UMSA
FACULTAD DE
INGENIERIA

Materia:

Laboratorio de programación
"ETN307"

Tema:

Algoritmos en C++ y PSeInt

Vol. 1

Nombre:
Univ. Mijahel Alexander
Ruelas Machicado

Laboratorio #1

Algoritmos en C++ y PSeInt

Objetivos. –

Implementar el uso de algoritmos en el lenguaje de Dev-C++ y de PSeInt para la resolución de los siguientes problemas matemáticos tipo:

- Series de Taylor y Maclaurin
- Ecuaciones de 2do grado
- Ecuaciones de 3er grado

- **SERIES DE TAYLOR Y MACLAURIN**

Marco teórico. –

La serie de Taylor de una función real o compleja f(x) infinitamente diferenciable en el entorno de un número real o complejo *a* es la siguiente serie de potencias:

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \cdots$$

donde n! denota el factorial de n. Utilizando la notación sigma, lo anterior puede ser escrito de manera compacta como

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

donde $f^{(n)}(a)$ denota la n-ésima derivada de evaluada en el punto a.

En particular, cuando a=0 , la serie es denominada **Serie de Maclaurin**.

- **Función analítica.**

  Si f(x) está dada por una serie de potencias convergente en un disco abierto centrada en b en el plano complejo entonces se dice que es analítica en el disco, por lo que para x en este disco, f está dada por la serie de potencia convergente

$$f(x) = \sum_{n=0}^{\infty} a_n (x-b)^n$$

derivando con respecto a x la fórmula anterior n veces y evaluando x=b obtenemos

$$\frac{f^{(n)}(b)}{n!} = a_n$$

y en tal caso, la expansión en series de potencia coincide con la serie de Taylor. Por lo tanto, una función es analítica en un disco abierto centrado en b si y sólo si su serie de Taylor converge al valor de la función en cada punto en el disco.

- **Series utilizadas en el algoritmo.**

  Para este punto se dividirán en 2 tipos de series

  a) Convergente para toda x

     i. Función exponencial

     Sea la función exponencial e^x , que tiene como serie:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

     ii. Funciones trigonométricas

     Las funciones usadas serán las del seno y la del coseno, cuyas series son:

$$\operatorname{sen} x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \quad \text{para toda } x$$

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \quad \text{para toda } x$$

  b) Convergente para un rango de x

     iii. Logaritmo natural

     El logaritmo natural tiene cono serie de Taylor:

$$\ln(x) = 2\sum_{n=0}^{\infty} \frac{1}{2n+1} \left(\frac{x-1}{x+1}\right)^{2n+1}$$

     Converge para 0<x<1.

iv. Función geométrica

La serie geométrica simple tiene como serie:

$$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$$

Converge para -1<x<1.

v. Función binomial

La serie binomial es la serie de potencias:

$$(1+x)^{\alpha} = \sum_{n=0}^{\infty} \binom{\alpha}{n} x^n$$

Cuyos coeficientes son los coeficientes binomiales generalizados

$$\binom{\alpha}{n} = \prod_{k=1}^{n} \frac{\alpha - k + 1}{k} = \frac{\alpha(\alpha-1)\cdots(\alpha-n+1)}{n!}$$

Converge para -1<x<1 para cualquier valor de alpha.

❖ Ejemplos prácticos. –

- Función exponencial

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{x!}$$

Sean los puntos de convergencia:

$$z_0 = \{\ 0\ ,\quad -2\ ,\ 5\ \}$$

$$e^0 = 1$$

$$e^{-2} = 0{,}1353352835$$

$$e^5 = 148{,}4131591$$

- Funciones trigonométricas
  - Seno:

$$\operatorname{sen} x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}\, x^{2n+1}$$

Sean los puntos de convergencia:

$$z_0 = \{\ 0\ ,\ 45\ ,\ 80\ \}$$
$$sen(0) = 0$$
$$sen(45) = 0{,}707106781$$
$$sen(80) = 0{,}984807753$$

  - Coseno:

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!}\, x^{2n}$$

Sean los puntos de convergencia:

$$z_0 = \{\ 0\ ,\ 15\ ,\ 42\ \}$$
$$cos(0) = 1$$
$$cos(15) = 0{,}965925826$$
$$cos(42) = 0{,}7431448255$$

- Logaritmo natural

$$\ln(x) = 2\sum_{n=0}^{\infty} \frac{1}{2n+1}\left(\frac{x-1}{x+1}\right)^{2n+1}$$

Sean los puntos de convergencia (0 < z_0 < 1)

$z_0 = \{\ 0{,}2\ ,\ 0{,}5\ ,\ 0{,}9\ \}$

$$Ln(0{,}2) = -1{,}609437912$$

$$Ln(0{,}5) = -0{,}6931471806$$

$$Ln(0{,}9) = -0{,}1053605157$$

- Función geométrica

$$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$$

Sean los puntos de convergencia (0 < z_0 < 1)

$z_0 = \{\ 0{,}1\ ,\ 0{,}6\ ,\ 0{,}8\ \}$

$$(1-0{,}1)^{-1} = 1{,}\overline{1111}$$

$$(1-0{,}6)^{-1} = 2{,}5$$

$$(1-0{,}8)^{-1} = 5$$

- Función binomial

$$(1+x)^{\alpha} = \sum_{n=0}^{\infty} \binom{\alpha}{n} x^n$$

Sean los puntos de convergencia (-1 < z_0 < 1)

$z_0 = \{ -0{,}7 \ , -0{,}123 \ , \ 0{,}9 \ \}$

$\alpha = \{2 \ , \ 4 \ , \ 7\}$

$(1 - 0{,}7)^2 = 0{,}09$

$(1 - 0{,}123)^4 = 0{,}5915594186$

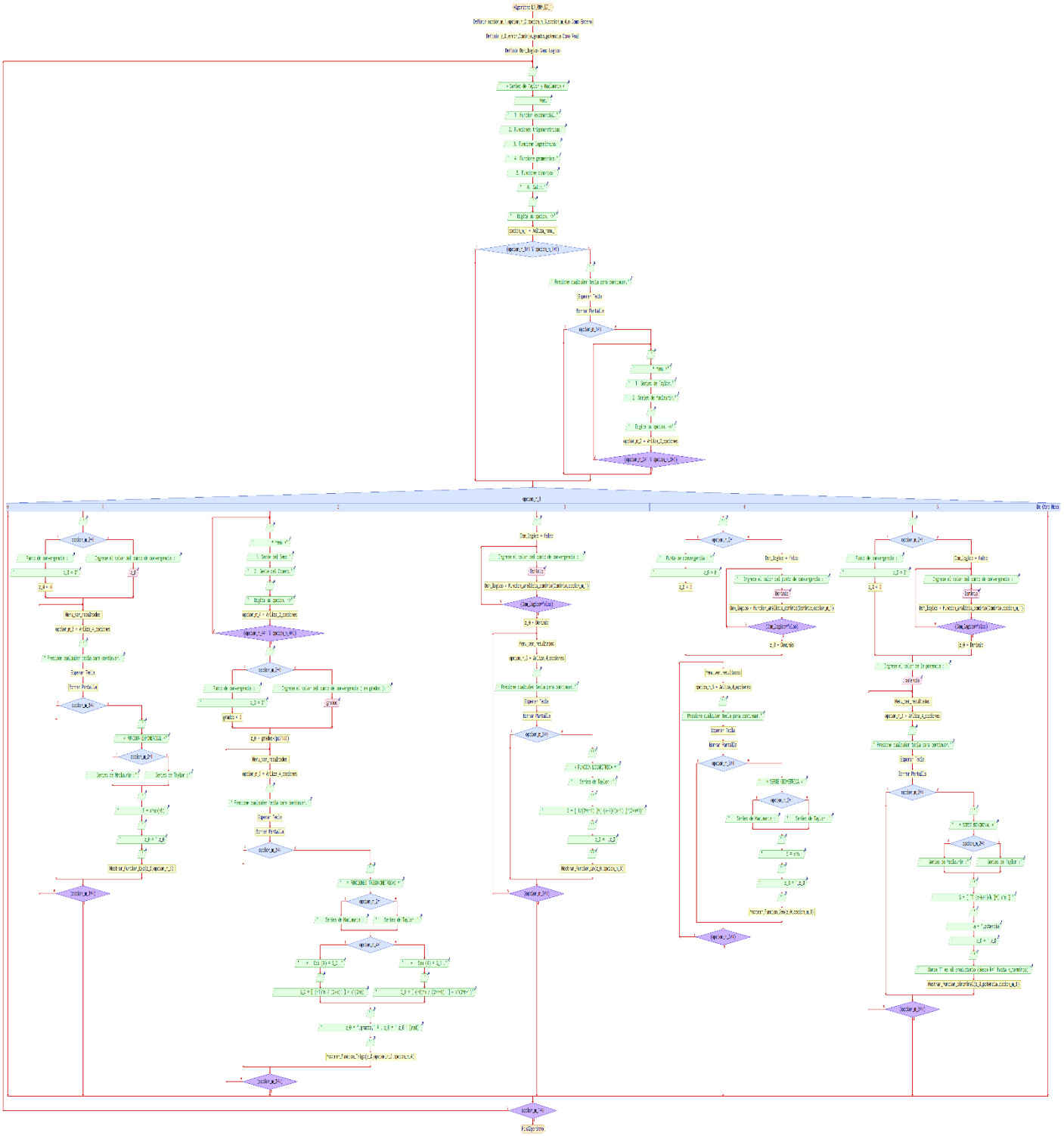
$(1 + 0{,}999)^7 = 127{,}5526714$

Diagramas de flujo. –

Para dibujar los diagramas de flujo de los algoritmos se empleó el programa “PSeInt”, y como estos están conformados por funciones se muestran de la siguiente manera; Además para poder visualizar de manera detallada se realizarán distintos acercamientos a cada diagrama de flujo.

