown que tiene seteado por defecto el valor 0, y además, No permite decimales, valores negativos ni Positivos mayores a 20 (Investigar a que se debe!).

El Texto que nos muestra el ComboBox al inicio, indica que el mismo esta Vacío, y en consecuencia, el botón Limpiar NO está habilitado.

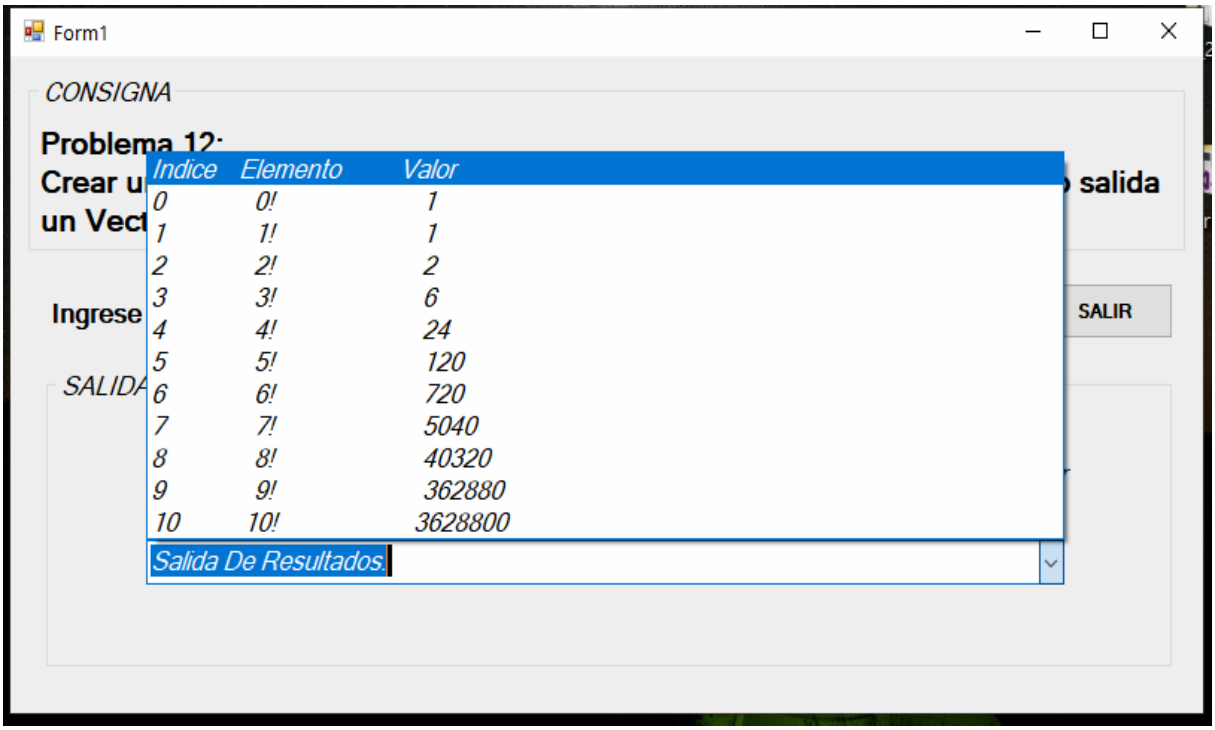


Si lo hacemos clic en Calcular, el programa ejecuta el Algoritmo creado y muestra los valores del Vector en el ComboBox y nos lo indica en el Título de este, y además podemos ver que deshabilita el botón Calcular y habilita el botón Limpiar:

