

UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional Reconquista

Programación en Computación
Ciclo Lectivo: 2020

Trabajo Práctico N.º 4

Guia_U4_Técnicas_de_formulación_de_Algoritmos

GRUPO N.º: 12

INTEGRANTES: Fulano, Mengano, Zutano

SITUACIONES PROBLEMÁTICAS:

1. Dados tres números, deducir cuál es el central.
2. Escribir un algoritmo que lea un número y deduzca si está entre 10 y 100, ambos inclusive.
3. Se desea realizar una estadística de los pesos de los alumnos de un colegio de acuerdo con la siguiente tabla:

Alumnos < 40kg
40kg <= Alumnos <= 50kg
50kg < Alumnos < 60kg
Alumnos >= 60kg

La entrada de los pesos de los alumnos se terminará cuando se introduzca el valor centinela -99. Al final se desea obtener cuántos alumnos hay en cada uno de los baremos.

4. Realizar un algoritmo que averigüe si dados dos números introducidos por teclado, uno es divisor del otro.
5. Realizar un algoritmo que pida el ingreso de un Nro. por teclado e informe si se encuentra comprendido en el entorno [-10; 10].
6. Realizar estadística de las alturas de alumnos s/ siguiente tabla:

Alumnos < 1.5 m
1.5 <= Alumnos <= 1.7 m
1.7 < Alumnos < 2 m
Alumnos >= 2 m

Al final se desea obtener cuántos alumnos hay en cada uno de los baremos.

7. Realizar estadística de las edades de alumnos s/ siguiente tabla:

Alumnos < 18
18 <= Alumnos <= 20
20 < Alumnos < 25
Alumnos >= 25

Al final se desea obtener cuántos alumnos hay en cada uno de los baremos.

8. Dados tres números, deducir si son distintos, y en ese caso indicar cual es el menor, el mayor y el central.

9. Ingresar un número e indicar si es mayor, menor o igual a 0.

10. Ingresar 10 número y determine cuales son primos.

12. Realizar un programa que resuelva la ecuación cuadrática, pero que valide si los valores ingresados como coeficientes son números

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. Realizar el **pseudocódigo**,
3. **Diagrama de Flujo** en RAPTOR, usando Módulos
4. Codificación en **VSC# Consola** usando Módulos
5. Codificación en **VSC# Ventana**. (Windows Form)

Conocimientos necesarios:

U 2) Algoritmos. Fases en la Creación de Programas. [Programa = Algoritmos + Estructuras de DATOS]. U 3) Tipos de Datos. Expresiones. Operadores y Operandos. Identificadores de Constantes y Variables. Instrucciones Básicas: (1- Asignación, 2- Entrada, 3- Salida, 4-Expresiones Aritméticas y 5- Expresiones Lógicas. 6- Palabras Reservadas).

U 4) Diagramación lógica. Diagramas de Flujo. Simbología. Algoritmos en pseudo - código. Diagramas estructurados (Nassi-Schneiderman)

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.

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 Capítulos 5.
Curso de Algoritmia.pdf,	En especial Capítulos 5.

Operatoria:

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

Resumen de estos:

El ***Diseño del Algoritmo***, [*Secuencia ordenada de pasos - Sin Ambigüedades – que conducen a la solución de un Problema*], que debe ser:
i - Preciso ii – Definido iii – Finito.

Esto implica identificar las tareas más importantes y ponerlas en el orden en que deben ejecutarse.

Estos pasos pueden repetirse (**Refinamiento Progresivo, Diseño Descendente o Top-Down**) hasta obtener un **Algoritmo**:

Claro, Preciso y Completo.

Además, un Algoritmo consta de tres partes principales:

- i - **Entrada** o información necesaria para la ejecución del **Algoritmo**.
- ii - **Proceso** u operaciones necesarias **para la Solución**.
- iii - **Salida** o respuestas dadas por el **Algoritmo**.

Un algoritmo puede ser escrito en Castellano narrativo, pero esta descripción suele ser demasiado ambigua. Para representarlo hay que buscar un método que consiga que sea fácilmente codificable y además que sea independiente de los lenguajes de programación.

Los métodos más usuales son:

A) Pseudo-Código

B) Diagramas de Flujo

C) Diagrama Nassi – Schneiderman, muy recomendado pero poco usado.

NOTA: Los errores más comunes en programación son:

1. **De Sintaxis (mucho más comunes)**, Errores en las palabras o identificadores permitidos o en los signos de puntuación. Por suerte si usamos el IDE del Lenguaje, este nos ayudará a resolverlos.
2. **De Lógica**, Se pensó mal la forma de resolver el problema. OJO, muchas veces es un Error muy difícil de detectar. Ayuda hacer una tabla con muchos y variados valores pre calculados
3. **En tiempo de Ejecución**. Cuando el programa ya se está ejecutando. Por Ej. La división por cero, o la raíz cuadrada de un número negativo. Es el programador el responsable de evitar estos Errores, ya sea validando los ingresos y/o capturando directamente los errores (Try catch) y ofreciendo soluciones.

1- Análisis del Problema, lo que implica Examinar cuidadosamente el problema para tener una idea clara de lo que se solicita y determinar los DATOS (de **Entrada, Salida y Auxiliares**).

✦ Aquí es evidente que estamos en presencia de la Resolvente de la Ecuación Cuadrática.

✦ Los datos de **Entrada** son tres valores numéricos que representan los coeficientes de la función: $f(x) = A \cdot X^2 + B \cdot X + C$.

A estos coeficientes los identificaremos con: **coef_A**, **coef_B** y **coef_C**. y los definiremos como datos de tipo Real. Hay lenguajes fuertemente tipeados (C#) que lo exigen, Raptor NO.

✦ En este caso, es necesario como dato **Auxiliar**, el valor que está bajo la raíz, (conocido como Discriminante), ya que como no manejamos el Cálculo Complejo no podríamos realizar ese cálculo si su valor fuera negativo. A este coeficiente lo identificamos con: **rDiscr**. Además, definiremos otra variable **Auxiliar** de Texto (string o cadena) para guardar mensajes de salida, y la llamaremos **sMensaje**.

✦ El **Diseño del Algoritmo**, [**Secuencia ordenada de pasos** - Sin Ambigüedades – **que conducen a la solución de un Problema**], que debe ser: **i – Preciso ii – Definido iii – Finito**.

✦ En este caso, el Algoritmo o **Proceso** viene dado por la fórmula de la Resolvente de la Ecuación, y con esta fórmula ya podemos continuar para obtener los valores **X₁** y **X₂** que serán las soluciones solicitadas, si existen.

✦ **La Salida**, que simplemente debe ser un Mensaje que imprime los valores de **X₁** y **X₂**, o algún mensaje de Error o indicación de algún problema.

2.A- Pseudo-Código:

Es un lenguaje muy libre, que utiliza **palabras reservadas** y exige las **sangrías**. La estructura básica es:

Algoritmo <TP4_G12_Ej12>

```
// Encontrar las Raíces Reales de una Ecuación Cuadrática con Validación de Entradas
// Usando la Resolvente de la Ecuación Cuadrática
// Fecha de Entrega: DD/MM/AA.
// Programador: Grupo 12 de PeC.
```

VARIABLES:

DE ENTRADA -	Real:	rCoef_A,	rCoef_B,	rCoef_C
DE SALIDA -	Real:	X₁,	X₂	
	Texto:	sMensaje		
AUXILIAR -	Real:	rDiscr		

INICIO

```

Leer_Numeros ( rCoef_A, rCoef_B, rCoef_C. ) // Función que carga Nros.
// Inicio del 1er. Condicional o Exterior
Si (rCoef_A != 0) // Este Condicional verifica que el Divisor A NO sea = 0.
    Asignar: rDiscr ← (rCoef_B^2 – (4*rCoef_A* rCoef_C) )
    // Inicio del 2do. Condicional o Interior o Anidado.
    Si (rDiscr >= 0) // Verifica que el Discr NO SEA Negativo.
        Asignar: X1 ← (-rCoef_B+Raiz2(rDiscr) ) / (2 * rCoef_A)
        Asignar: X2 ← (-rCoef_B-Raiz2(rDiscr))/ (2 * rCoef_A)
        Asignar: sMensaje ← "Raiz X1 = " + X1 + ", Raiz X2 = " + X2
        Imprimir ( sMensaje )
    SiNo // Si el Discr < 0, se ejecuta esta parte del código
        Imprimir ("Por ser el Discriminante = (B^2 – 4 * A * C) < 0")
        Imprimir ("Esta Ecuación tiene Solución Compleja")
        Imprimir ("No se Resuelve en esta Versión del Programa.")
    FinSi // Termina el Condicional interior

// Continúa con el condicional exterior
SiNo // Si el Divisor A = 0, se ejecuta esta parte del código
    Imprimir ("Por ser el Coeficiente del término Cuadrático A = 0")
    Imprimir ("Estamos en presencia de una Ecuación Lineal")
    Asignar: X ← ((- rCoef_C) / rCoef_B)
    Imprimir ("Solución Única X = (-C/B) = " + X)
FinSi // Termina el Condicional exterior

```

FIN // Termina el Programa

A continuación, se debería verificar que las respuestas dadas sean correctas para un grupo amplio de valores, realizando una tabla con los mismos y teniendo en consideración los valores especiales (A = 0; Discr < 0, etc...)

3.B- Diagramas de Flujo.

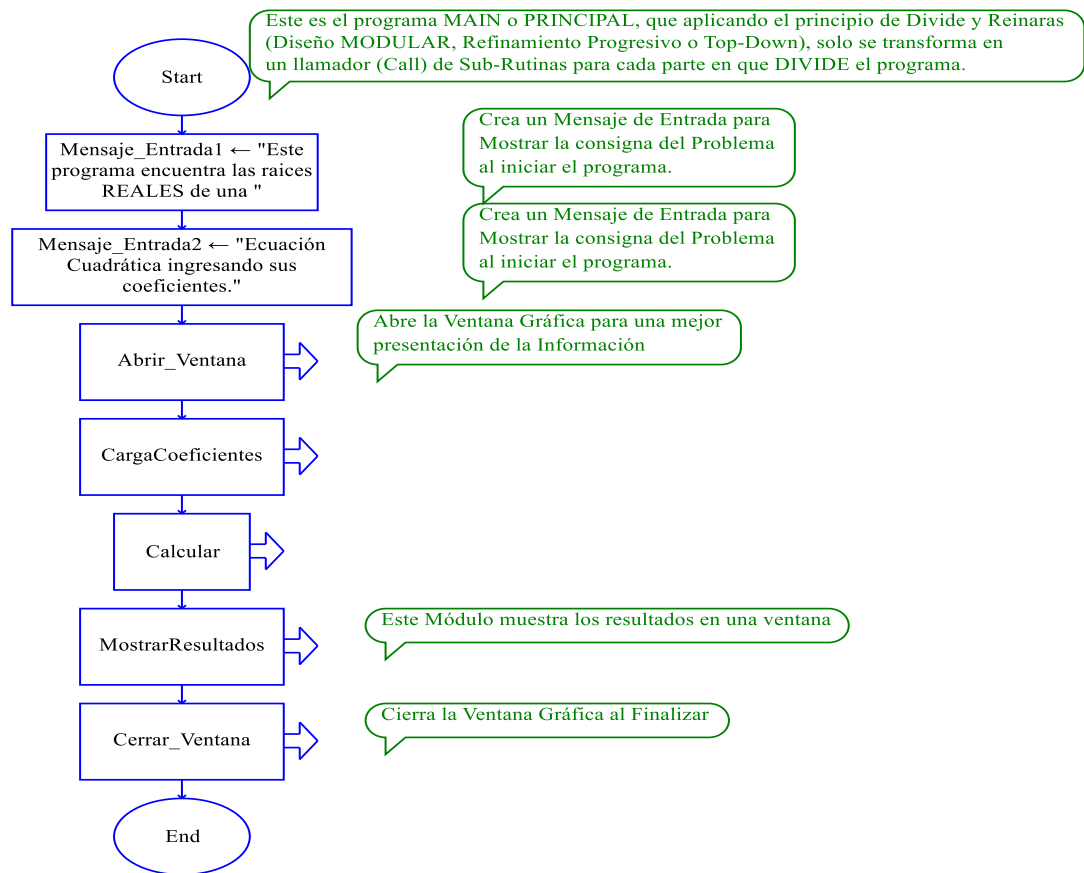
Utiliza **símbolos normalizados** unidos por flechas (**líneas de flujo**) que indican el orden en que debe ejecutarse cada paso.

Es muy importante porque permite:

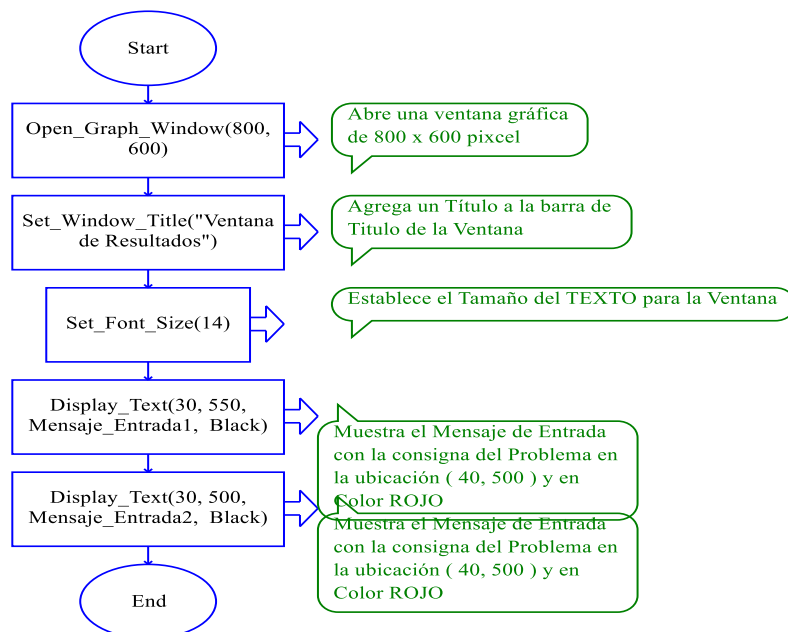
- Ver el flujo del Programa.
- Analizar la semántica del Condicional.
- Entender como piezas individuales interactúan para formar programas más complejos.