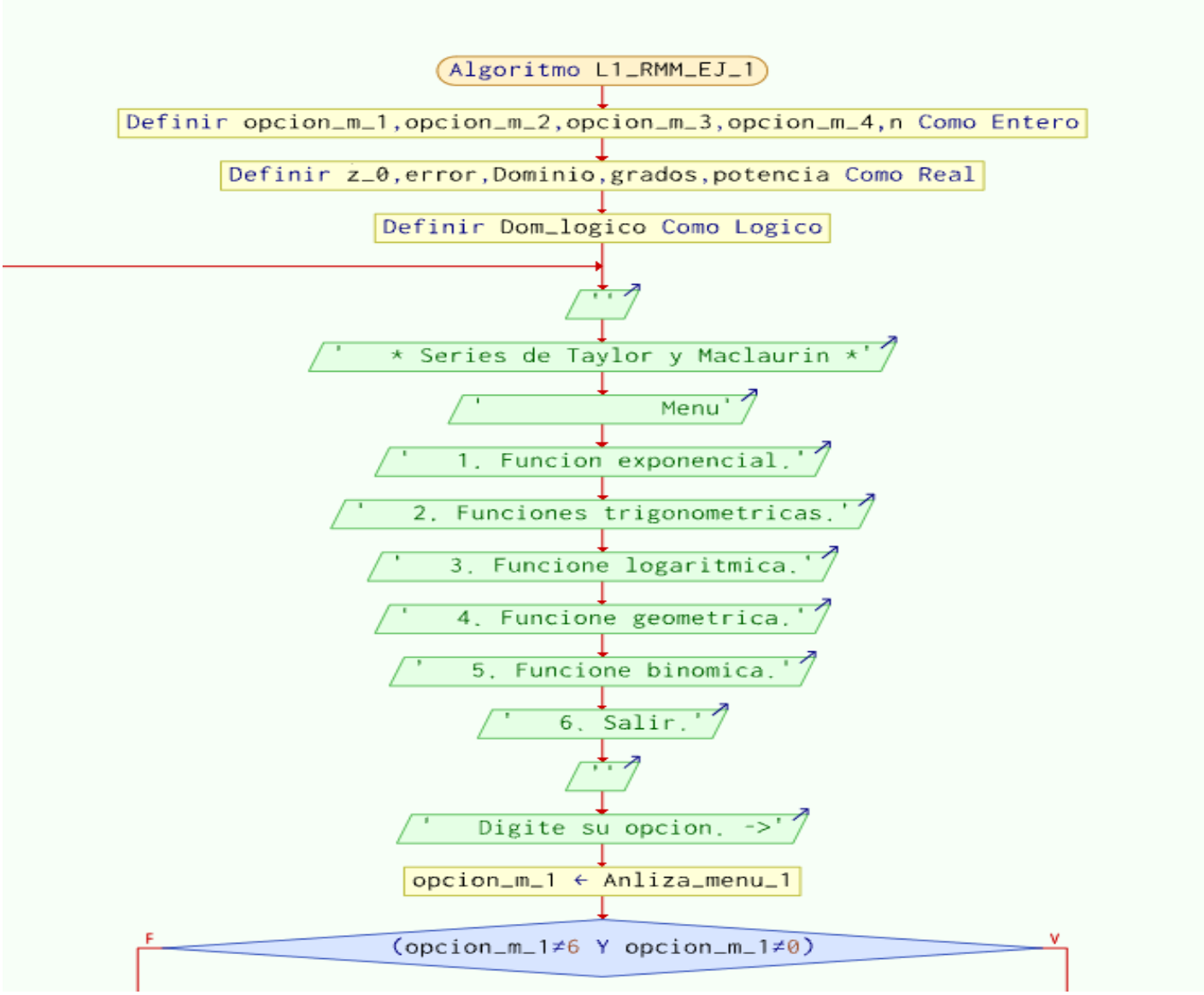
- Series de Taylor y Maclaurin (L1_RMM_EJ_1 es el nombre del archivo)

- Diagrama principal



Zoom programa principal

- Inicio del algoritmo: el algoritmo se llama L1_RMM_EJ_1
- Definir variables: se observa las de tipo entero que servirán para el manejo de las opciones del menú y las del tipo real que constarían de valores decimales, estos se utilizaron para el desarrollo matemático del programa y finalmente una variable del tipo lógico que se usó para una sección en la que existe un radio del disco de convergencia, estos se verán en las series del logaritmo natural, serie geométrica y la serie binominal.
- Se inicia un ciclo do{}while() que termina si y solo si el usuario digita la opción 6
- Impresión en la pantalla: Muestra el menú de inicio y pide ingresar su opción.
- Analiza_menu_1: Es un subprograma que evita que el usuario ingrese una opción errónea

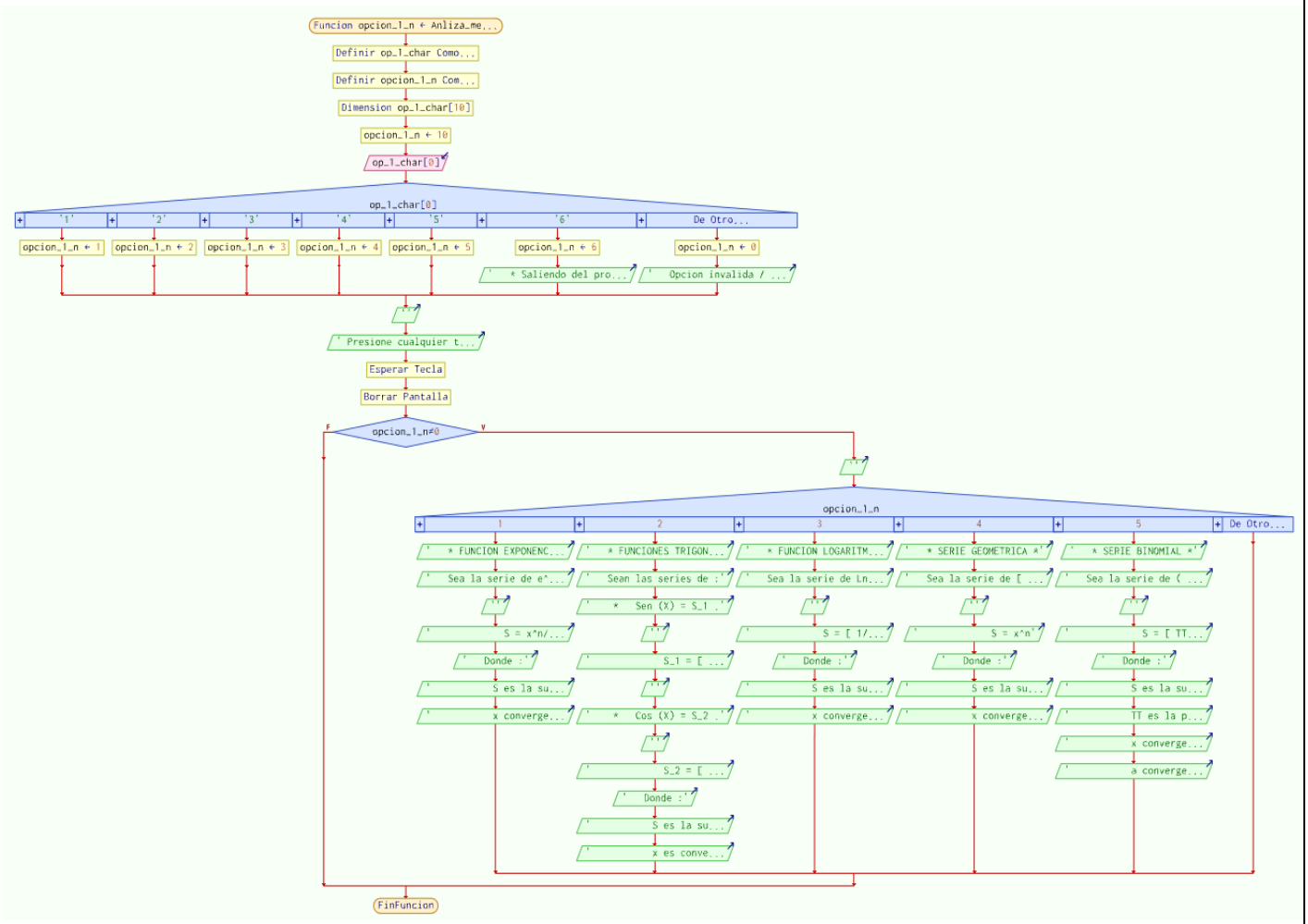


A. Subprograma Analiza_menu_1

- **Para que sirve.** -El usuario digitará algún valor, y esta será almacenada en el carácter opción_m_1(se creó dicho carácter para que el usuario ingrese cualquier letra, numero o simplemente un espacio), esto con el propósito de evaluar el valor ingresado en una condicional switch, donde se verá que para los valores validos (existentes en el menú) tomen un valor numérico entero en la variable opción_1_n (esto se apreciara y detallara más adelante), sin embargo si la opción ingresada es invalida entrara en el bloque de DEFAULT, y por consiguiente opción_n_1 tomara el valor de cero además que en la pantalla se imprimirá Valor invalido/Inténtelo nuevamente; También se debe aclarar que para la opción de 6 (la opción de salir del programa esto en el menú), es imprime * Saliendo del programa *.