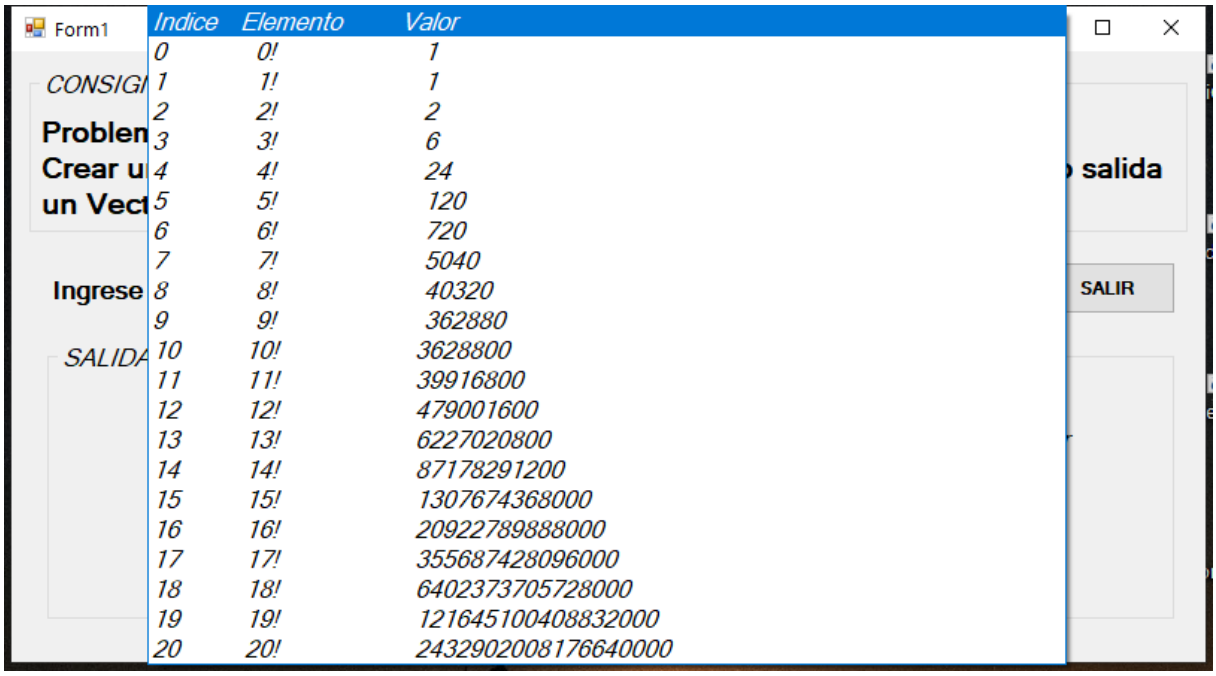


Si limpiamos para realizar otro Calculo, obtendremos una ventana igual a la anterior.

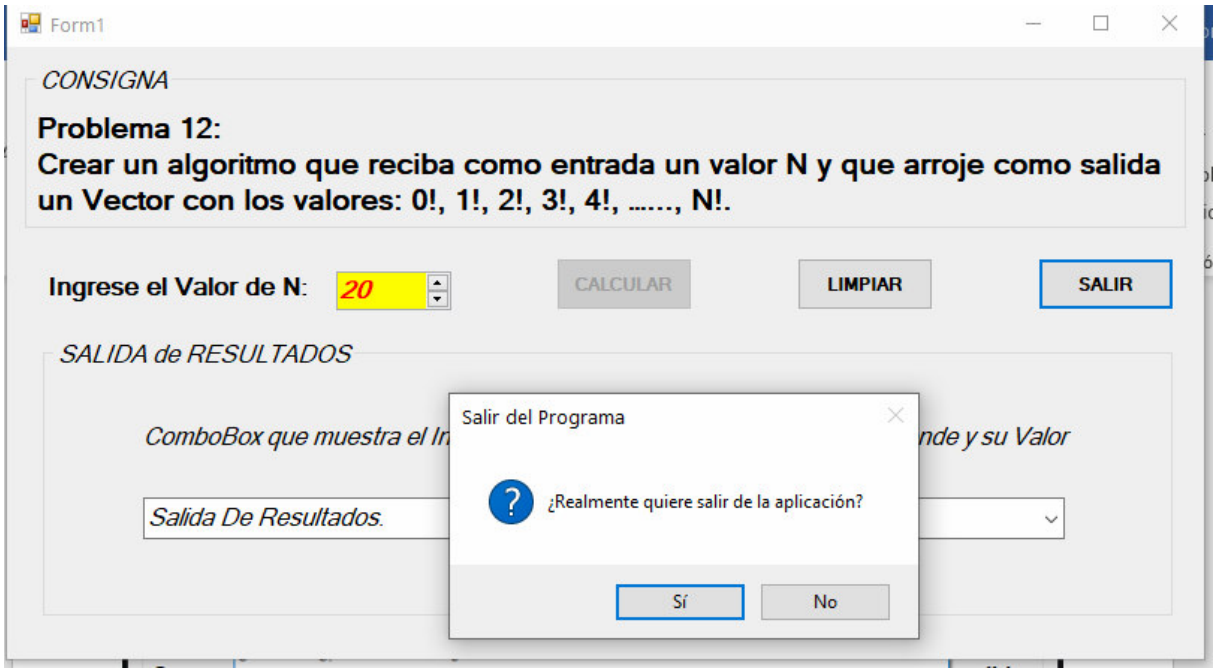
Si ahora cambiamos el valor de N a 10, y expandimos el ComboBox, veremos:



Si limpiamos y cambiamos el valor de N a 20, y expandimos el ComboBox, veremos:

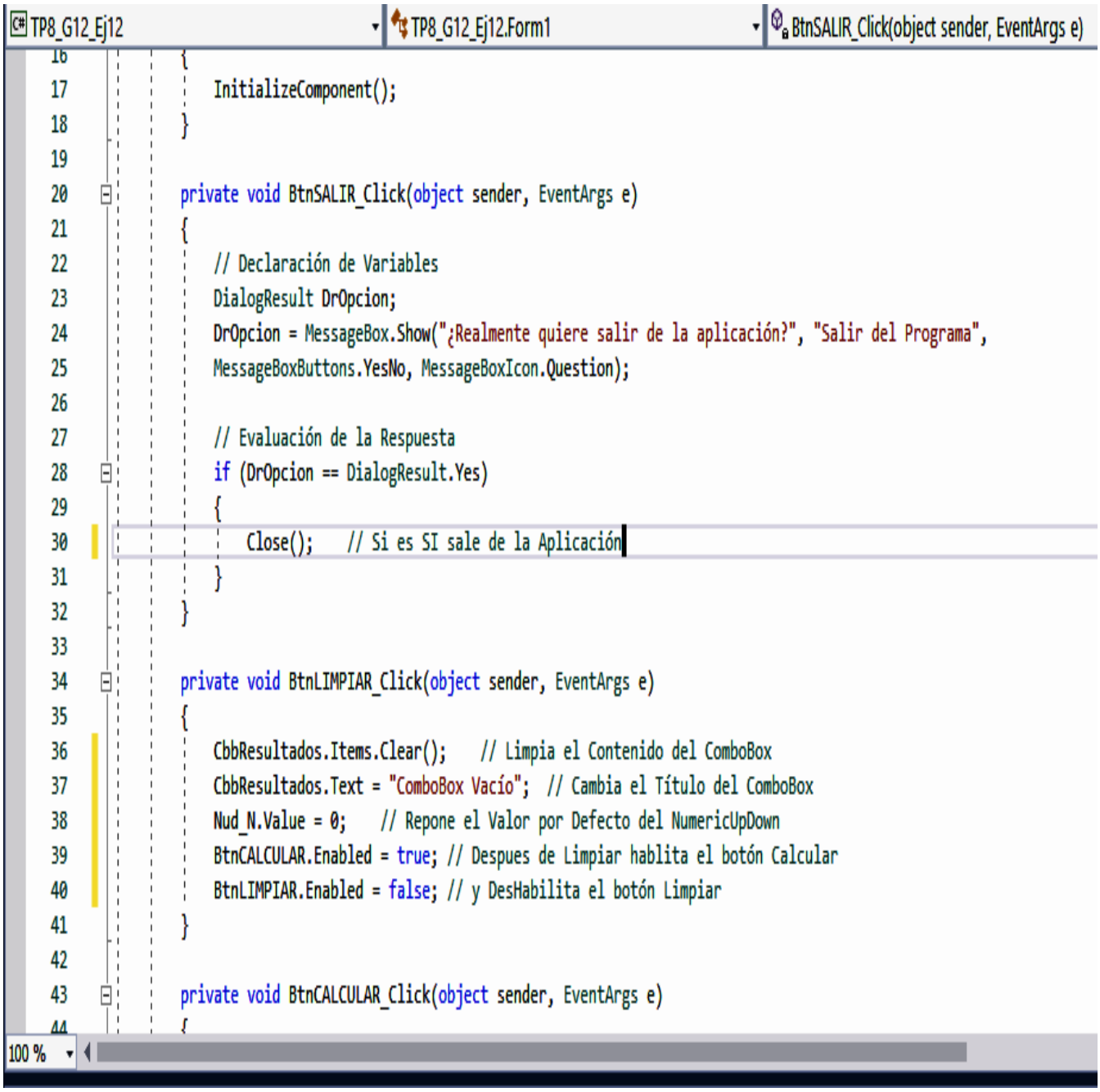


Como corresponde, el botón salir esta siempre habilitado:



Análisis del Código.