DF en Raptor con **Refinamiento Progresivo** y uso de Ventana Gráfica.

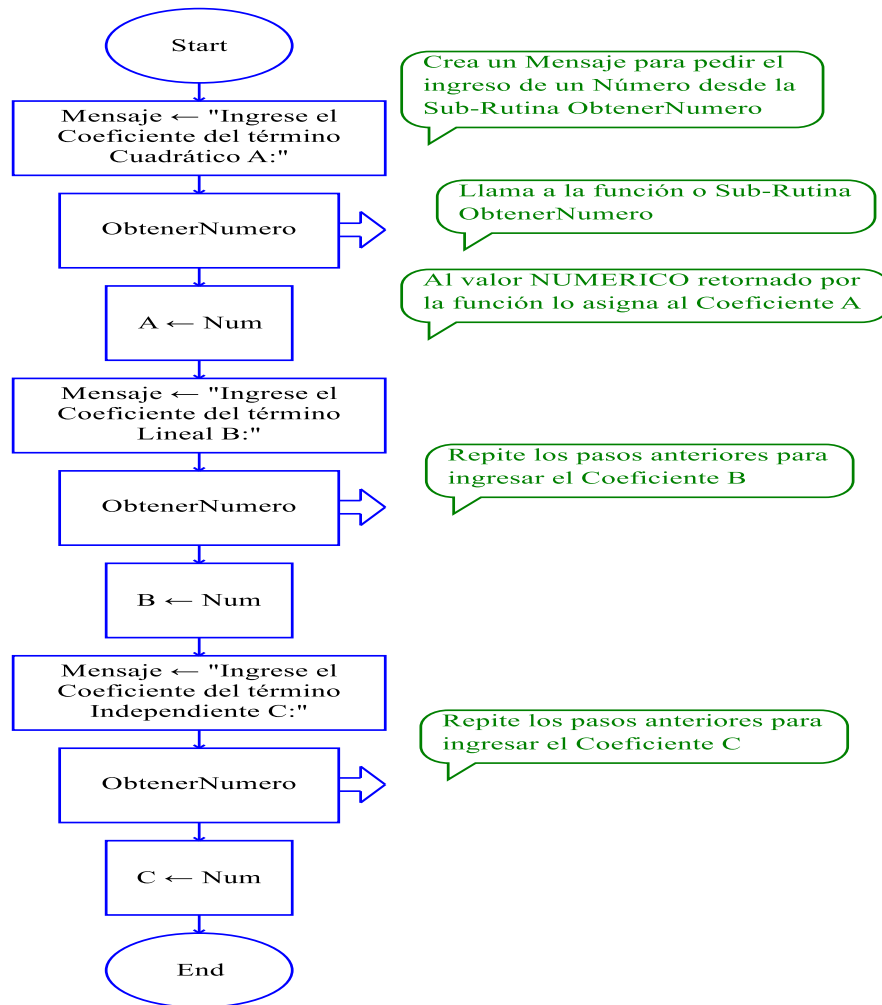
▪ **Programa Main()**



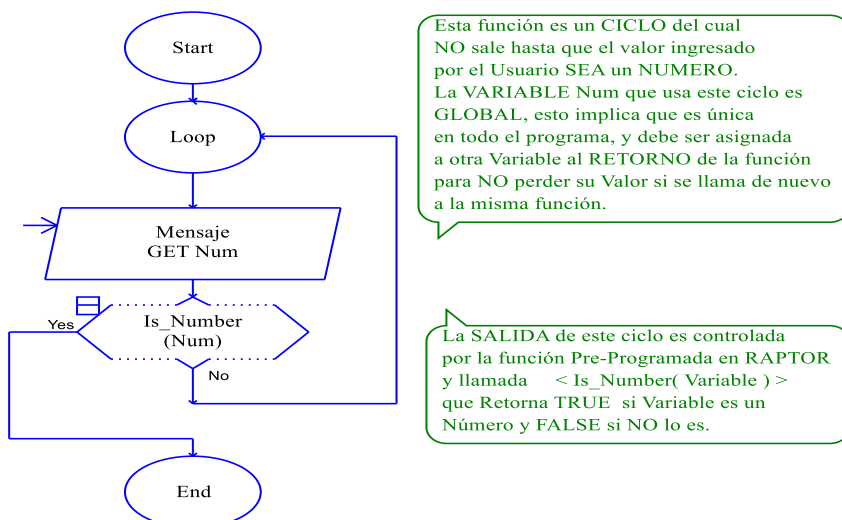
▪ **Sub Programa Abrir_Ventana()**



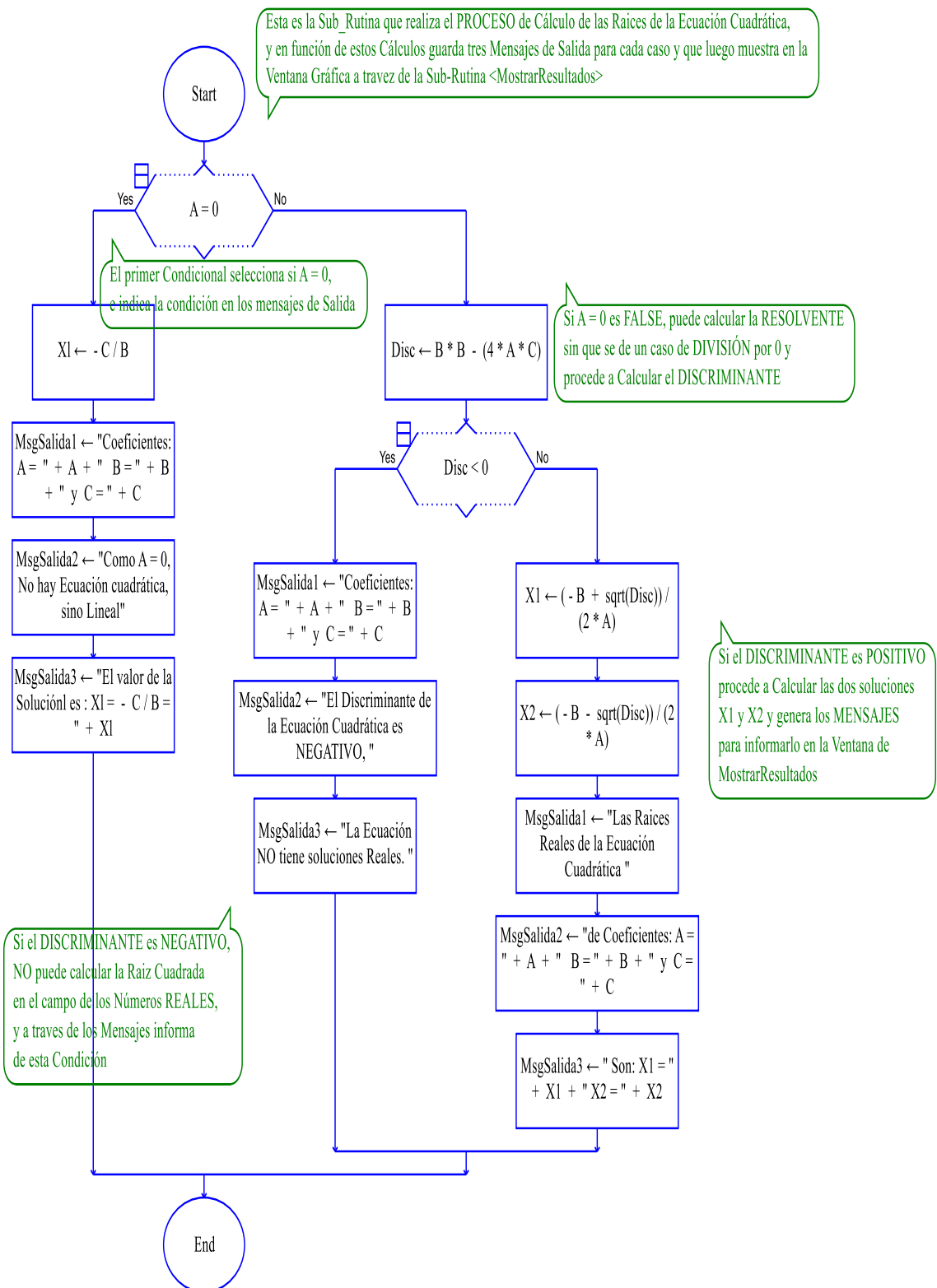
▪ **Sub Programa CargaCoeficientes()**

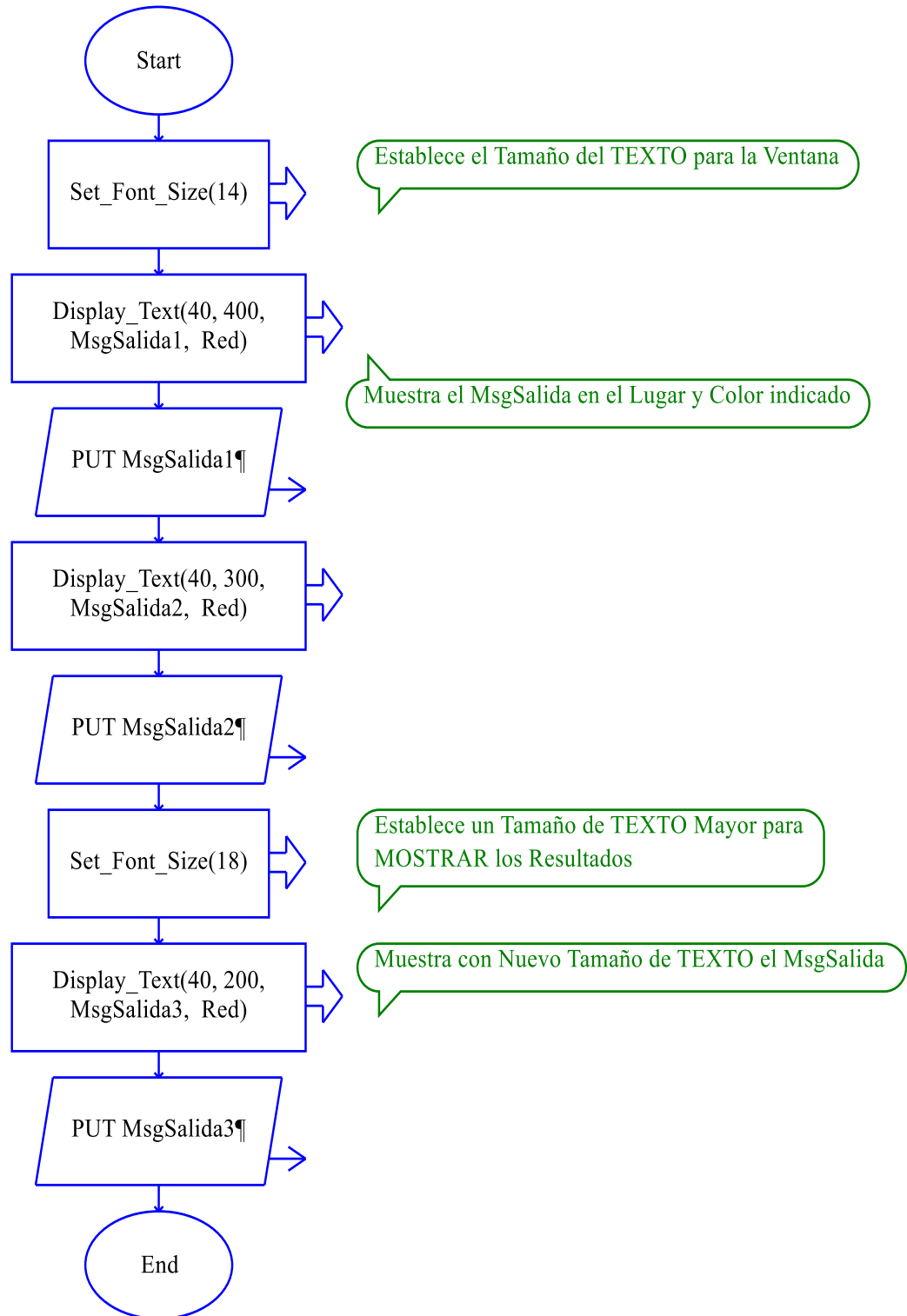


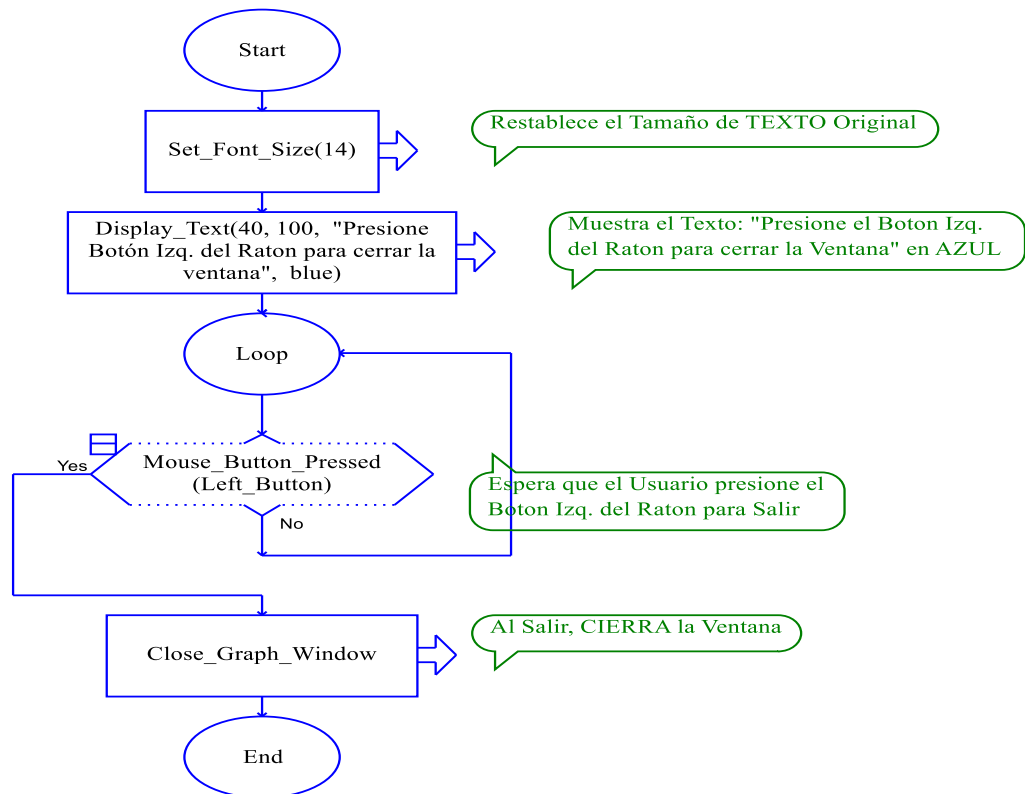
▪ **Sub Programa OtenerNumero(), usado por Sub Prog. CargaCoeficientes().**