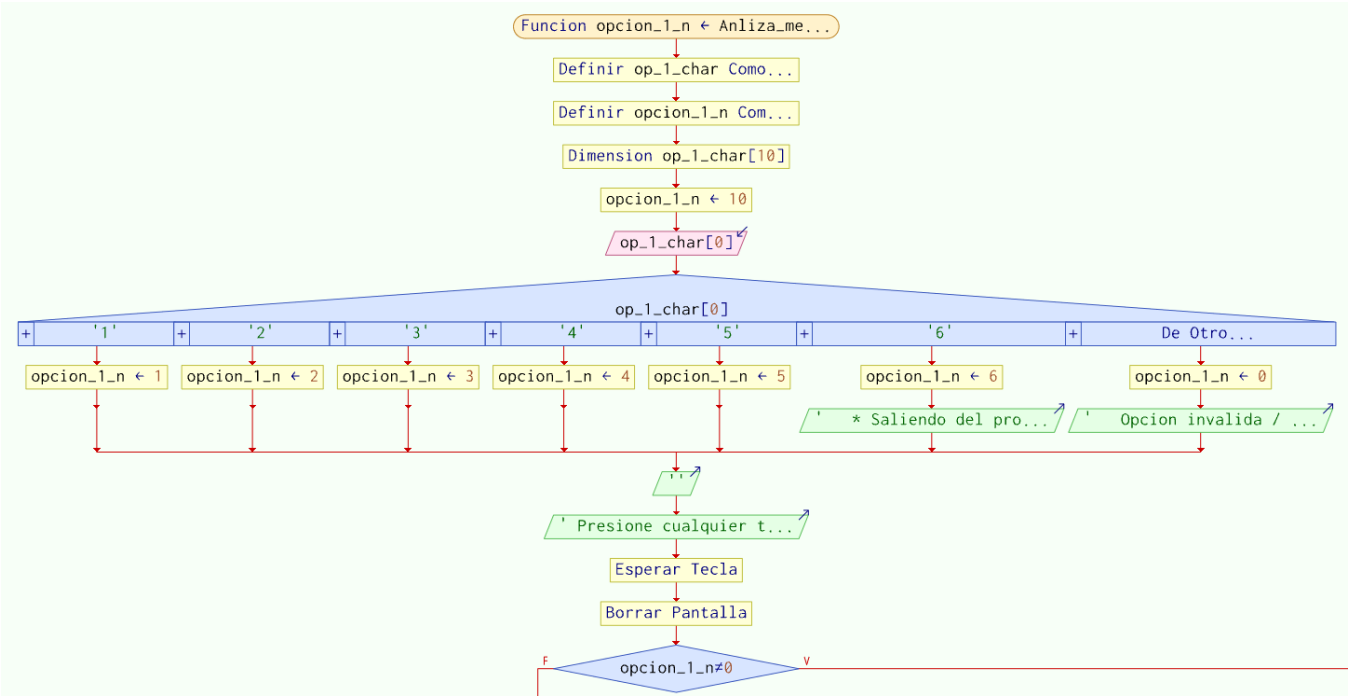
  Luego de evaluar y asignar un valor ya sea erróneo o valido a la variable opción_1_n, es imprime el texto de Presione cualquier tecla para continuar, al efectuar dicha acción se procede a limpiar todo lo mostrado en pantalla anteriormente.

  En la 2da mitad se ingresa el valor guardado de opción_1_n para ver si cumple o no con la condicional if() que evaluara dicho valor, si el valor fuera '0' finalizaría el subprograma, caso contrario si este fuera diferente de '0' ingresara a una condicional switch(esta dependiente del valor que tiene opción_n_1) que mostrara en la pantalla un breve resumen de la serie que el usuario digito, finalmente luego de realizar dicha acción el subprograma termina.

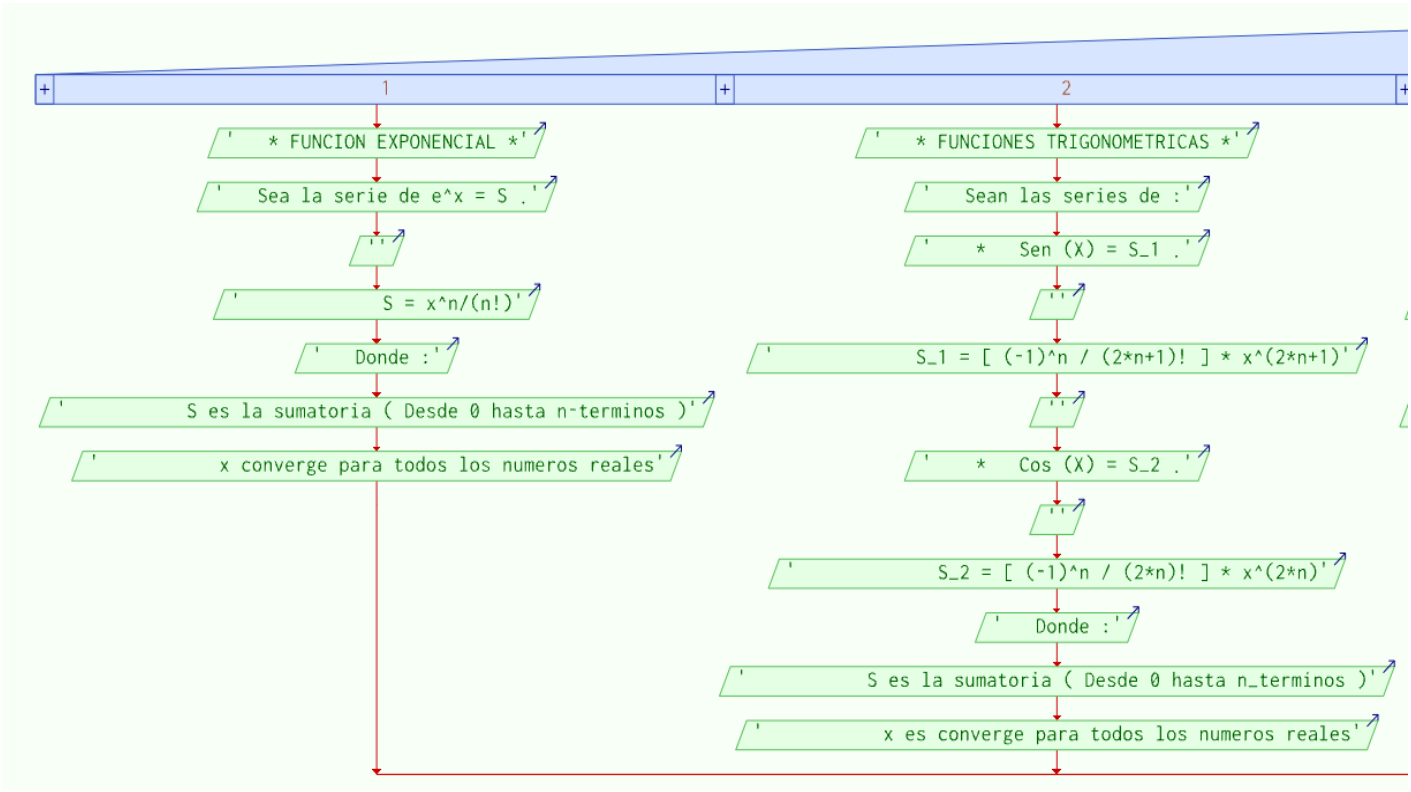


Zoom_sub_programa_Analiza_menu_1

- El subprograma se llama Analiza_menu_1
- Es necesario definir nuevas variables que solo trabajan dentro de dicho subprograma
- Es necesario aclarar que esta función o subprograma, es del tipo int en lenguaje de C++, es decir que cuando finalice enviara un valor al programa principal (En este caso en particular el valor servirá para activar las tareas y subprogramas que el usuario solicito).



zoom a la 2ª condicional switch () si el valor de opción_1 igual a la unidad o a 2.

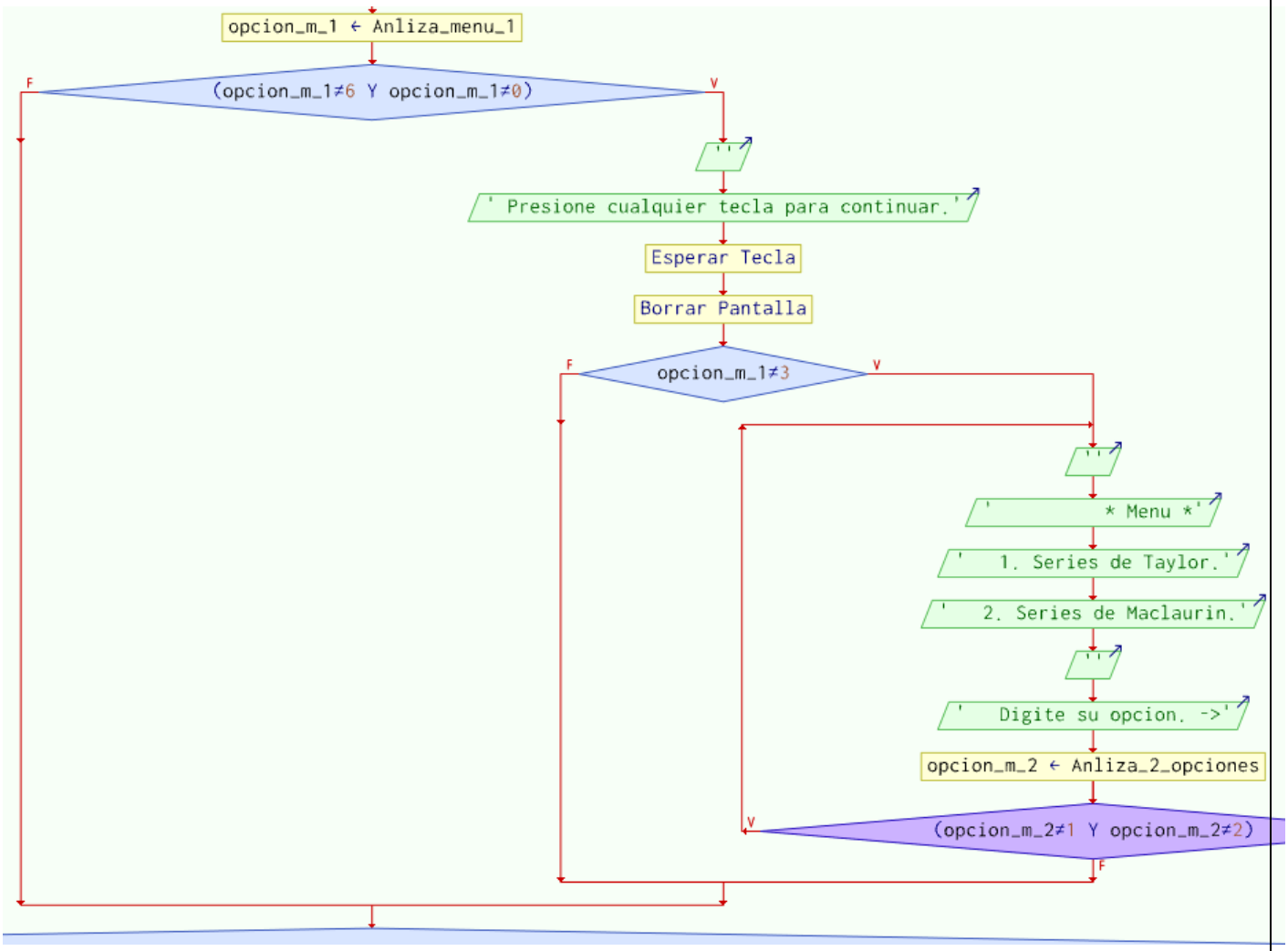


Retornando a la función principal:

Se ve que al valor retornado del subprograma Analiza_menu_1 es guardado en la variable opción_1_m y a continuación esta es evaluada en una condicional if(), si el usuario digitara la opción de salir o cometiera el error de ingresar un valor no valido el algoritmo pasara directamente a la siguiente sección del programa, la cual se trata de una condicional switch(); Sin embargo si el valor digitado por el usuario es correcto, el algoritmo imprimirá Presione cualquier tecla para continuar a continuación se procede a limpiar la pantalla para luego ingresar a un if().