Sobre el código de Salir, No hay nada nuevo, ya lo hemos visto y analizado en otra guía.



Sobre el código de Limpiar, la novedad está dada por la limpieza del contenido del ComboBox y el cambio de su título (líneas 36 y 37 del código)

Código del botón Calcular:

```
42
43      private void BtnCALCULAR_Click(object sender, EventArgs e)
44      {
45          CbbResultados.Items.Clear();    // Limpiamos el ComboBox
46          Int16 N = Convert.ToInt16(Nud_N.Value);   // Obtenemos N como n° NATURAL del NumericUpDown
47
48          // Definimos el Vector, indicando el TIPO de Dato, El NOMBRE y la CANTIDAD de Elementos
49          Int64[] Vector_Factorial = new Int64[N + 1];    // Tendrá un elemento + por el 0 inicial
50
51          // Mostramos Los Títulos de las Columnas en el ComboBox Agregando (Add) Items.
52          // CbbResultados.Items.Add("Indice \t Elemento \t Valor"); //Probe != opciones, pero
53          CbbResultados.Items.Add("Indice".PadRight(10)+ "Elemento".PadRight(18) + "Valor");
54          // Terminé usando la Función PadRight() para alinear elementos en el Combo. Investigarla!
55
56          Vector_Factorial[0] = 1;    // Predefinimos el Factorial de 0, 0! = 1
57          // Agregamos el primer elemento del Vector en el ComboBox
58          CbbResultados.Items.Add("0".PadRight(16) + "0!".PadRight(27) + Vector_Factorial[0]);
59
60          // Recorremos el Vector, iniciando en 1, xq' 0! tiene valor por Definición, NO por cálculo.
61          for (int i = 1; i <= N; i++)
62          {   // Calculamos el Valor del Vector para [i]
63              Vector_Factorial[i] = (Vector_Factorial[i-1] * i);
64
65              // Mostrar en el ComboBox el i, i!, y el Valor Calculado para cada iteración del ciclo FOR
66              if (i < 10) // Esta división es para alinear las columnas de los textos.
67              {
68                  CbbResultados.Items.Add( i +"".PadRight(15) + i + "!".PadRight(26) + Vector_Factorial[i]);
69              }
70              else
71              {
72                  CbbResultados.Items.Add(i + "".PadRight(11) + i + "!".PadRight(24) + Vector_Factorial[i]);
73              }
74          }
75
76          // Imprimimos Título en el ComboBox para informar del fin del proceso de CALCULO
77          CbbResultados.Text = "Salida De Resultados.";
78          BtnCALCULAR.Enabled = false;
79          BtnLIMPIAR.Enabled = true;
80      }
81  }
82  }
83
```

100 %

Para entenderlo, seguir las indicaciones de las líneas Verdes, pero hay dos puntos importantes a destacar:

1) La Definición de un Vector, que se define estableciendo el tipo de Datos que va a contener, con la indicación del Nombre del Vector y con un Numero Entero Positivo para indicar la cantidad de elementos:

```
Int16 N = Convert.ToInt16(Nud_N.Value);   // Obtenemos N como nº NATURAL del NumericUpDown
Int64[] Vector_Factorial = new Int64[N + 1];    // Tendrá un elemento + por el 0 inicial
```

2) Como agregar texto al ComboBox con .Items.Add(), que, como el TextBox, reconoce Texto:

```
CbbResultados.Items.Add("Indice".PadRight(10)+ "Elemento".PadRight(18) + "Valor");   // o
CbbResultados.Items.Add( i +"".PadRight(15) + i + "!".PadRight(26) + Vector_Factorial[i]);
```

El Método String .PadRight/PadLeft (Int32, Char), agrega Char, y si este NO se especifica agrega espacios en blanco

Al terminar el proceso, Des-Habilitamos el botón Calcular, Habilitamos el botón Limpiar y cambiamos el Título del ComboBox para indicar que ya se han cargado los valores.

Enviar el Archivo ***TP8_Gx_Ejx.ZIP*** correspondiente al *Grupo y Ejercicio Resuelto,* que debe contener:

A) El informe en PDF (similar a este que he realizado para el Ej.12), y que debe contener: 1- El Análisis del Problema, 2- El Diagrama de Flujo en Raptor y 3- La GUI y el Código del Formulario de VSC#.
B) El Archivo referenciado en el punto 2 anterior. (Solo él **.rap**, NO los **.rap.backupX)**
C) La carpeta con el proyecto generado en el punto 3.