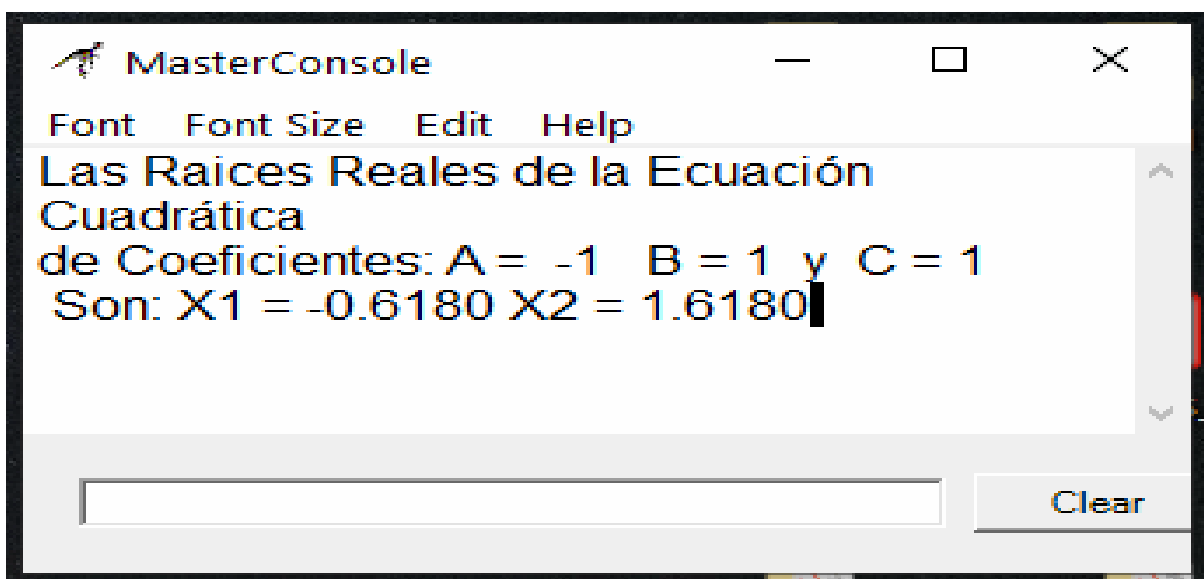
▪ Sub Programa Calcular()

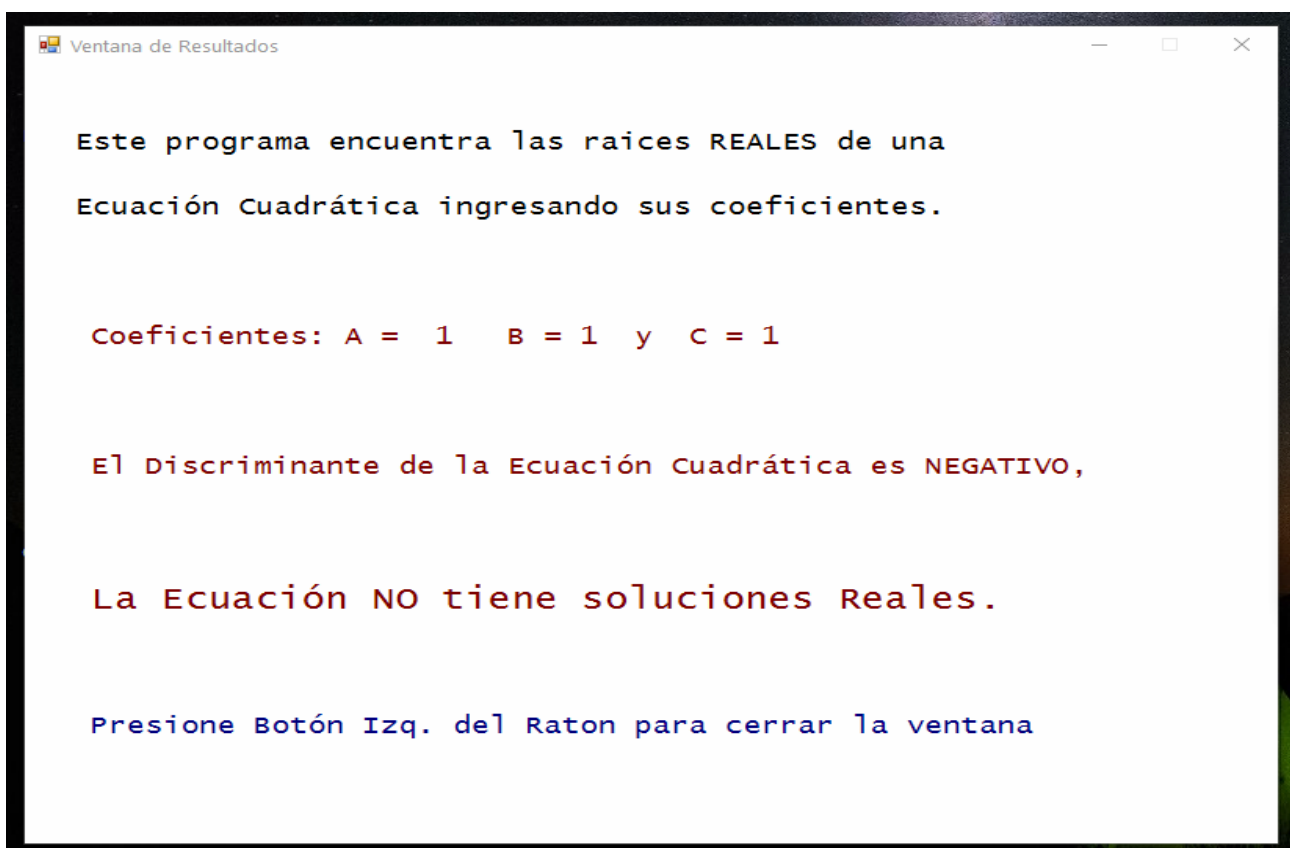
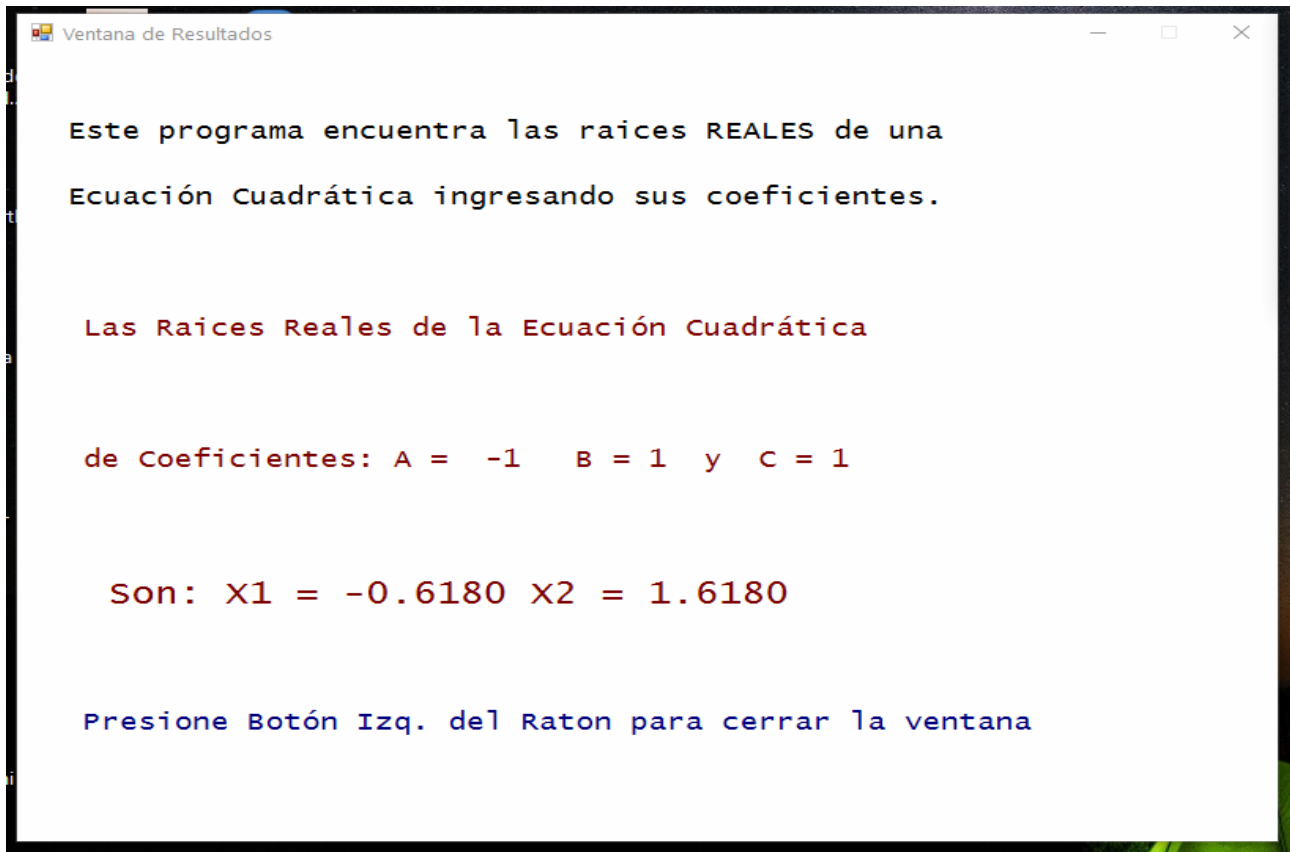


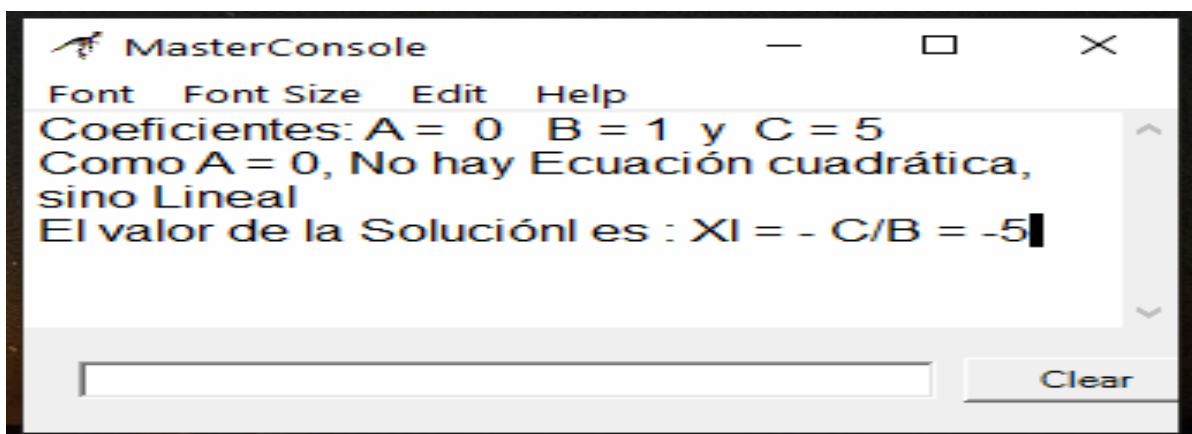
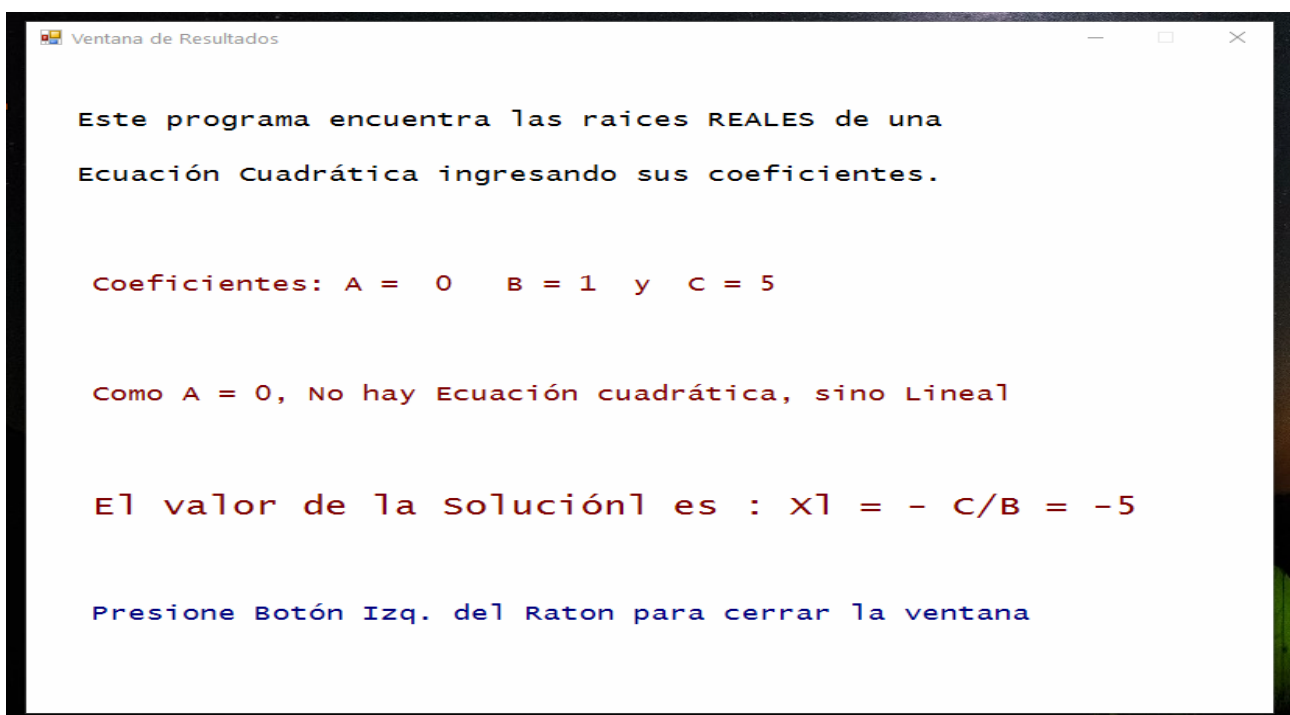
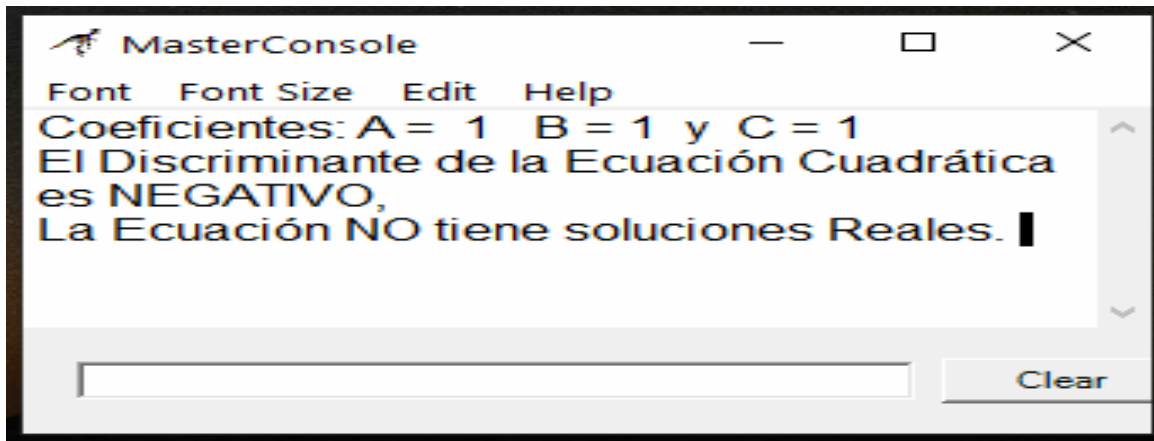
▪ Sub Programa *MostrarResultados()*