Se creó esta condicional if() para evitar mostrar el menú 2 que se ve en la imagen, el cual propone entre 2 opciones al usuario, elegir si desea preceder con una serie de Taylor o una de Maclaurin, sin embargo para la opción 3 es decir para la Función del Logaritmo natural realizar una serie de Maclaurin (punto de convergencia igual a cero) sería imposible. Continuando con el análisis de la 2da condicional de la imagen, si resulta opción_m_1 distinta de cero como verdadero, se procede a mostrar en la pantalla el 2do menú que dará a elegir al usuario entre 2 opciones para luego ingresar a un subprograma llamado Analiza_2_opciones.

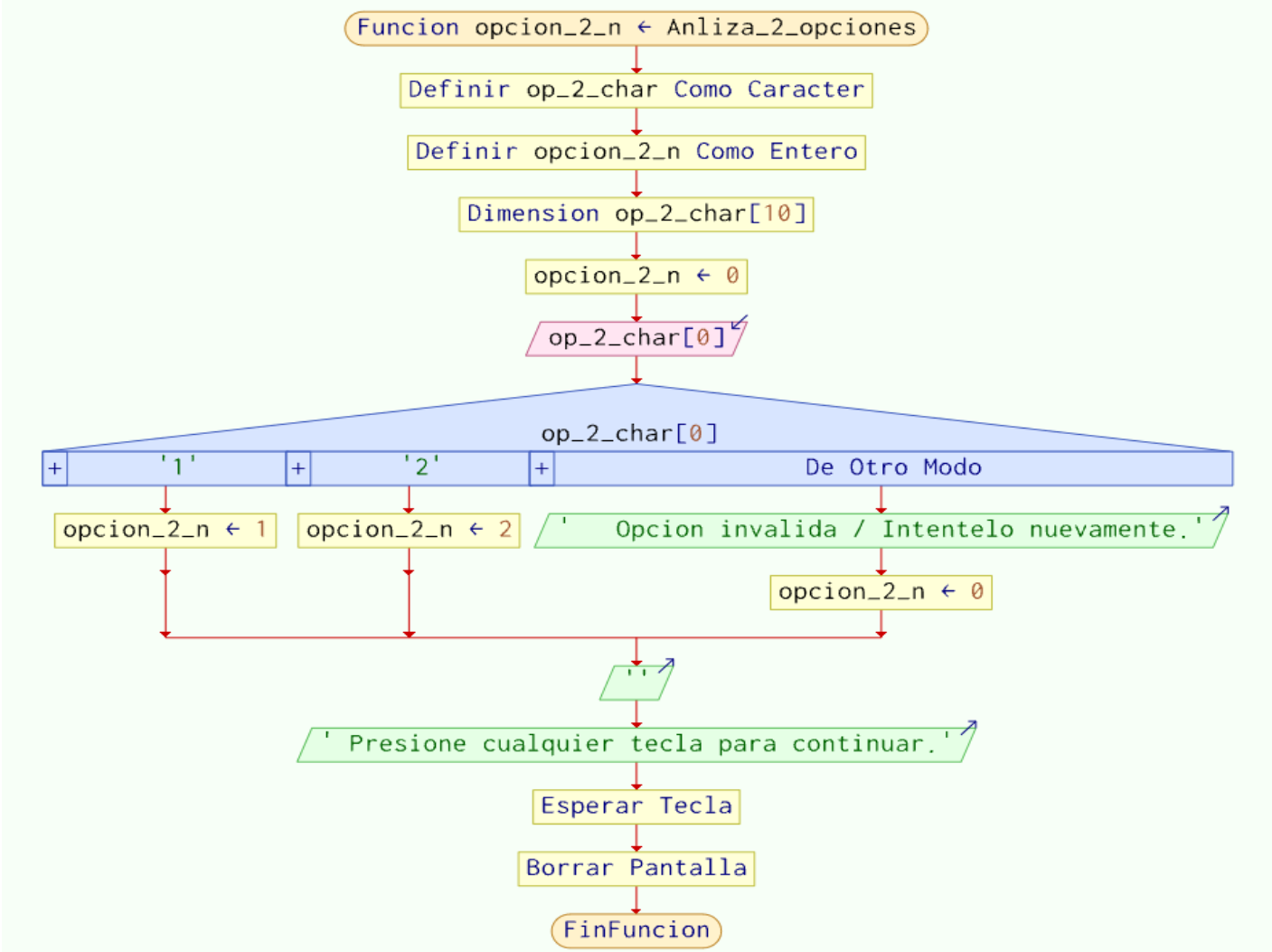
Cuando el subprograma Analiza_2_opcion finaliza y entrega un valor, el cual es almacenado en la variable tipo int llamada opción_m_2, la cual será analizada por el ciclo do{}while() de la imagen, dicha condicional repetirá el proceso hasta que el usuario ingrese un valor correcto del menú.



Subprograma Analiza_2_ociones

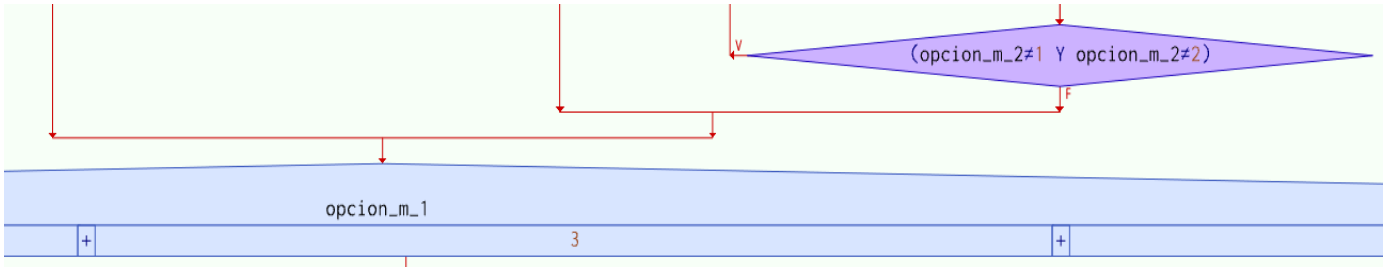
Para que sirve. –Similar al subprograma Analiza_menu_1, este realiza la tarea de evitar que el usuario ingrese valores erróneos al momento de escoger entre las opciones del menú 2.

- Esta función es del tipo int, retornando un valor numérico entero cuando termina de realizar su tarea, este valor numérico que servirá para los procesos posteriores del algoritmo.

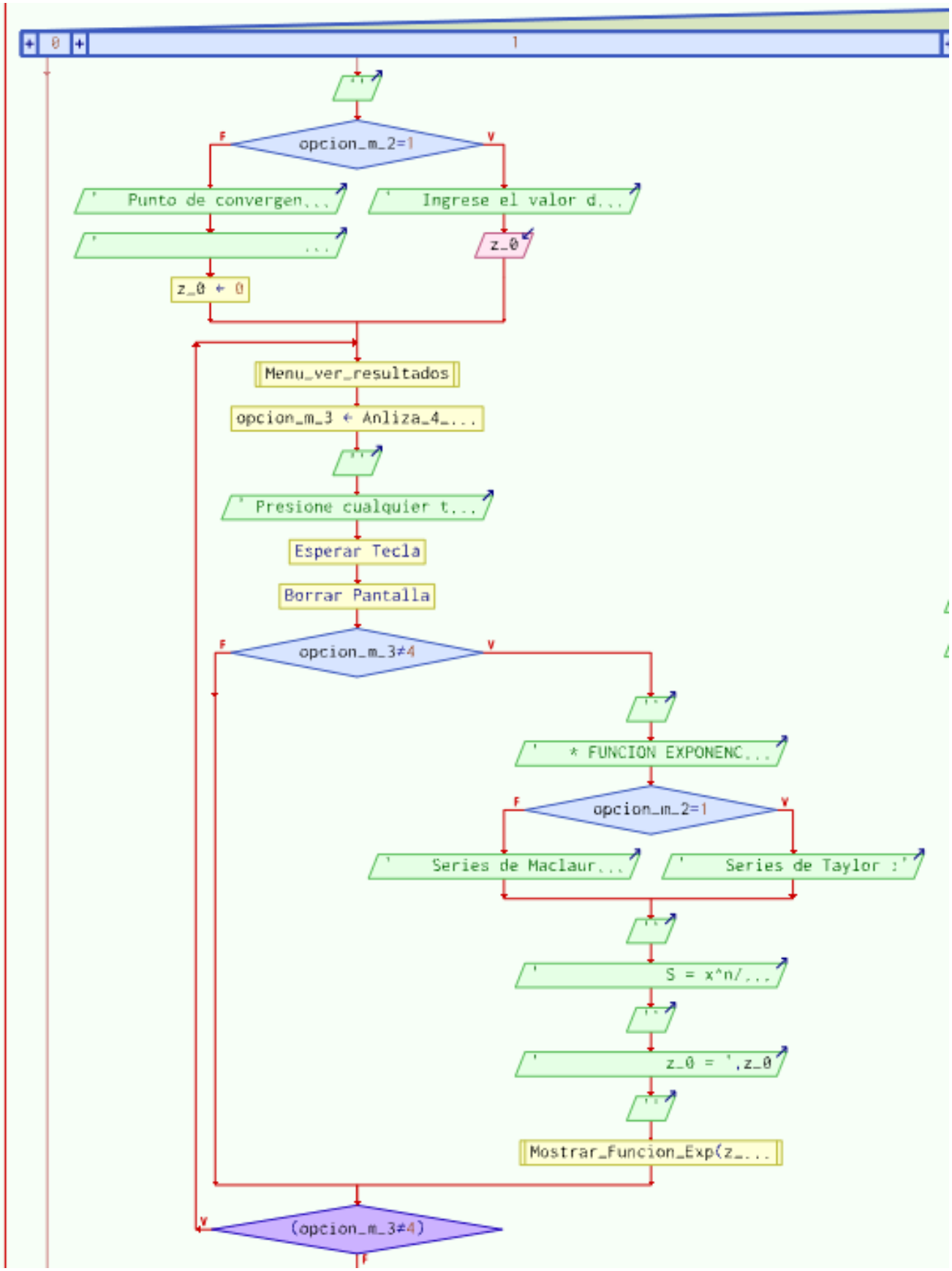


Retornando a la función principal:

El algoritmo ingresa a una condicional switch que será visto por partes puesto que este realizará las tareas para cada serie ya sea exponencial, geométrica, etc.