▪ Sub Programa Cerrar_Ventana()

Salidas en RAPTOR, de Consola y Ventana para las distintas Soluciones según los valores de Entrada.







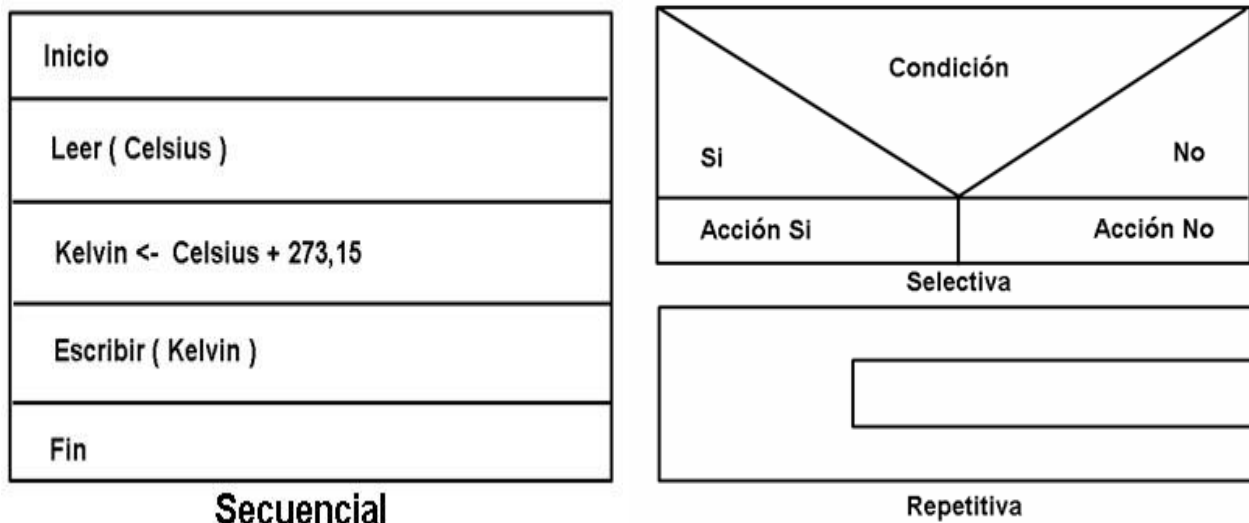
3.C- Diagrama Nassi-Schneiderman o de Chapin.

Reúne características propias de los anteriores y favorece la programación estructurada.

Consta de una serie de cajas contiguas que se leerán siempre de Arriba-Abajo.

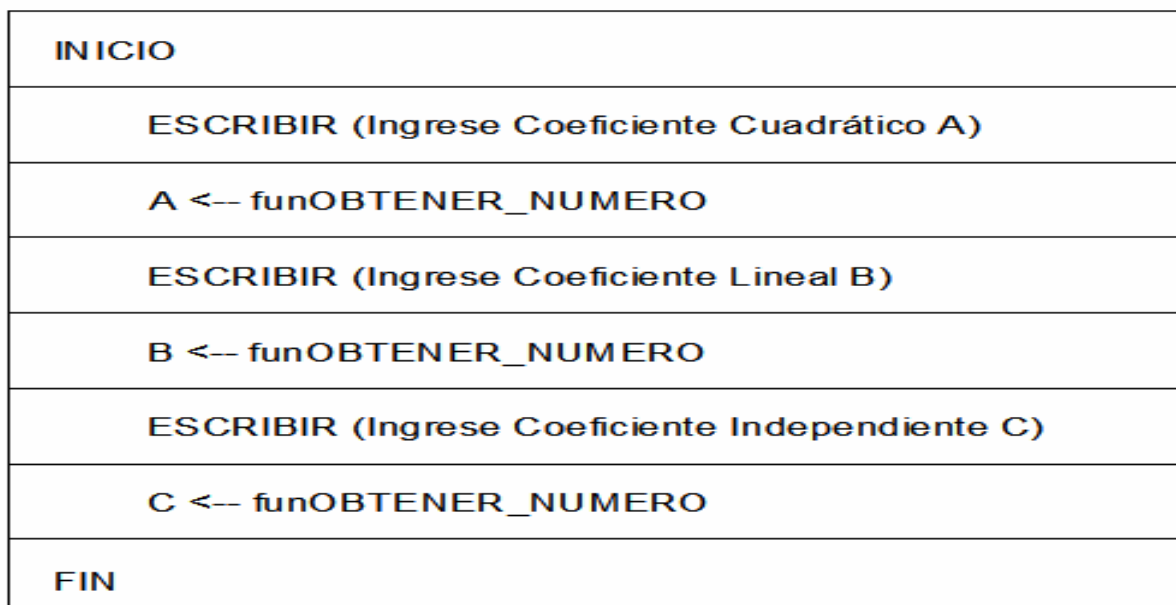
Las tres estructuras básicas tienen su propia representación:

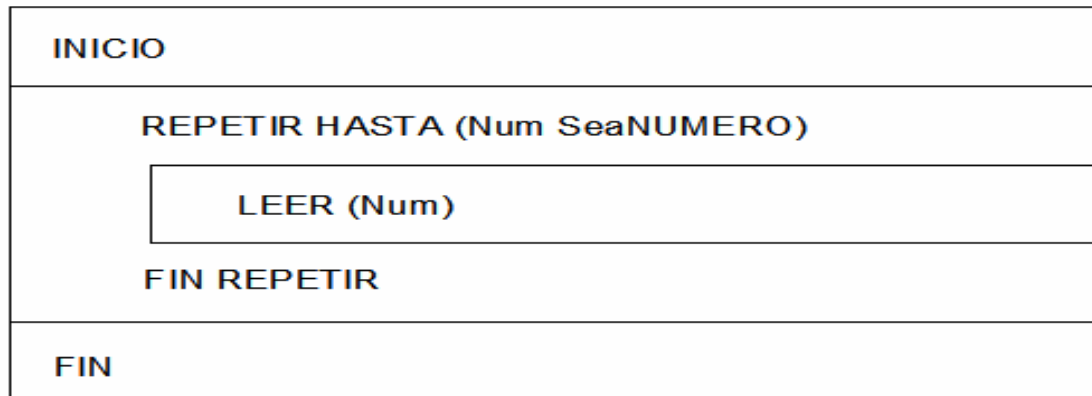
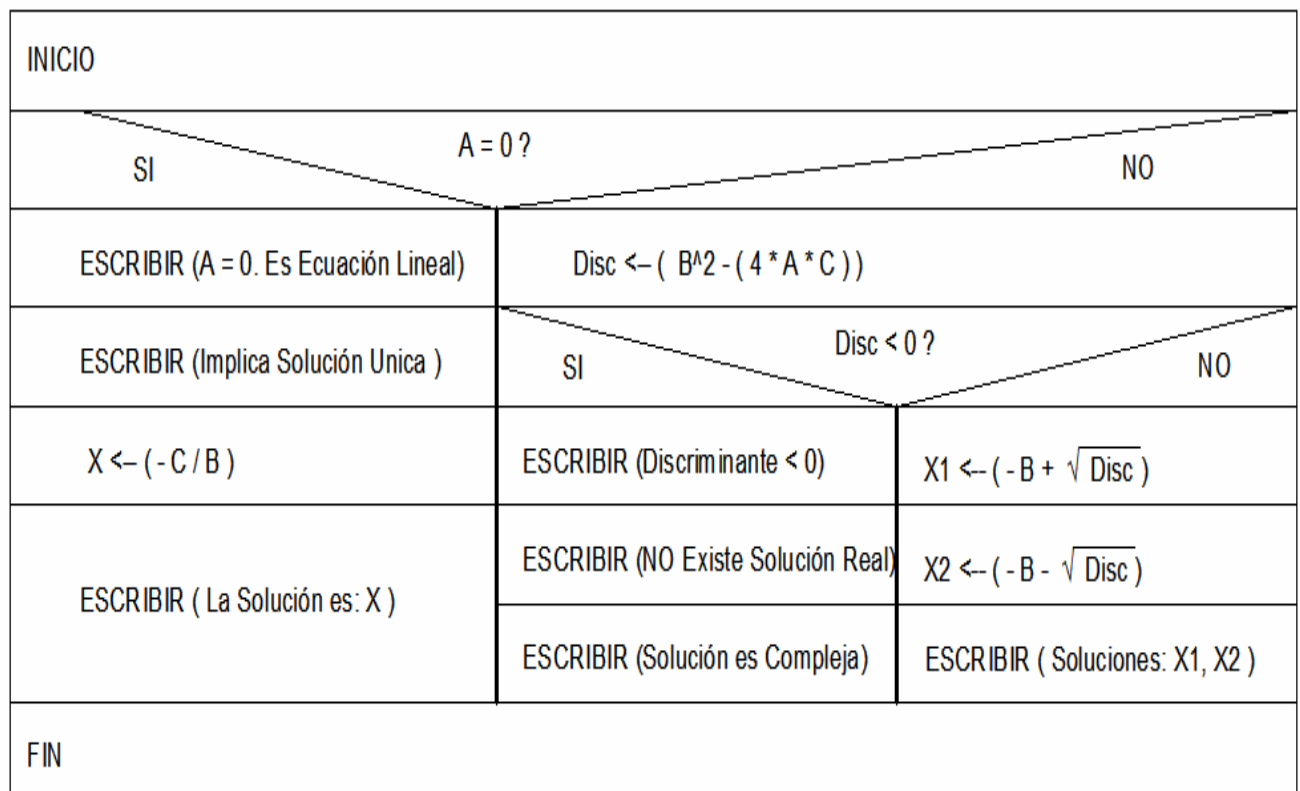
- **Secuenciales.** Ejecuta todas las instrucciones en el orden dado.
- **Selectivas o Condicionales.** Se ejecutan en función de una condición
- **Repetitivas.** Repiten una porción de código en base a una condición.



Representamos como Ejemplo de las tres Estructuras Básicas, Algunas SubRutinas de RAPTOR:

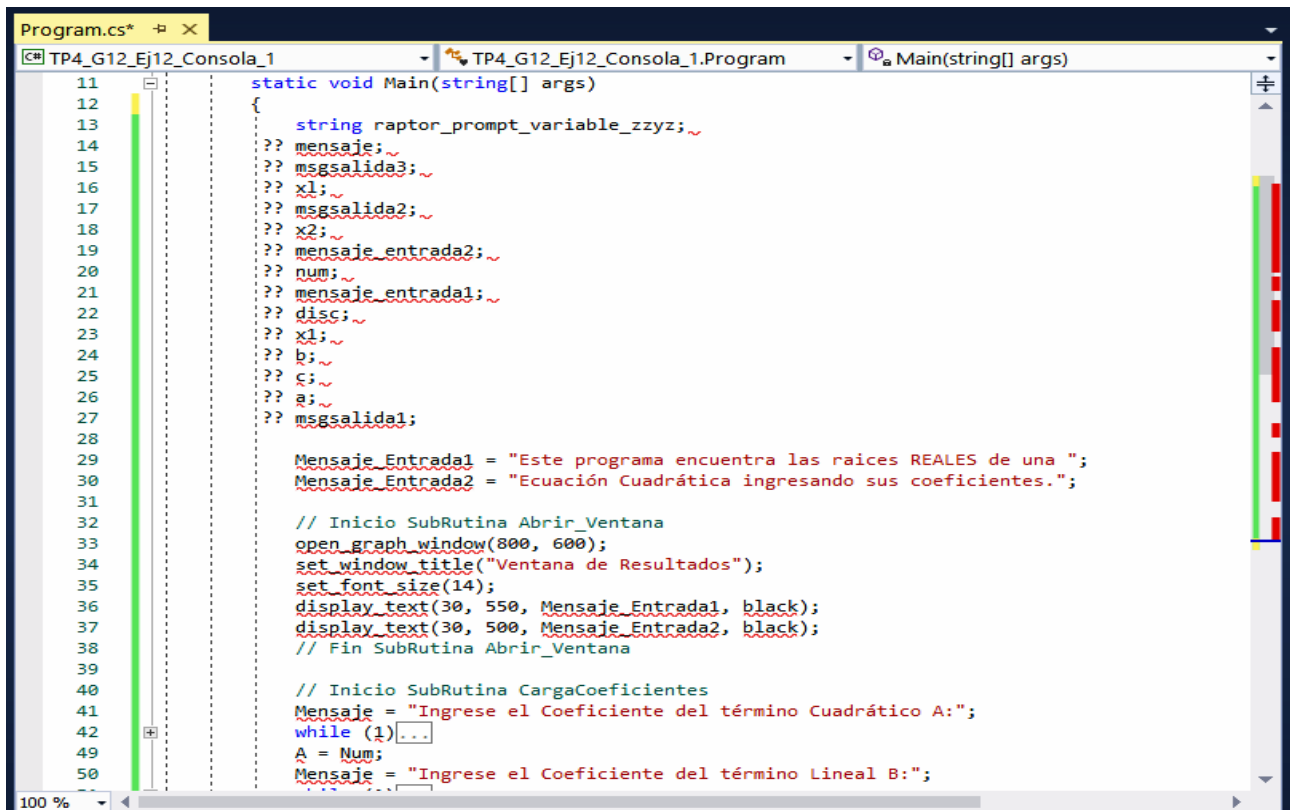
SubRutina CARGAR_COEFICIENTES



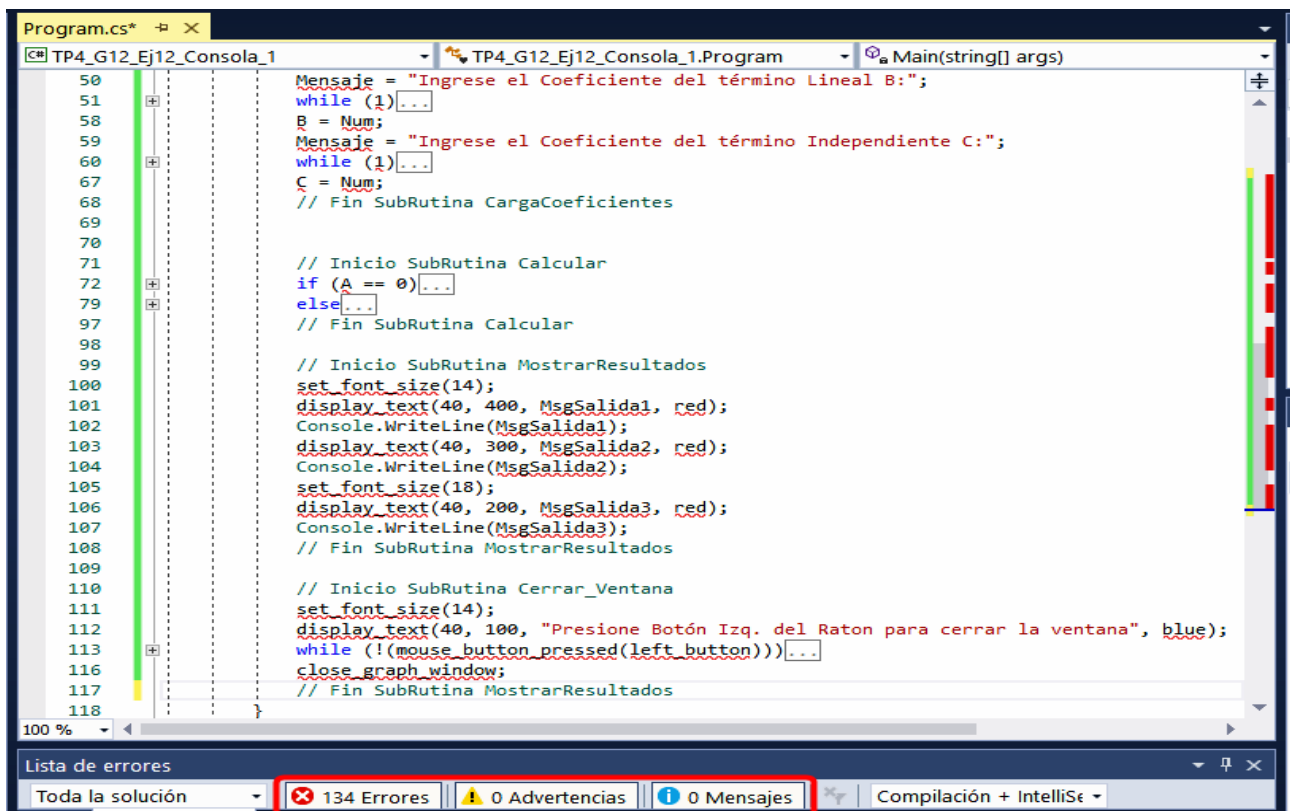
SubRutina funOBTENER_NUMERO**SubRutina CALCULAR**

4. Codificación en VSC# Consola usando Módulos

Comenzamos copiando el Código generado en Raptor a VSC# en Consola.



```
Program.cs* [x]
TP4_G12_Ej12_Consola_1 TP4_G12_Ej12_Consola_1.Program Main(string[] args)
11 static void Main(string[] args)
12 {
13     string raptor_prompt_variable_zzyz;
14     ?? mensaje;
15     ?? msgsalida3;
16     ?? x1;
17     ?? msgsalida2;
18     ?? x2;
19     ?? mensaje_entrada2;
20     ?? num;
21     ?? mensaje_entrada1;
22     ?? disc;
23     ?? x1;
24     ?? b;
25     ?? c;
26     ?? a;
27     ?? msgsalida1;
28
29     Mensaje_Entrada1 = "Este programa encuentra las raices REALES de una ";
30     Mensaje_Entrada2 = "Ecuación Cuadrática ingresando sus coeficientes.";
31
32     // Inicio SubRutina Abrir_Ventana
33     open_graph_window(800, 600);
34     set_window_title("Ventana de Resultados");
35     set_font_size(14);
36     display_text(30, 550, Mensaje_Entrada1, black);
37     display_text(30, 500, Mensaje_Entrada2, black);
38     // Fin SubRutina Abrir_Ventana
39
40     // Inicio SubRutina CargaCoeficientes
41     Mensaje = "Ingrese el Coeficiente del término Cuadrático A:";
42     while (1) ...
43     A = Num;
44     Mensaje = "Ingrese el Coeficiente del término Lineal B:";
```

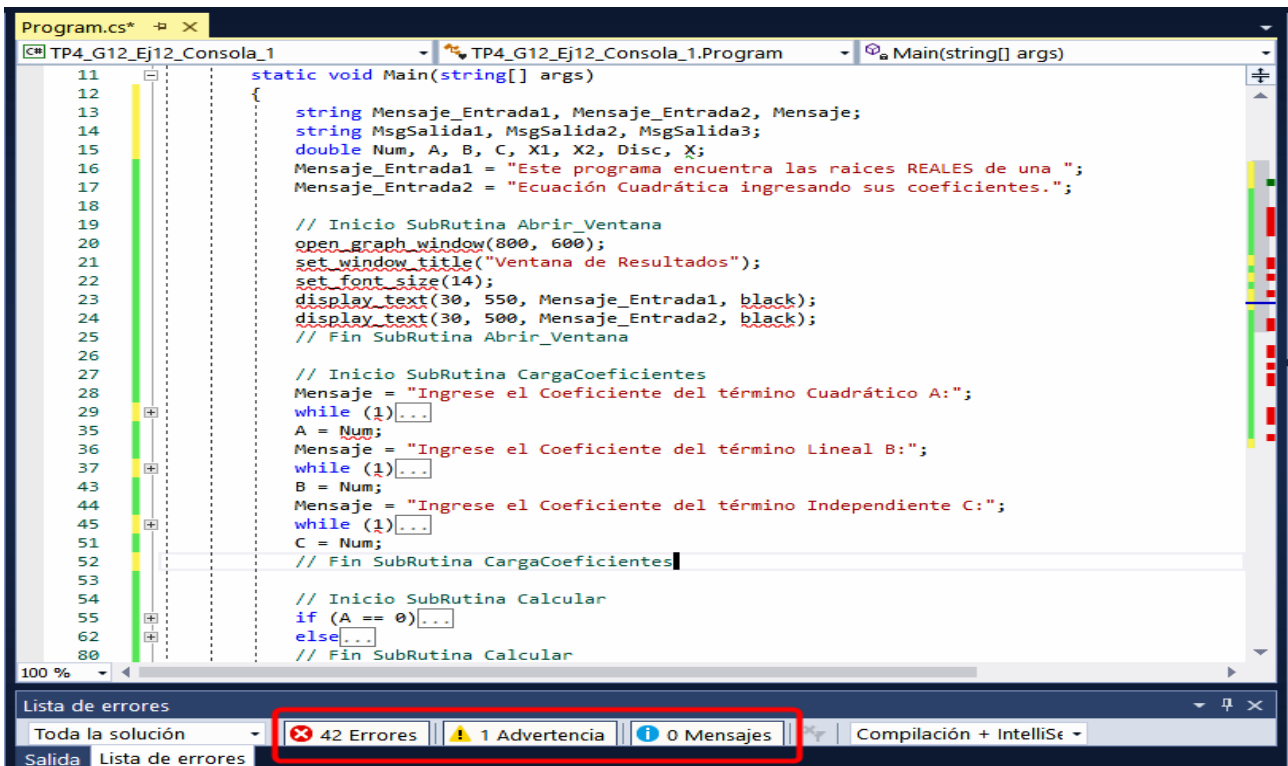


```
Program.cs* [x]
TP4_G12_Ej12_Consola_1 TP4_G12_Ej12_Consola_1.Program Main(string[] args)
50 Mensaje = "Ingrese el Coeficiente del término Lineal B:";
51 while (1) ...
52 B = Num;
53 Mensaje = "Ingrese el Coeficiente del término Independiente C:";
54 while (1) ...
55 C = Num;
56 // Fin SubRutina CargaCoeficientes
57
58 // Inicio SubRutina Calcular
59 if (A == 0) ...
60 else ...
61 // Fin SubRutina Calcular
62
63 // Inicio SubRutina MostrarResultados
64 set_font_size(14);
65 display_text(40, 400, MsgSalida1, red);
66 Console.WriteLine(MsgSalida1);
67 display_text(40, 300, MsgSalida2, red);
68 Console.WriteLine(MsgSalida2);
69 set_font_size(18);
70 display_text(40, 200, MsgSalida3, red);
71 Console.WriteLine(MsgSalida3);
72 // Fin SubRutina MostrarResultados
73
74 // Inicio SubRutina Cerrar_Ventana
75 set_font_size(14);
76 display_text(40, 100, "Presione Botón Izq. del Raton para cerrar la ventana", blue);
77 while (!(mouse_button_pressed(left_button))) ...
78 close_graph_window;
79 // Fin SubRutina MostrarResultados
80 }
```

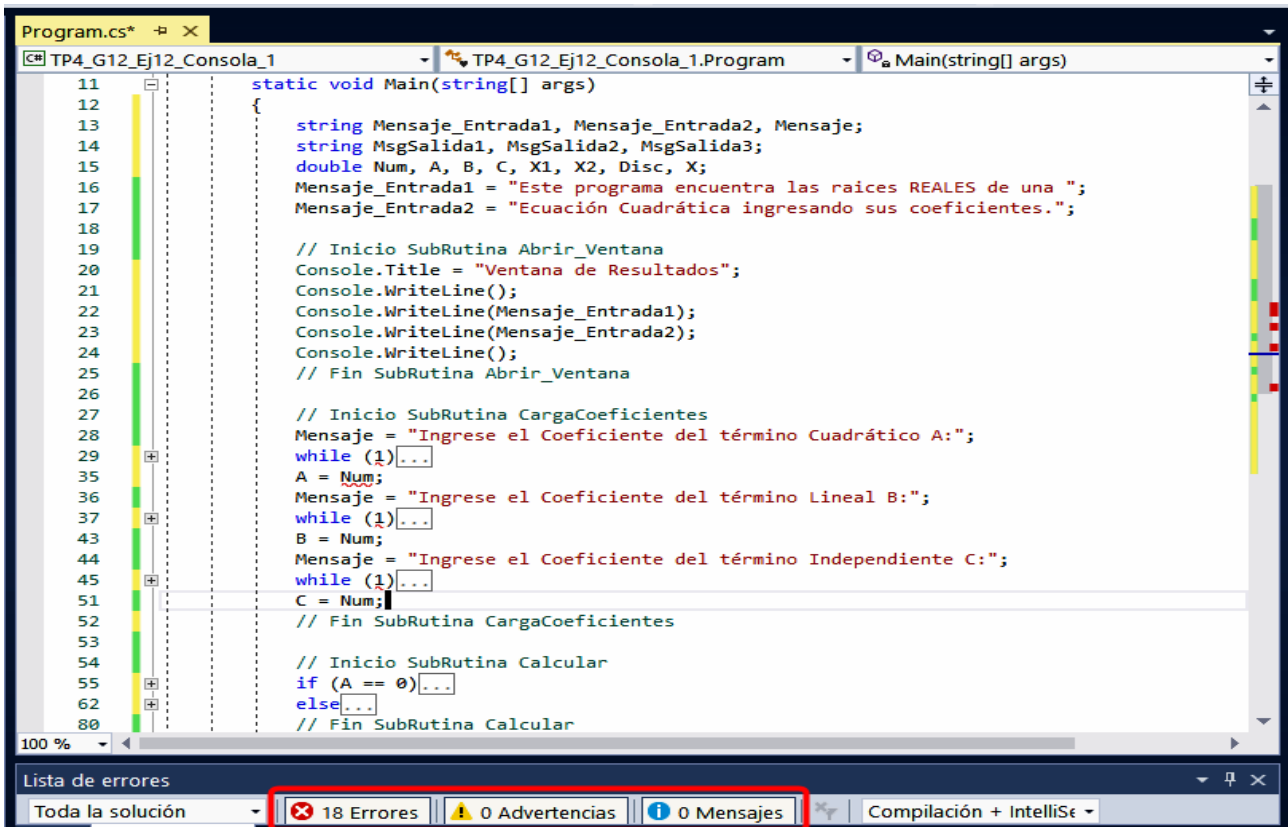
Lista de errores

Toda la solución 134 Errores 0 Advertencias 0 Mensajes Compilación + IntelliSense

Vemos 134 Errores, que sabemos que se reducen en gran cantidad, redefiniendo las variables:



Después de Redefinir las variables, vemos solo 42 Errores, y observando el código, aparecen en las Ventanas gráficas de Raptor, que evidentemente VSC# Consola NO maneja, y debemos adaptarlas.