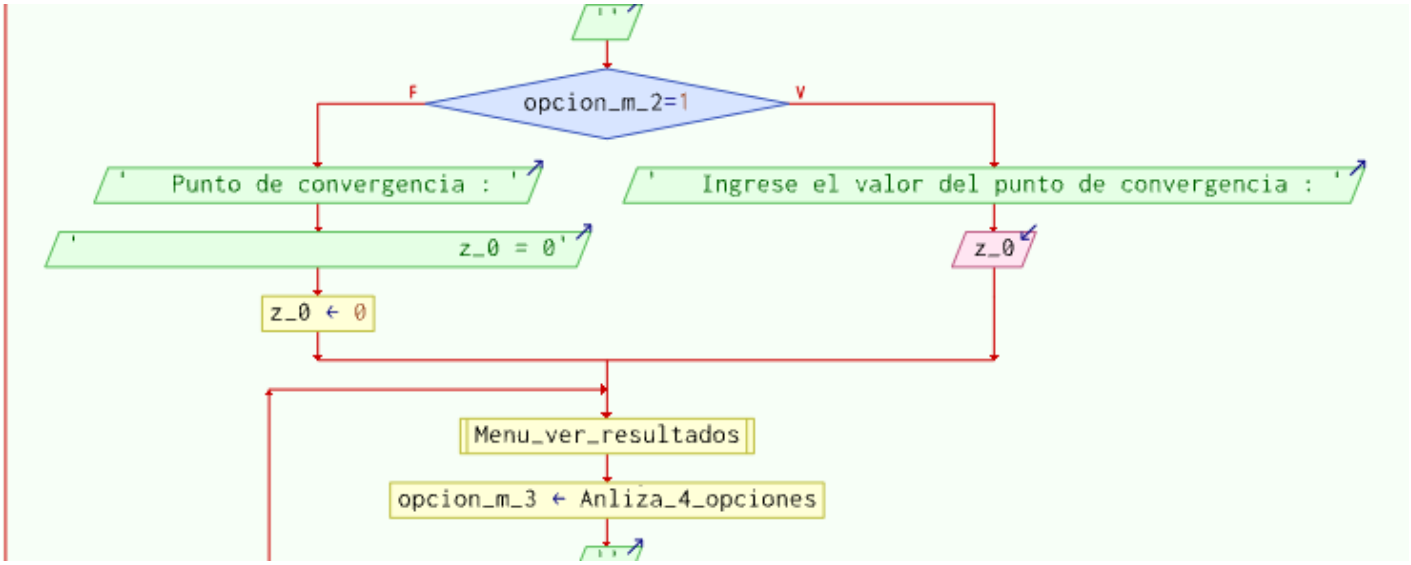


- Si opción_m_1 tiene los valores de 0 o 1.



Para el caso de opción_m_1 fuera igual a cero llegaría a dar por terminado la condicional switch sin realizar ninguna acción; sin embargo, si el valor fuera igual a la unidad finalmente empecería a procesarse las tareas necesarias para realizar la serie exponencial, como se explicará a continuación.

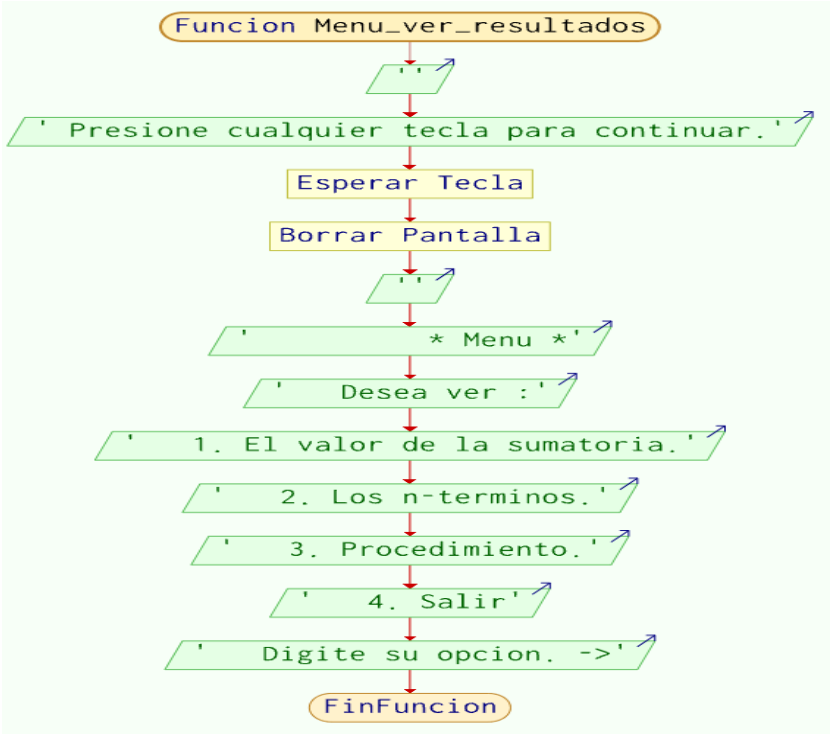
Bloque de tareas para la Serie exponencial 1ra parte:



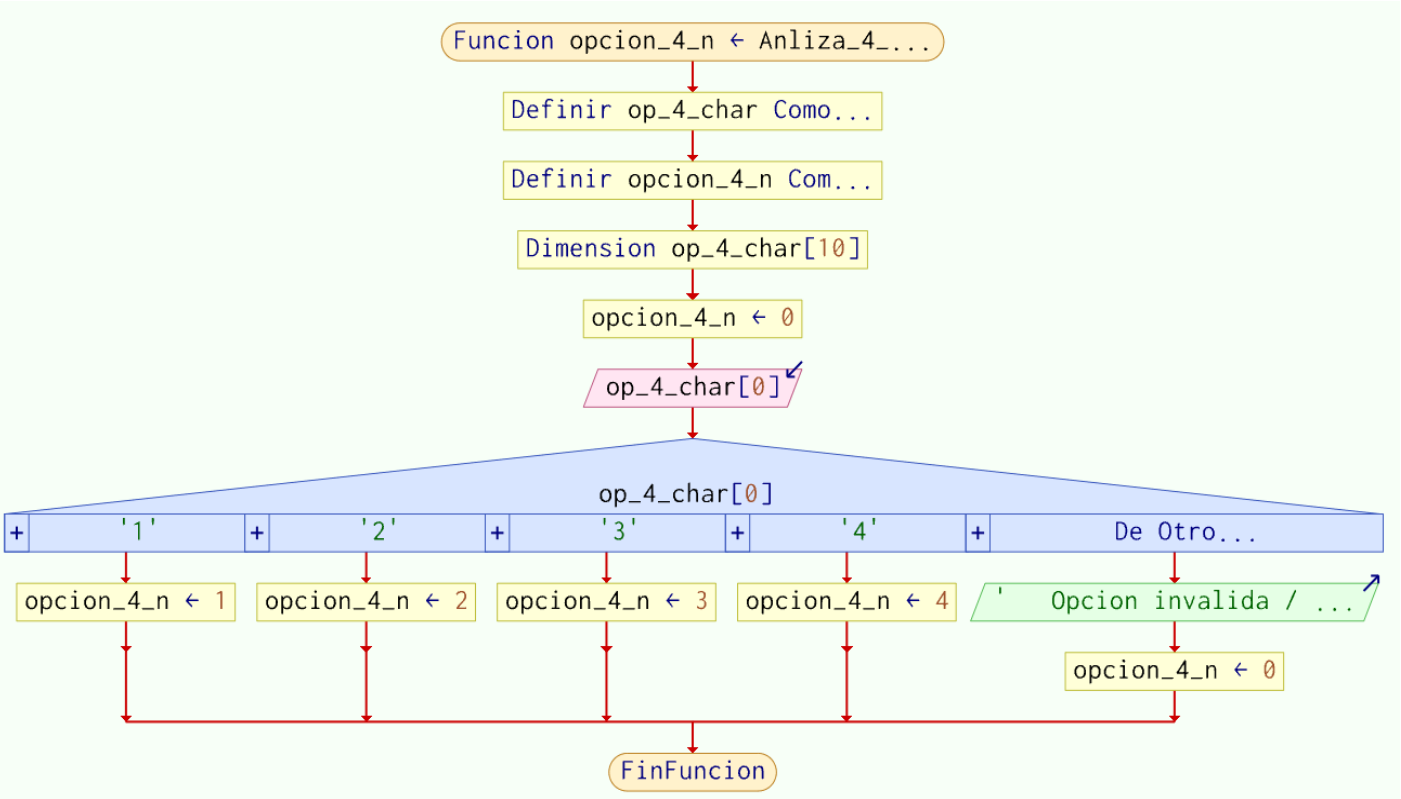
El bloque inicia con la con la condicional if() del valor de opción_m_2 el cual correspondía a si se deseaba una serie de Taylor o una de Maclaurin, para el caso de serie de Taylor mostrada en la pantalla que ingrese el valor del punto de convergencia para luego guardar el valor en la variable de tipo flotante z_0; Pero si se deseara realizar una serie de Maclaurin es valor por defecto es de cero.