Después de cambiar las **Graph_Windows** de Raptor por **Console.WriteLine**, estamos en los 18 Errores, y vemos que estos se encuentran en las Subrutinas: **CargarCoeficientes** y **Calcular**, por lo que expandimos los signos [+] de la Izquierda (al lado del número de línea), y procedemos a investigar estos Errores.

El primer Error que encontramos es: **while** (1) [Este 1 en Raptor es el **True** de C#]. Si cambiamos a **while** (**true**), salvamos el Error, y el ciclo es eterno, hasta que ejecute un **break**; que aparece después de cumplir una condición. **if** (Is_Numeric(Num))) **break**; y este break quiere decir que debe interrumpir el ciclo (y de una forma poco elegante)

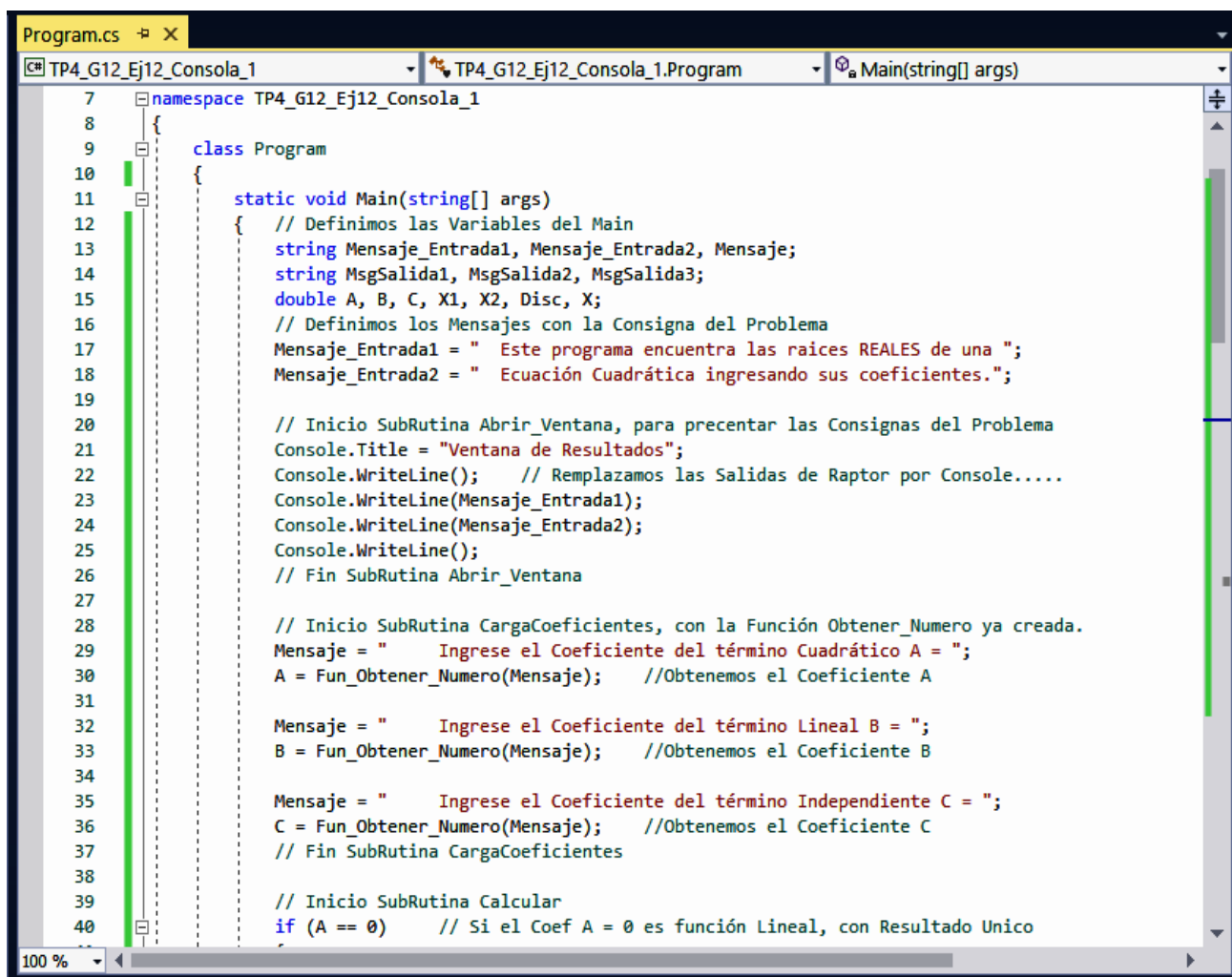
Otro problema, es que la función (Is_Numeric(Num)), No la reconoce el VSC#, y debemos buscar alguna que acepte (Google).

Otros dos Errores aparecen cuando calculamos las soluciones:

$X1 = (-B + \sqrt{\text{Disc}}) / (2 * A)$; [NO reconoce la función **Sqrt()**]; pero le indicamos que está en el paquete de Funciones Matemáticas y listo con: $X1 = (-B + \text{Math.Sqrt}(\text{Disc})) / (2 * A)$;

Además, vemos que la parte que se refiere a la función **Fun_Obtener_Numero()**, el código que traemos desde Raptor se repite tres veces, que son las veces que se llama a esta subrutina (para cargar los Coeficientes A, B y C), entonces, es conveniente crear ya esta función para evitar la repetición del Código.

Con los cambios sugeridos, llegamos al código que mostramos a continuación:



```
Program.cs x
C# TP4_G12_Ej12_Consola_1 TP4_G12_Ej12_Consola_1.Program Main(string[] args)
7 namespace TP4_G12_Ej12_Consola_1
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        { // Definimos las Variables del Main
13            string Mensaje_Entrada1, Mensaje_Entrada2, Mensaje;
14            string MsgSalida1, MsgSalida2, MsgSalida3;
15            double A, B, C, X1, X2, Disc, X;
16            // Definimos los Mensajes con la Consigna del Problema
17            Mensaje_Entrada1 = " Este programa encuentra las raices REALES de una ";
18            Mensaje_Entrada2 = " Ecuación Cuadrática ingresando sus coeficientes.";
19
20            // Inicio SubRutina Abrir_Ventana, para precentar las Consignas del Problema
21            Console.Title = "Ventana de Resultados";
22            Console.WriteLine(); // Reemplazamos las Salidas de Raptor por Console.....
23            Console.WriteLine(Mensaje_Entrada1);
24            Console.WriteLine(Mensaje_Entrada2);
25            Console.WriteLine();
26            // Fin SubRutina Abrir_Ventana
27
28            // Inicio SubRutina CargaCoeficientes, con la Función Obtener_Numero ya creada.
29            Mensaje = " Ingrese el Coeficiente del término Cuadrático A = ";
30            A = Fun_Obtener_Numero(Mensaje); //Obtenemos el Coeficiente A
31
32            Mensaje = " Ingrese el Coeficiente del término Lineal B = ";
33            B = Fun_Obtener_Numero(Mensaje); //Obtenemos el Coeficiente B
34
35            Mensaje = " Ingrese el Coeficiente del término Independiente C = ";
36            C = Fun_Obtener_Numero(Mensaje); //Obtenemos el Coeficiente C
37            // Fin SubRutina CargaCoeficientes
38
39            // Inicio SubRutina Calcular
40            if (A == 0) // Si el Coef A = 0 es función Lineal, con Resultado Unico
```

```

Program.cs
TP4_G12_Ej12_Consola_1
TP4_G12_Ej12_Consola_1.Program
Main(string[] args)

39 // Inicio SubRutina Calcular
40 if (A == 0) // Si el Coef A = 0 es función Lineal, con Resultado Unico
41 {
42     X = -C / B;
43     MsgSalida1 = " Coeficientes: A = " + A + " B = " + B + " y C = " + C;
44     MsgSalida2 = " Como A = 0, No hay Ecuación cuadrática, sino Lineal";
45     MsgSalida3 = " El valor de la Solución1 es : X1 = - C/B = " + X;
46 }
47 else // Si (A != 0) NO hay problema de Division por 0, y La Ecuación es Cuadrática.
48 {
49     Disc = B * B - (4 * A * C); // Calculamos el Discriminante
50     if (Disc < 0) // Si (Disc < 0), tenemos Raiz Cuadrada imposible con los Reales
51     {
52         MsgSalida1 = " Coeficientes: A = " + A + " B = " + B + " y C = " + C;
53         MsgSalida2 = " El Discriminante de la Ecuación Cuadrática es NEGATIVO,";
54         MsgSalida3 = " La Ecuación NO tiene soluciones Reales. ";
55     }
56     else // Caso contrario calculamos la Raiz Cuadrada y obtenemos X1 y X2
57     {
58         X1 = (-B + Math.Sqrt(Disc)) / (2 * A); //Otros 2 Errores se solucionan aquí
59         X2 = (-B - Math.Sqrt(Disc)) / (2 * A); //Con <Math.Sqrt(> para la Raiz
60         MsgSalida1 = " Las Raices Reales de la Ecuación Cuadrática ";
61         MsgSalida2 = " de Coeficientes: A = " + A + " B = " + B + " y C = " + C;
62         MsgSalida3 = " Son: X1 = " + X1 + " X2 = " + X2;
63     }
64 }
65 // Fin SubRutina Calcular
66
67 // Inicio SubRutina MostrarResultados, de nuevo, Salidas de Raptor por Console....
68 Console.WriteLine();
69 Console.WriteLine(MsgSalida1);
70 Console.WriteLine();
71 Console.WriteLine(MsgSalida2);
72 Console.WriteLine();

```

```

Program.cs
TP4_G12_Ej12_Consola_1
TP4_G12_Ej12_Consola_1.Program
Main(string[] args)

66
67 // Inicio SubRutina MostrarResultados, de nuevo, Salidas de Raptor por Console....
68 Console.WriteLine();
69 Console.WriteLine(MsgSalida1);
70 Console.WriteLine();
71 Console.WriteLine(MsgSalida2);
72 Console.WriteLine();
73 Console.WriteLine(MsgSalida3);
74 Console.WriteLine();
75 // Fin SubRutina MostrarResultados
76
77 // Inicio SubRutina Cerrar, esperamos la acción del Usuario para Salir.
78 Console.WriteLine();
79 Console.WriteLine(" Presione cualquier tecla para cerrar la ventana");
80 Console.ReadKey();
81 // Fin SubRutina Cerrar
82 }
83
84 static double NumReturn; //Valor de RETORNO de la función fuera de Módulos y Estático
85
86 // La función, recibe un string (El Mensaje a mostrar al Usuario) y retorna un DOBLE.
87 static double Fun_Obtener_Numero(string StrMensaje)
88 {
89     //Creamos una función para obtener los Coeficientes, y asegurarnos que sean NUMEROS
90     //Definimos las variables Locales
91     string StrNum; //Representa el Texto que ingreso el Usuario por Teclado
92     double Num; //Representa el Número Convertido a Doble
93     Boolean Bandera = true; //Definimos una variable Bool para manejar el Ciclo While
94
95     Console.WriteLine(); //Sin comentarios, ya sabemos que hace.
96
97     while (Bandera) //Como Bandera = True, el Ciclo es eterno hasta que cambie a False
98     {
99         // Inicio Ciclo While
100         Console.WriteLine(StrMensaje); //Muestra el Mensaje que se envió al llamar la Función
101         StrNum = Console.ReadLine(); //Recoge el Texto Tipeado por el Usuario

```

```

Program.cs
TP4_G12_Ej12_Consola_1
TP4_G12_Ej12_Consola_1.Program
Main(string[] args)

87 static double Fun_Obtener_Numero(string StrMensaje)
88 { //Creamos una función para obtener los Coeficientes, y asegurarnos que sean NUMEROS
89 //Definimos las variables Locales
90 string StrNum; //Representa el Texto que ingreso el Usuario por Teclado
91 double Num; //Representa el Número Convertido a Doble
92 Boolean Bandera = true; //Definimos una variable Bool para manejar el Ciclo While
93
94 Console.WriteLine(); //Sin comentarios, ya sabemos que hace.
95
96 while (Bandera) //Como Bandera = True, el Ciclo es eterno hasta que cambie a False
97 { // Inicio Ciclo While
98 Console.Write(StrMensaje); //Muestra el Mensaje que se envió al llamar la Función
99 StrNum = Console.ReadLine(); //Recoge el Texto Tipeado por el Usuario
100
101 if ((Double.TryParse(StrNum, out Num)))
102 /* TryParse(String, Double)
103 * Convierte la representación en forma de cadena de un número en
104 * el número de punto flotante de precisión doble equivalente.
105 * Un valor devuelto indica si la conversión se realizó correctamente
106 * o si se produjeron errores. */
107 { // Si la conversión fue exitosa, debemos salir del Ciclo,
108 Bandera = false; //y lo hacemos asignando el valor de Bandera = false
109 NumReturn = Num; //y pasamos el Num retornado por TryParse al valor a Retornar
110 }
111 else // Caso contrario, podríamos NO hacer NADA, xq' seguiría Bandera = true
112 {
113 Bandera = true; // Pero lo repetimos para afirmar el concepto
114 }
115 } // Fin Ciclo While
116 return NumReturn; //Estamos seguros que el Usuario Ingreso un Numero y lo Retornamos
117 } // Fin de la Función que SOLO me permitirá ingresar NUMEROS.
118
100 %
Lista de errores
0 Errores 0 Advertencias 1 Mensaje

```

El programa está Auto explicado y sin Errores. A continuación presentamos la Consola de VSC#:

```

Ventana de Resultados

Este programa encuentra las raices REALES de una
Ecuación Cuadrática ingresando sus coeficientes.

Ingrese el Coeficiente del término Cuadrático A = KLJAÑLJÑKJFK
Ingrese el Coeficiente del término Cuadrático A = -3

Ingrese el Coeficiente del término Lineal B = KJÑFHPRQIN7676
Ingrese el Coeficiente del término Lineal B = 5

Ingrese el Coeficiente del término Independiente C = M.,.NXB4325
Ingrese el Coeficiente del término Independiente C = 7

Las Raices Reales de la Ecuación Cuadrática
de Coeficientes: A = -3 B = 5 y C = 7

Son: X1 = -0,906717751485092 X2 = 2,57338441815176

Presione cualquier tecla para cerrar la ventana

```

Ahora, solo nos queda plantear una nueva versión, la `Consola_2`, que sea totalmente Modular.

```

Program.cs*
[TP4_G12_Ej12_Consola_2] TP4_G12_Ej12_Consola_2.Program Main(string[] args)
7 namespace TP4_G12_Ej12_Consola_2
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        {
13            // Definimos las Variables del Main
14            string Mensaje_Entrada1, Mensaje_Entrada2, Mensaje;
15            string MsgSalida1, MsgSalida2, MsgSalida3;
16            double A, B, C, X1, X2, Disc, X;
17
18            Console.Title = ".#. TP4_G12_Ej12 .#. "; //Pone Titulo a la ventana
19
20            // Definimos los Mensajes con la Consigna del Problema
21            Mensaje_Entrada1 = " Este programa encuentra las raices REALES de una ";
22            Mensaje_Entrada2 = " Ecuación Cuadrática ingresando sus coeficientes.";
23            Console.WriteLine();
24
25            Sub_Mostrar_Mensaje(Mensaje_Entrada1); //Llamo al procedimiento Sub_Mostrar_Mensaje
26            Sub_Mostrar_Mensaje(Mensaje_Entrada2);
27            Console.WriteLine();
28
29            // Inicio Carga de Coeficientes, con la Función Obtener_Numero ya creada.
30            Mensaje = " Ingrese el Coeficiente del término Cuadrático A = ";
31            A = Fun_Obtener_Numero(Mensaje); //Obtenemos el Coeficiente A
32
33            Mensaje = " Ingrese el Coeficiente del término Lineal B = ";
34            B = Fun_Obtener_Numero(Mensaje); //Obtenemos el Coeficiente B
35
36            Mensaje = " Ingrese el Coeficiente del término Independiente C = ";
37            C = Fun_Obtener_Numero(Mensaje); //Obtenemos el Coeficiente C
38            // Fin Carga de Coeficientes
39
40            // Inicio del código de Calcular
41            if (A == 0) ...
42            else ... // Fin del código de Calcular
43
44            // Inicio Mostrar Resultados,
45            Sub_Mostrar_Mensaje(MsgSalida1); //Llamo al procedimiento Sub_Mostrar_Mensaje
46            Sub_Mostrar_Mensaje(MsgSalida2);
47            Sub_Mostrar_Mensaje(MsgSalida3);
48            // Fin Mostrar Resultados
49
50            // Inicio SubRutina Cerrar, esperamos la acción del Usuario para Salir.
51            Console.WriteLine();
52            Sub_Mostrar_Mensaje(" Presione cualquier tecla para SALIR");
53            Console.ReadKey();
54            // Fin SubRutina Cerrar
55        }
56
57        // SubRutina Mostrar Mensaje
58        static void Sub_Mostrar_Mensaje(string StrMensaje) ...
59
60        static double NumReturn; //Valor de RETORNO de la función fuera de Módulos y Estático
61
62        // La función, recibe un string (El Mensaje a mostrar al Usuario) y retorna un DOBLE.
63        static double Fun_Obtener_Numero(string StrMensaje) ... // Fin de la Función que SOLO me perm
64    }
65 }
  
```

Lista de errores

Toda la solución 0 Errores 0 Advertencias 1 Mensaje Compilación + IntelliSense

Hemos creado, además de la `Fun_Obtener_Numero()`, la `Sub_Mostrar_Mensaje()`, y con ellas hemos modularizado y simplificado todo el desarrollo del código del Problema.

El Código de la **Subrutina Calcular**, lo hemos dejado sin modificaciones (como lo generó Raptor), y que las versiones de la Subrutina **MostrarResultados** y **Salir** de la Aplicación, podemos verificar que es identico a la Versión Consola_1

```

Program.cs*  TP4_G12_Ej12_Consola_2
TP4_G12_Ej12_Consola_2.Program  Main(string[] args)

38 // Fin Carga de Coeficientes
39
40 // Inicio del código de Calcular
41 if (A == 0) // Si el Coef A = 0 es función Lineal, con Resultado Unico
42 {
43     X = -C / B;
44     MsgSalida1 = " Coeficientes: A = " + A + " B = " + B + " y C = " + C;
45     MsgSalida2 = " Como A = 0, No hay Ecuación cuadrática, sino Lineal";
46     MsgSalida3 = " El valor de la Solución1 es : X1 = - C/B = " + X;
47 }
48 else // Si (A != 0) NO hay problema de Division por 0, y La Ecuación es Cuadrática.
49 {
50     Disc = B * B - (4 * A * C); // Calculamos el Discriminante
51     if (Disc < 0) // Si (Disc < 0), tenemos Raiz Cuadrada imposible con los Reales
52     {
53         MsgSalida1 = " Coeficientes: A = " + A + " B = " + B + " y C = " + C;
54         MsgSalida2 = " El Discriminante de la Ecuación Cuadrática es NEGATIVO,";
55         MsgSalida3 = " La Ecuación NO tiene soluciones Reales. ";
56     }
57     else // Caso contrario calculamos la Raiz Cuadrada y obtenemos X1 y X2
58     {
59         X1 = (-B + Math.Sqrt(Disc)) / (2 * A); //Otros 2 Errores se solucionan aquí
60         X2 = (-B - Math.Sqrt(Disc)) / (2 * A); //Con <Math.Sqrt(> para la Raiz
61         MsgSalida1 = " Las Raices Reales de la Ecuación Cuadrática ";
62         MsgSalida2 = " de Coeficientes: A = " + A + " B = " + B + " y C = " + C;
63         MsgSalida3 = " Son: X1 = " + X1 + " X2 = " + X2;
64     }
65 } // Fin del código de Calcular
66
67 // Inicio Mostrar Resultados,

```

```

Program.cs*  TP4_G12_Ej12_Consola_2
TP4_G12_Ej12_Consola_2.Program  Sub_Mostrar_Mensaje(string StrMensaje)

80 // SubRutina Mostrar Mensaje. --- Esta Sub simplemente muestra Mensajes ---
81 static void Sub_Mostrar_Mensaje(string StrMensaje)
82 {
83     Console.WriteLine(); // Presento la Consigna del problema
84     Console.WriteLine(StrMensaje);
85 }
86
87 static double NumReturn; //Valor de RETORNO de la función fuera de Módulos y Estático
88
89 // La función, recibe un string (El Mensaje a mostrar al Usuario) y retorna un DOBLE.
90 static double Fun_Obtener_Numero(string StrMensaje)
91 {
92     //Creamos una función para obtener los Coeficientes, y asegurarnos que sean NUMEROS
93     //Definimos las variables Locales
94     string StrNum; //Representa el Texto que ingreso el Usuario por Teclado
95     double Num; //Representa el Número Convertido a Doble
96     Boolean Bandera = true; //Definimos una variable Bool para manejar el Ciclo While
97
98     while (Bandera) //Como Bandera = True, el Ciclo es eterno hasta que cambie a False
99     {
100         // Inicio Ciclo While
101         Console.WriteLine(StrMensaje); //Muestra el Mensaje que se envió al llamar la Función
102         StrNum = Console.ReadLine(); //Recoge el Texto Tipeado por el Usuario
103
104         if ((Double.TryParse(StrNum, out Num)))...
105         else...
106     } // Fin Ciclo While
107     return NumReturn; //Estamos seguros que el Usuario Ingreso un Numero y lo Retornamos
108 } // Fin de la Función que SOLO me permitirá ingresar NUMEROS.

```

Mostramos Resultados de la Version Consola-2:

```
TP4_G12_Ej12_Consla-2 .#.

Este programa encuentra las raices REALES de una
Ecuación Cuadrática ingresando sus coeficientes.

Ingrese el Coeficiente del término Cuadrático A =
Ingrese el Coeficiente del término Cuadrático A =
Ingrese el Coeficiente del término Cuadrático A = 0.000
Ingrese el Coeficiente del término Lineal B = 1
Ingrese el Coeficiente del término Independiente C = 1

Coeficientes: A = 0 B = 1 y C = 1
Como A = 0, No hay Ecuación cuadrática, sino Lineal
El valor de la Solución1 es : X1 = - C/B = -1

Presione cualquier tecla para SALIR
```

```
TP4_G12_Ej12_Consla-2 .#.

Este programa encuentra las raices REALES de una
Ecuación Cuadrática ingresando sus coeficientes.

Ingrese el Coeficiente del término Cuadrático A =
Ingrese el Coeficiente del término Cuadrático A = 1,05
Ingrese el Coeficiente del término Lineal B = 3,333
Ingrese el Coeficiente del término Independiente C = 9,028

Coeficientes: A = 1,05 B = 3,333 y C = 9,028
El Discriminante de la Ecuación Cuadrática es NEGATIVO,
La Ecuación NO tiene soluciones Reales.

Presione cualquier tecla para SALIR
```

```
TP4_G12_Ej12_Consla-2 .#.

Este programa encuentra las raices REALES de una
Ecuación Cuadrática ingresando sus coeficientes.

Ingrese el Coeficiente del término Cuadrático A = -1,003
Ingrese el Coeficiente del término Lineal B = 10,10
Ingrese el Coeficiente del término Independiente C = 11,55

Las Raices Reales de la Ecuación Cuadrática
de Coeficientes: A = -1,003 B = 10,1 y C = 11,55
Son: X1 = -1,03681156808036 X2 = 11,106602196196

Presione cualquier tecla para SALIR
```

5. Codificación en VSC# Ventana. (Windows Form)

Etiqueta (Label) con consigna del problema.
 Este programa Encuentra las Raices Reales de la Ecuación Cuadrática: $f(x) = A \cdot x^2 + B \cdot x + C$

Ingreso de Coeficientes
CajaAgrupadora (GroupBox) para el Ingreso de los Datos
 El Coeficiente A =
 El Coeficiente B =
 El Coeficiente C =

Area de botones de Comando que realizarán las distintas tareas
 Ingresar Datos
 Calcular
 Limpiar
 Salir

Resultados
CajaAgrupadora (GroupBox) para Mostrar los Resultados
 Coeficiente A = 0, implica Ecuación Lineal y Solución Unica : X =
 Solución : X1 = Solución : X2 =
 Discriminante < 0, implica Solucion Compleja. NO se CALCULA en esta Versión.

Como siempre, la versión con Formulario de Windows, la iniciamos con el diseño del Front-End o interfaz para el Usuario.

- 1- Informamos la consigna del problema en una **Etiqueta** que aparece en el primer línea después de la barra de Título de la ventana.
- 2- Luego tenemos un **GroupBox "Ingreso de Coeficientes"**, para mostrar el valor de los Coeficientes ingresados con el **boton "Ingresar Datos"**.
- 3- A la derecha, tenemos el Area de Procesos o **"Botones de Comando"**, que son donde escribiremos el código para que realice las distintas tareas.
- 4- Por último tenemos un **GroupBox "Resultados"**, para mostrar los Mensajes generados por el Botón **"Calcular"** ante las distintas situaciones que se presenten según los Datos ingresados.

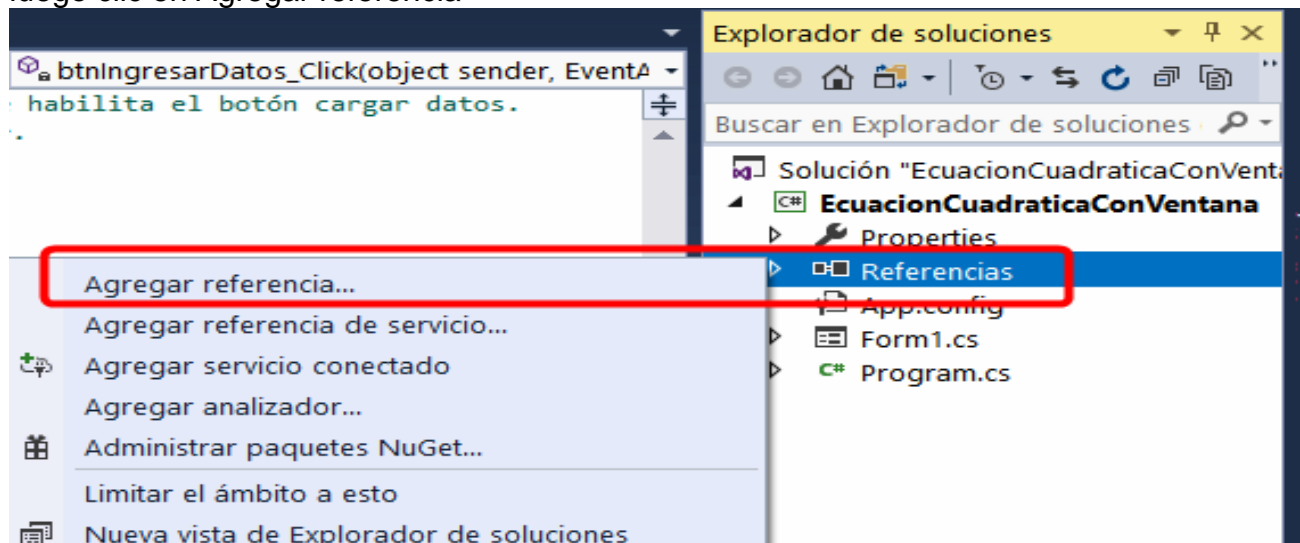
El código está comentado y no será difícil seguirlo.