Luego se inicia un ciclo do{} while() que evitara que el menú que veremos continuación llegara acerrare sin que el usuario así lo deseara.

Subprograma Menu_ver_resultados :

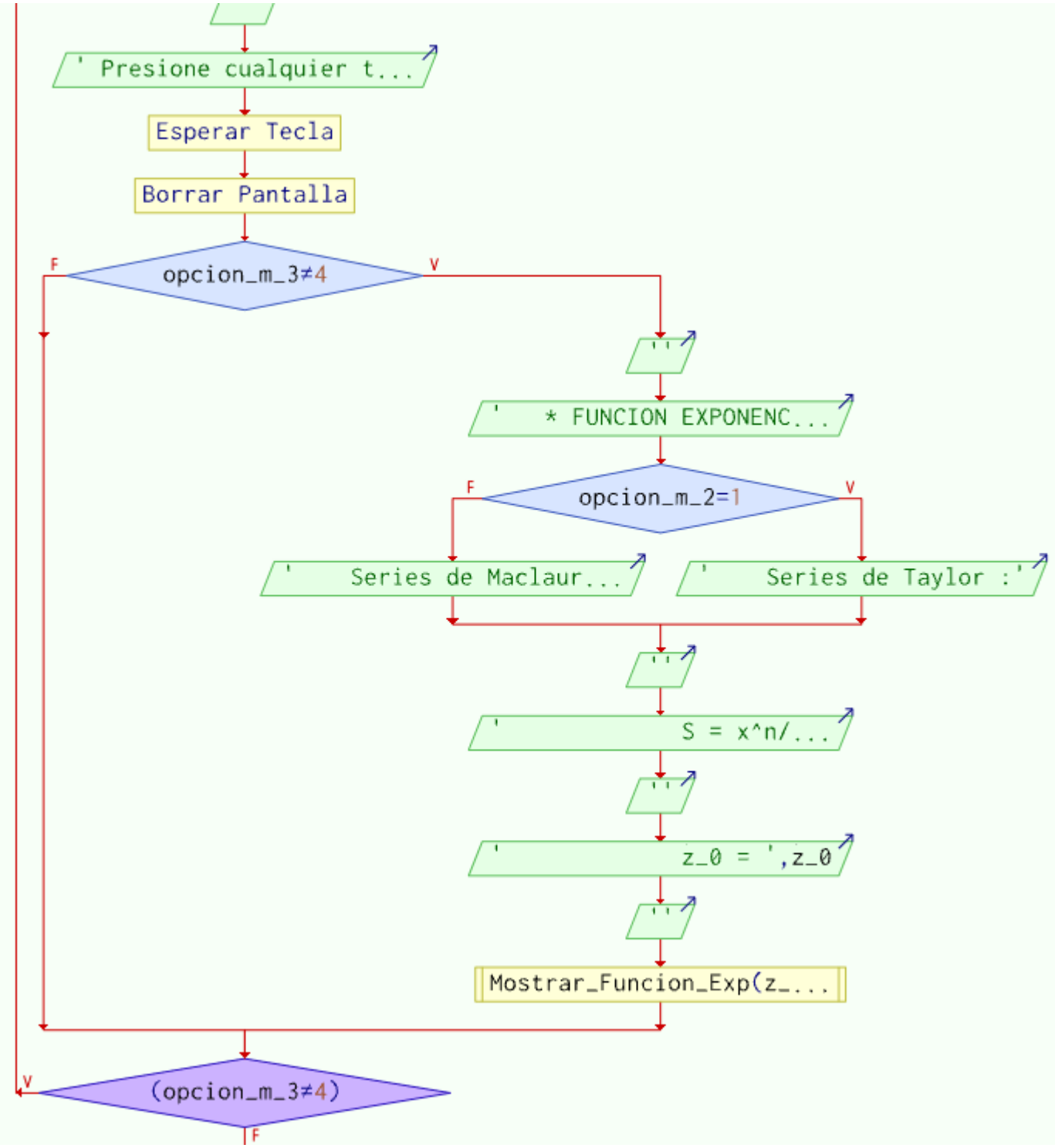


Subprograma Analiza_4_opciones:



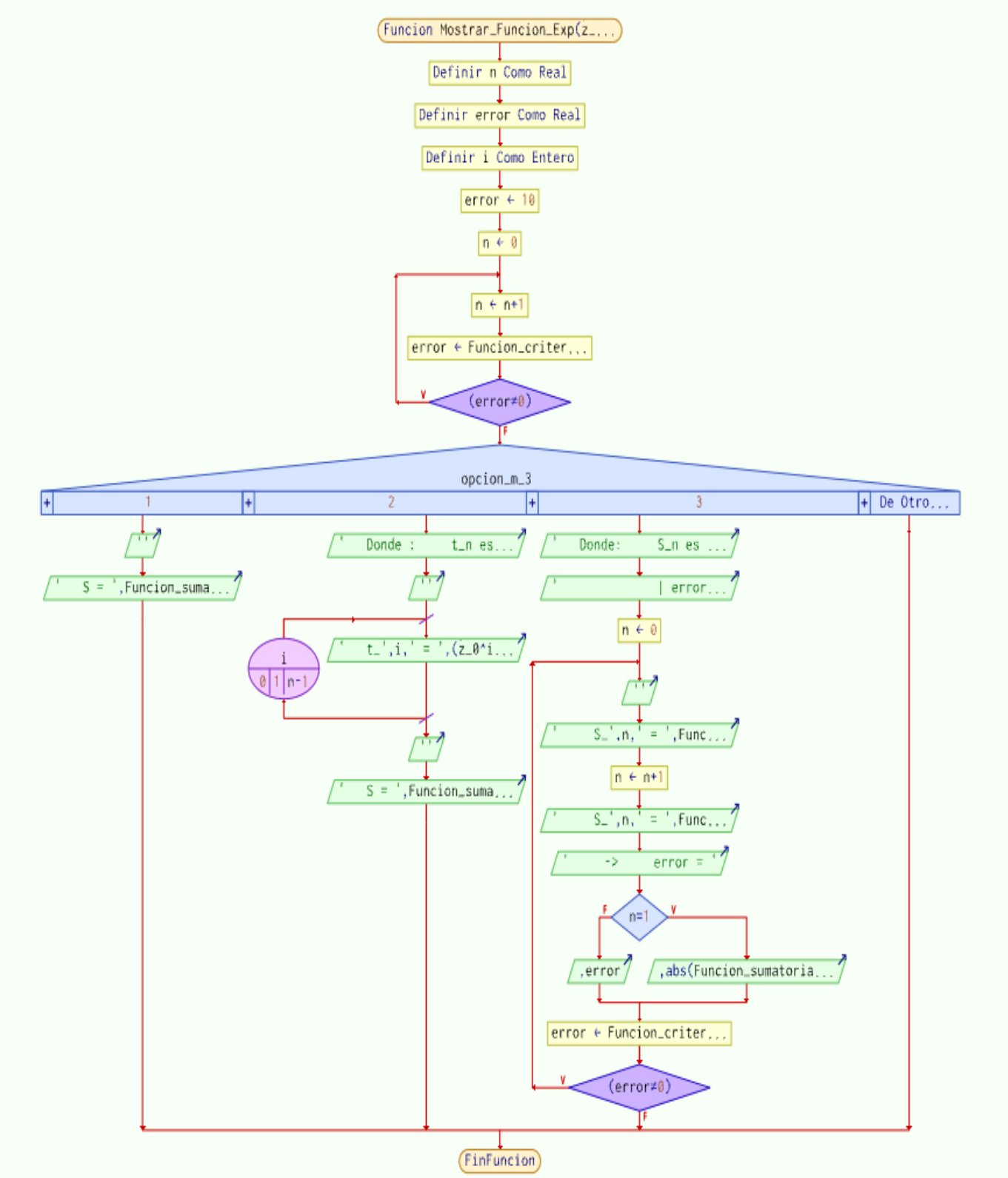
- Como se aprecia en los diagramas de flujo el subprograma Menu_ver_resultados realizara la tarea simple de mostrar en la pantalla los mensajes del 3er menú (ver la página anterior).
- Para el caso del subprograma Analiza_4_opciones funciona al igual que los anteriores, es decir para que el usuario elija entre 4 opciones, si este valor ingresado es válido se convierte en un valor numerito para la variable tipo int opción_4_n; Pero si el valor fuera erróneo el valor de la variable opción_4_n será de 0 y gracias al ciclo do{}while() este proceso se repetirá hasta que se digite la opción de salir.

Bloque de tareas para la Serie exponencial 2da parte:



La segunda parte del bloque inicia borrando lo que antes se encontraba en la pantalla de la consola, luego inicia una condicional if() evaluando el valor de la opción ingresada si el valor fuera igual a 4 se saltaría a la parte final del condicional do{}while() realizado en la 1ra parte, esto visto desde la consola se traduciría en que el menú de mostrar resultados se repitiera hasta que el usuario digite el valor de la opción de salida el cual es opción_m_3=4.

Si la opción seleccionada fuera distinta de 4 se mostrara los mensajes como se puede ver en la imagen.



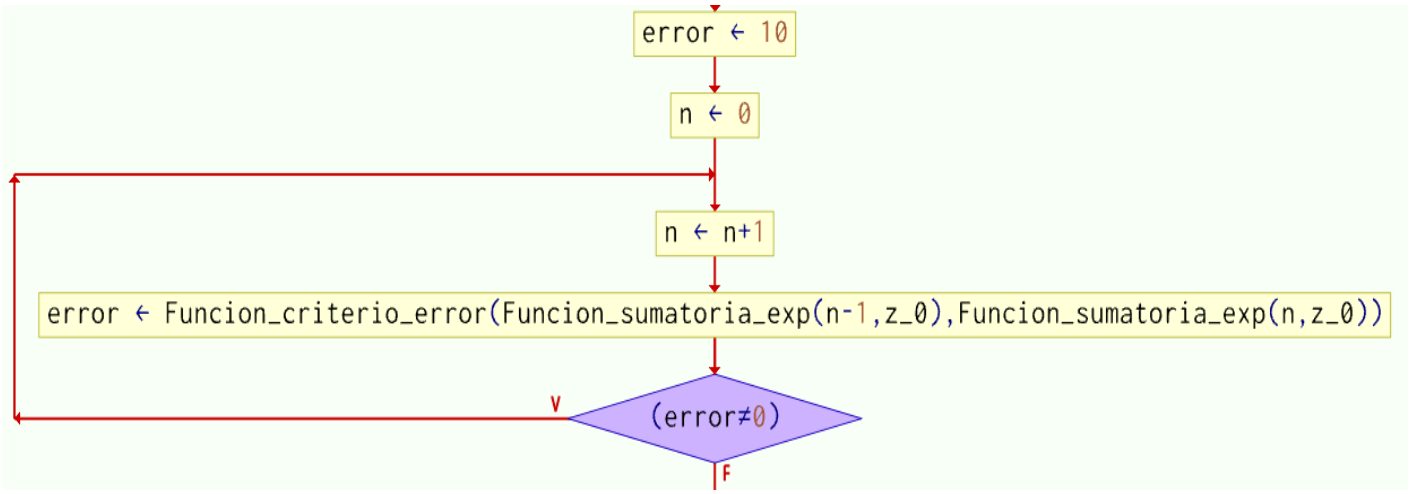
B. Subprograma Mostrar_Funcion_... (z_0, opción_m_3)

Para que sirve. –Este subprograma o Función, realiza las tareas necesarias para generar los n-términos de la serie, esto gracias a un par de funciones (quizá las más importantes para este algoritmo), cabe aclarar que para las demás series es similar la aplicación de Funciones (de tipo matemáticas).

Además de realizar dichos procesos, como su nombre lo dice, la función mostrara los resultados finales que el usuario desee ver (esto entre 3 opciones).

Inicia definiendo las variables que necesitara en el transcurso de las diferentes tareas, es importante notar que 2 variables tendrán valores, la variable error tiene el valor de 10 puesto que se crea un ciclo do{}while() que generara los n-termino y este acaba cuando el error esa 0, también le otorga el valor de 0 a la variable n, esta variable nos dirá cuántos términos fueron necesarios para generar el resultado final de la sumatoria

Zoom a la parte del inicio ciclo do{}while()



Cabe aclarar que dentro del ciclo yace la función matemática Funcion_criterio_error, la cual es de tipo flotante (retornara un valor decimal el cual será almacenado en la variable error), sin embargo, de mayor importancia resulta la dependencia de esta variable, puesto que depende de la Funcion_sumatoria_exp la cual a su vez depende de la cantidad de términos "n", y del punto de convergencia; Aunque no se vea por el momento, ambas funciones dependen de la función Funcion_factorial.

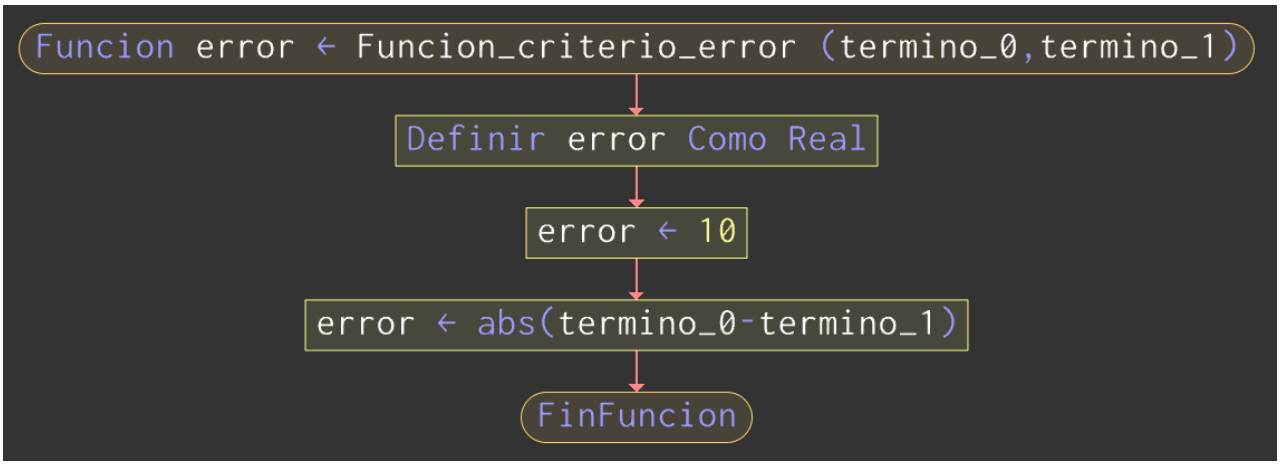
Antes de continuar será necesario ver a las funciones matemáticas dentro de este ciclo do{}while().

C. Funciones matemáticas.

Como ya se mencionó estas son las más importantes a la hora de entregar los valores finales de las series, estas son:

- **Funcion_criterio_error(,)**

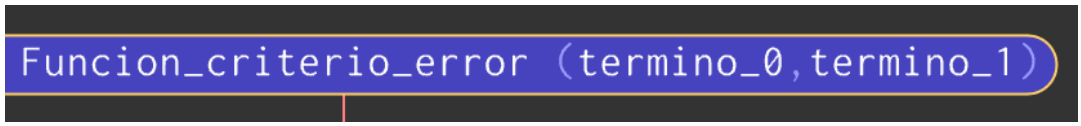
Esta evalúa de diferencia que hay entre la suma acumulada y la anterior, como también evalúa si el termino n-simo tiende a el valor nulo.



Para que esta función realice la tarea solicitada dentro de ciclo do{}while() (sin provocar algún error de programación), el contador "n" inicia antes que esta tarea de criterio de efectué para que al momento de realizar la diferencia "ya exista un termino_0".

Para este punto es necesario aclarar la dependencia de las Funciones de ciertas variables o´ para este caso funciones que entregaran valores; Para esto tomaremos prestado a la función desde su llamado dentro del algoritmo principal, y se la comparara con la forma en la que esta se creó:

a) Forma en la que está creada la función



Básicamente la función F_criterio_error fue creada para que dependa de 2 valores numéricos de tipo flotante, estos son termino_0 y termino_1.

b) Llamando a la función dentro del algoritmo principal

```
Funcion_criterio_error(Funcion_sumatoria_exp(n-1,z_0),Funcion_sumatoria_exp(n,z_0))
```

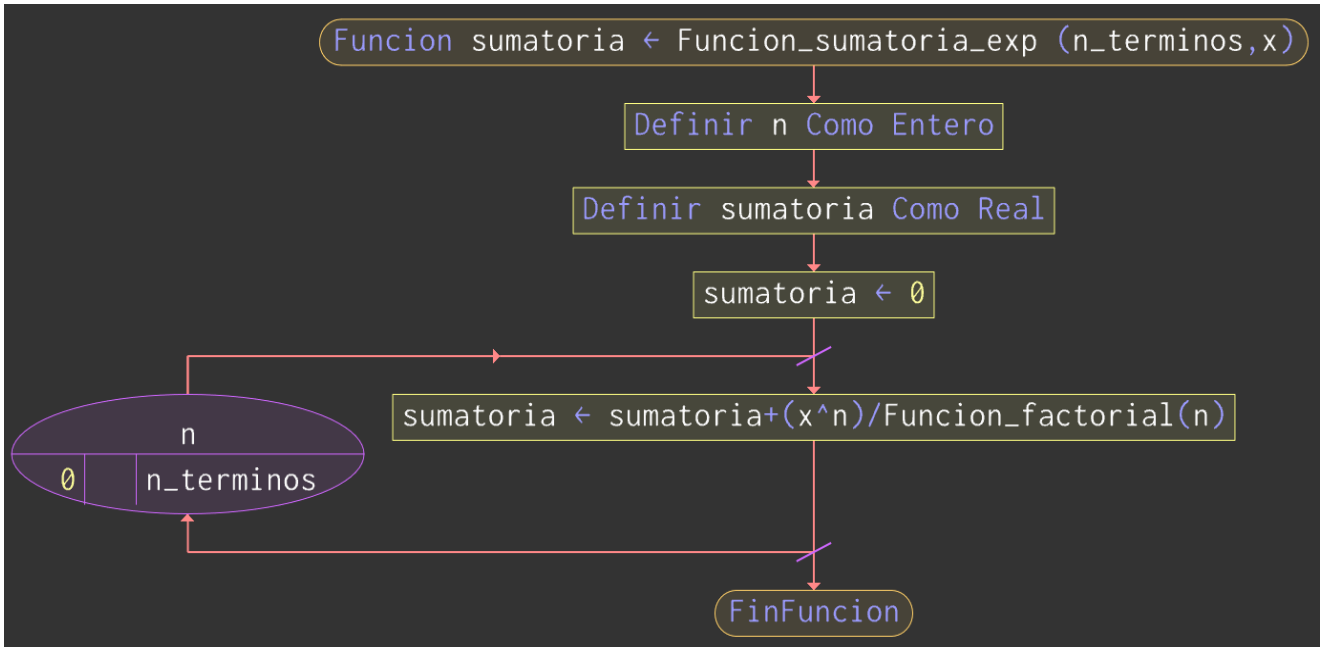
Al momento de ser convocada en el algoritmo principal, esta dependencia será de los valores que la Funcion_sum_exp() le entregue; Notando claramente en la imagen que a su vez esta función depende de las variables "n" y z_0.

- **Funcion_sumatoria_... (,)**

- Esta función acumula el valor de la fórmula de la serie que se desee realizar.
- Depende básicamente de la cantidad de termino y del punto de convergencia.
- Dentro de la función se definieron a n como entero ya que esta variable será utilizada en la fórmula de la serie, también se definió a una variable llamada "sumatoria" además que se le otorgo un valor de 0 para que al momento de ir acumulando el valor final de la sumatoria final este variable no tenga valores parásitos.
- Mediante un ciclo for(){}, iniciando desde 0 hasta el valor de n-términos, se irán generando los valores de los n-términos
- Dentro del ciclo for(){} ya mencionado se encuentra la fórmula matemática de la serie; Notando que en esta fórmula se encuentra la Funcion_factorial(n).