Como hemos creado un Módulo (**Ingresar Datos**) No existen TextBox para la entrada de datos por parte del Usuario, para ello usamos una función Pre Definida en **Microsoft.VisualBasic.Interaction.InputBox()**, y que es simplemente una ventana para la carga de Datos por parte del Usuario (Ídem al InputBox de Raptor)

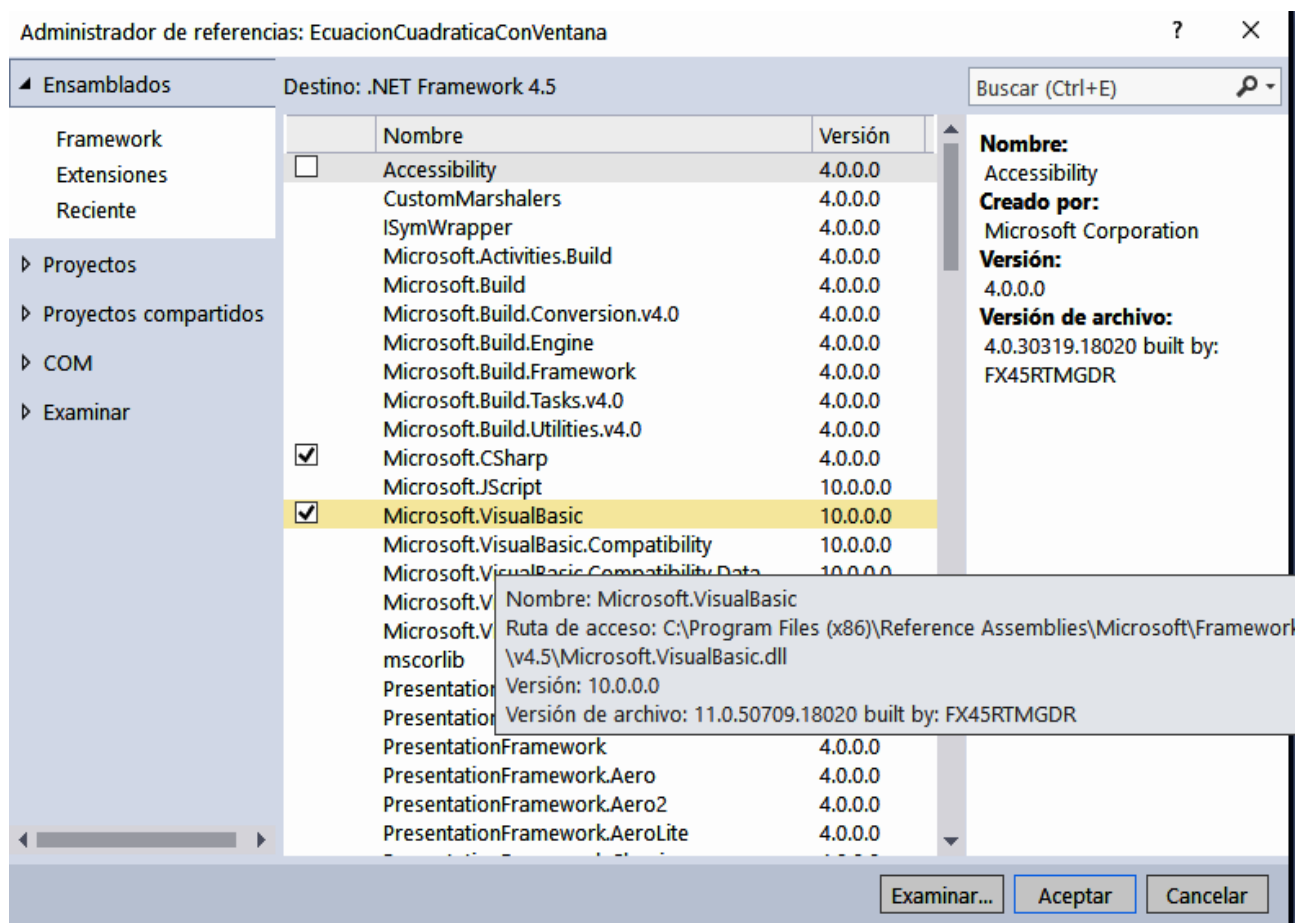
```
sNum = Microsoft.VisualBasic.Interaction.InputBox( funMensaje, "Ingresar Valor Numérico");
```

Como su Nombre lo indica, esta ventana retornará Texto, (Aunque ingrese números); por ello, debo verificar que el ingreso del Usuario sea texto que represente un Número y convertirlo a Número, y esto lo hacemos llamando a otra función predefinida en **Microsoft.VisualBasic**, pero ahora dentro del paquete **Information.IsNumeric(sNum)**

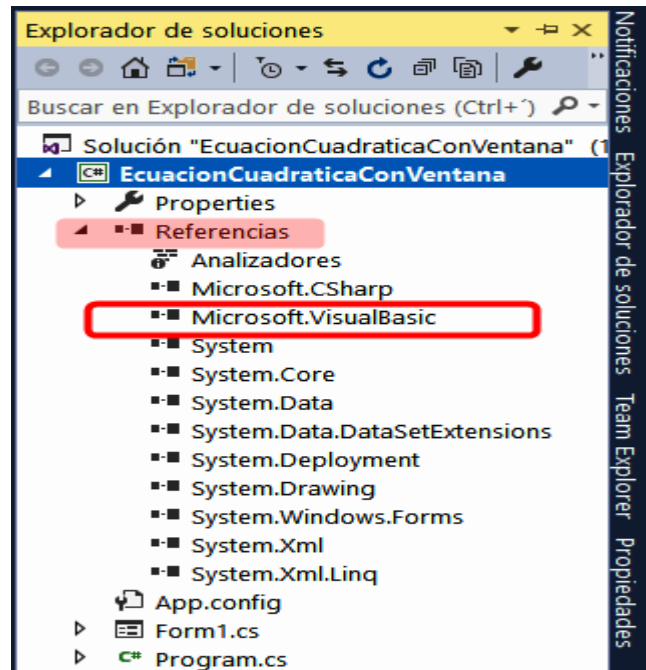
La parte mas importante del código de “Ingresar Datos” son los aspectos ya comentados, donde hacemos uso de funciones predefinidas en Visual Basic, y para poder usar estas funciones debemos ir al Explorador de Soluciones, hacer clic derecho en Referencias, y luego clic en Agregar referencia



Y luego tildar Microsoft.VisualBasic, para que quede como referencia de nuestro programa y poder usarlo en nuestro código.



Luego de esto, veremos que en el **Explorador de Soluciones** ya tenemos una **Referencia** a **Microsoft.VisualBasic**, y esto nos indica que ya podemos usar todas las funciones predefinidas dentro de ese lenguaje como si fueran propias del C#



Finalmente, la Función `static double funLeerNumero(string funMensaje)` puede verse a continuación:

```
// inicio una función que recibe un parámetro string y retorna un Número double
static double funLeerNumero(string funMensaje)
{ // inicio funLeerNumero()

    string sNum;    // la función define dos variables internas: una String
    double dNum;    // para cargar el valor introducido que teclado y otra para
                    // su valor numérico convertido a double.

    Do // inicio del ciclo HACER MIENTRAS sNum NO es Número.
    { // aquí usa el argumento strMensaje pasado al llamar la función.
      // y llama a Microsoft.VisualBasic para usar InputBox()
      sNum = Microsoft.VisualBasic.Interaction.InputBox( funMensaje,
        "... Ingresar Valor Numérico ..." );
      // aquí lee el valor introducido por el Usuario, que es string.
    }
    while ( ! Microsoft.VisualBasic.Information.IsNumeric ( sNum ) );
    // el ciclo continuo hasta que sNum sea un número, y cuando es SALE.
    dNum = Double.Parse(sNum); // Al salir convierte sNum a double.
    return dNum; // y retorna el valor al llamado de la función.

} // fin de funLeerNumero()
```

A continuación presentaremos algunas Ventanas de salida de la aplicación para los distintos casos:

Este programa Encuentra las Raices Reales de la Ecuación Cuadrática: $f(x) = A.x^2 + B.x + C$

Ingreso de Coeficientes

El Coeficiente A =

El Coeficiente B =

El Coeficiente C =

Ingresar Datos

Calcular

Limpiar

Salir

Resultados

Como vemos la opción preseleccionada es **<Ingresar Datos>** que es el módulo que permitirá cargar los coeficientes de la ecuación, controlando que estos sean solo valores Numéricos (Carga VALIDADA de Datos)

Ingresar Valor Numérico

Ingrese el Valor Numérico del Coeficiente A:

Aceptar

Cancelar

KHKJALGHJAH

Esta ventana de ingreso de Datos solo aceptará valores Numéricos (Valida el Ingreso)

Ventanas de Salida para los distintos coeficientes ingresados:

Este programa Encuentra las Raices Reales de la Ecuación Cuadrática: $f(x) = A.x^2 + B.x + C$

Ingreso de Coeficientes

El Coeficiente A = 0

El Coeficiente B = 5,55678

El Coeficiente C = -3,00987

Ingresar Datos

Calcular

Limpiar

Salir

Resultados

Coeficiente A = 0, implica Ecuación Lineal y Solución Unica : $X = 0,541657218748988$

.. Ecuación Cuadrática ..

Este programa Encuentra las Raices Reales de la Ecuación Cuadrática: $f(x) = A.x^2 + B.x + C$

Ingreso de Coeficientes

El Coeficiente A = -5,555

El Coeficiente B = 3

El Coeficiente C = 0,0897

Botones: Ingresar Datos, Calcular, Limpiar, Salir

Resultados

Solución : X1 = -0,028405899220359 Solución : X2 = 0,568459904620

.. Ecuación Cuadrática ..

Este programa Encuentra las Raices Reales de la Ecuación Cuadrática: $f(x) = A.x^2 + B.x + C$

Ingreso de Coeficientes

El Coeficiente A = 1,111

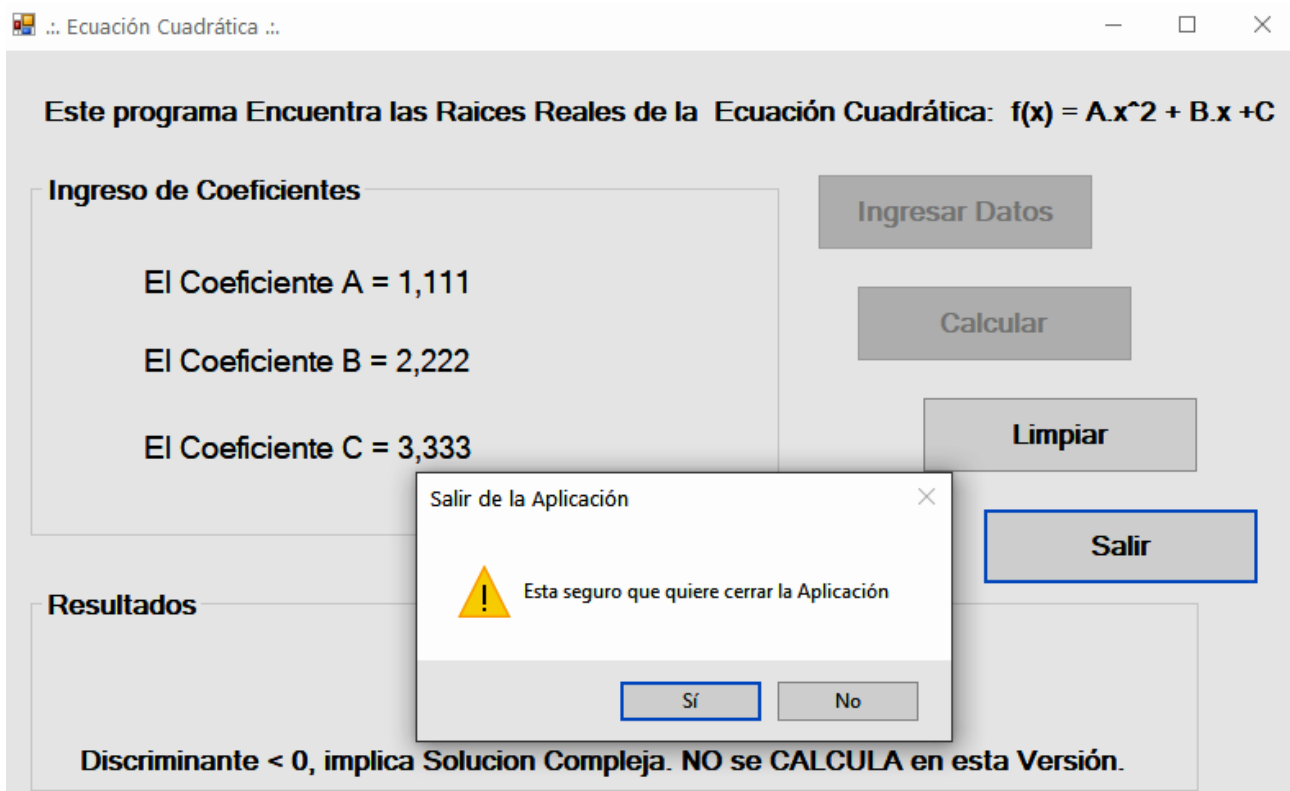
El Coeficiente B = 2,222

El Coeficiente C = 3,333

Botones: Ingresar Datos, Calcular, Limpiar, Salir

Resultados

Discriminante < 0, implica Solucion Compleja. NO se CALCULA en esta Versión.



Con esta pantalla final, ya podemos salir de la aplicación.