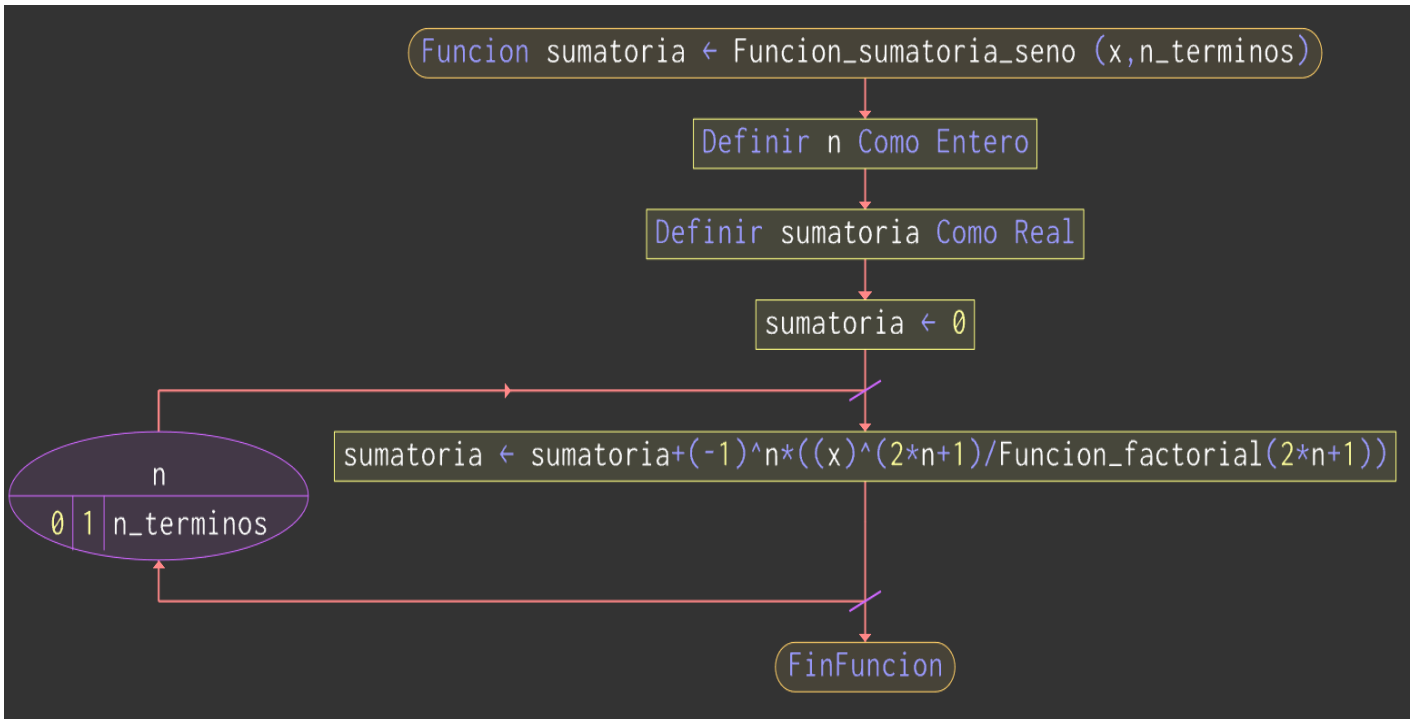
Continuación se verán los diagramas de flujos de las diferentes series.

1. Función serie exponencial

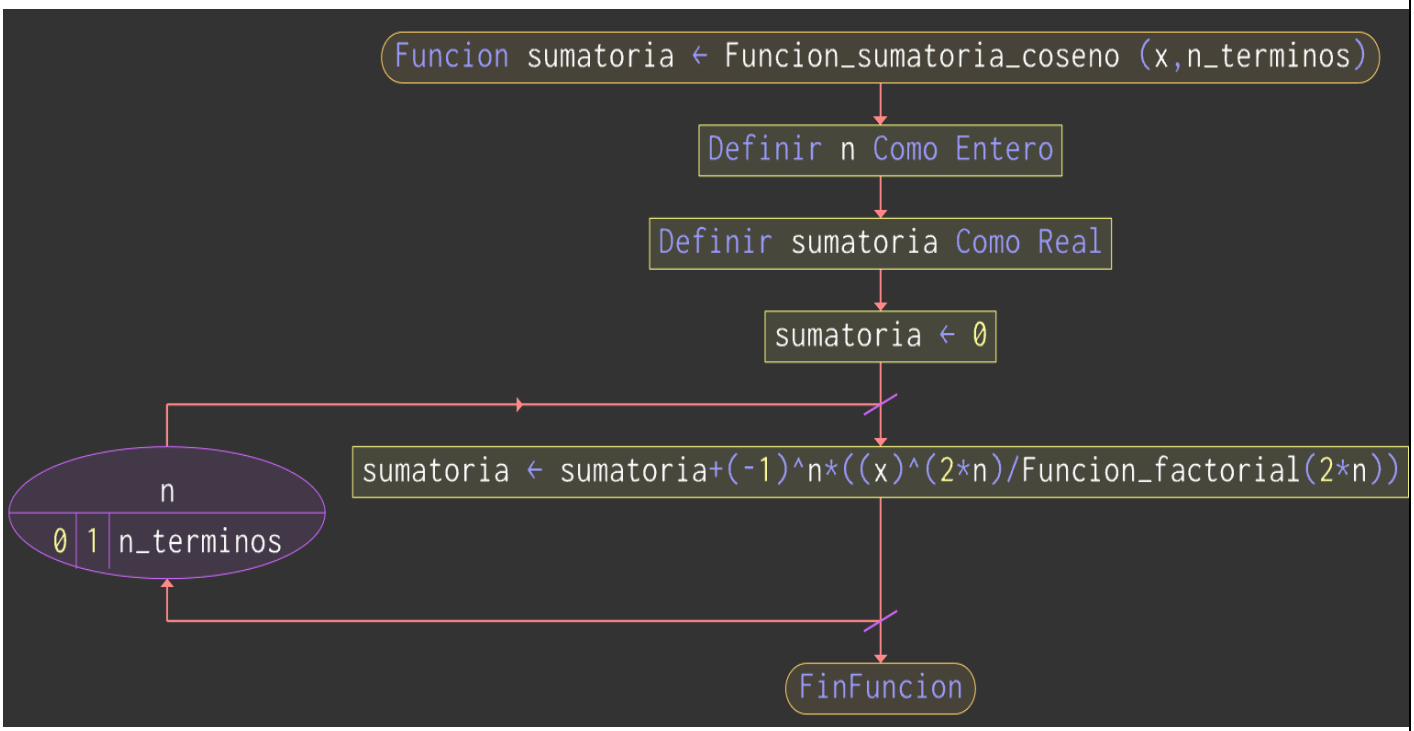


2. Funciones series trigonométricas

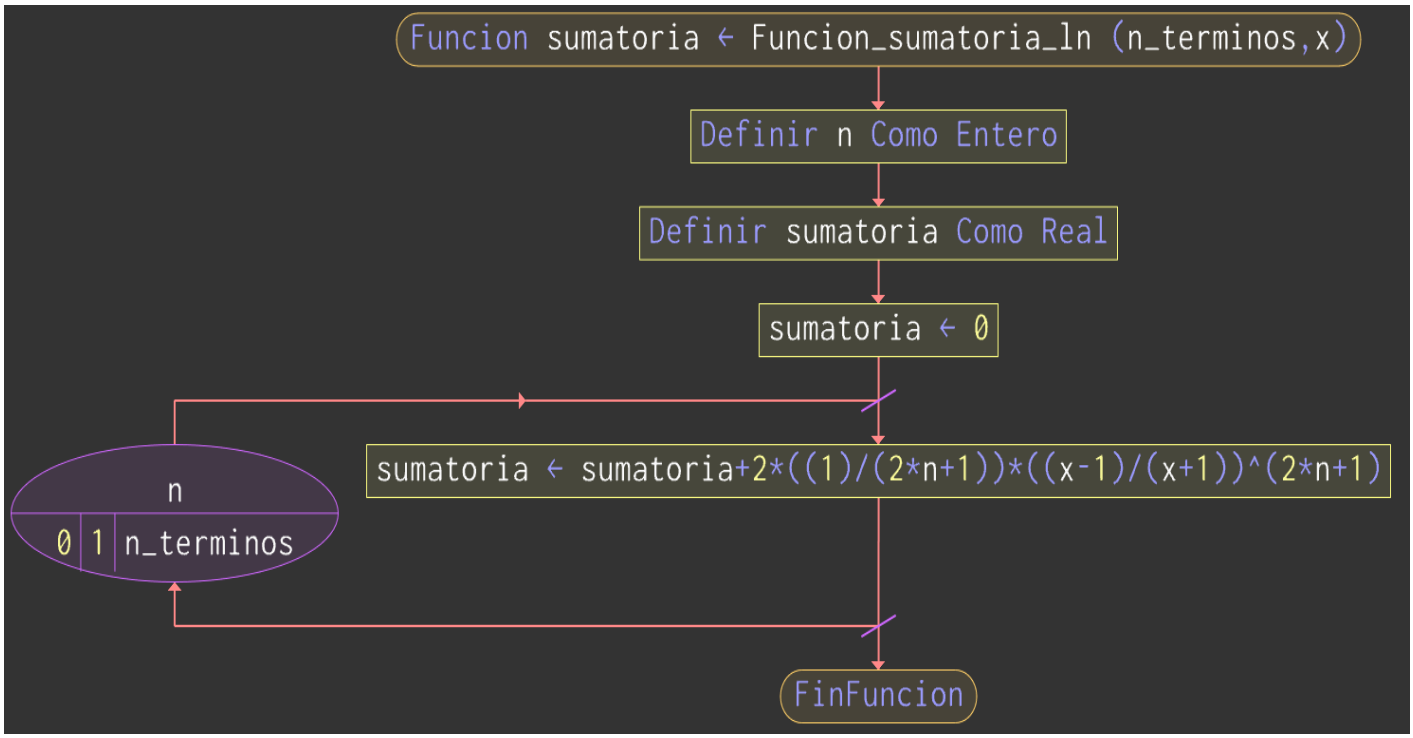
- **Función seno**



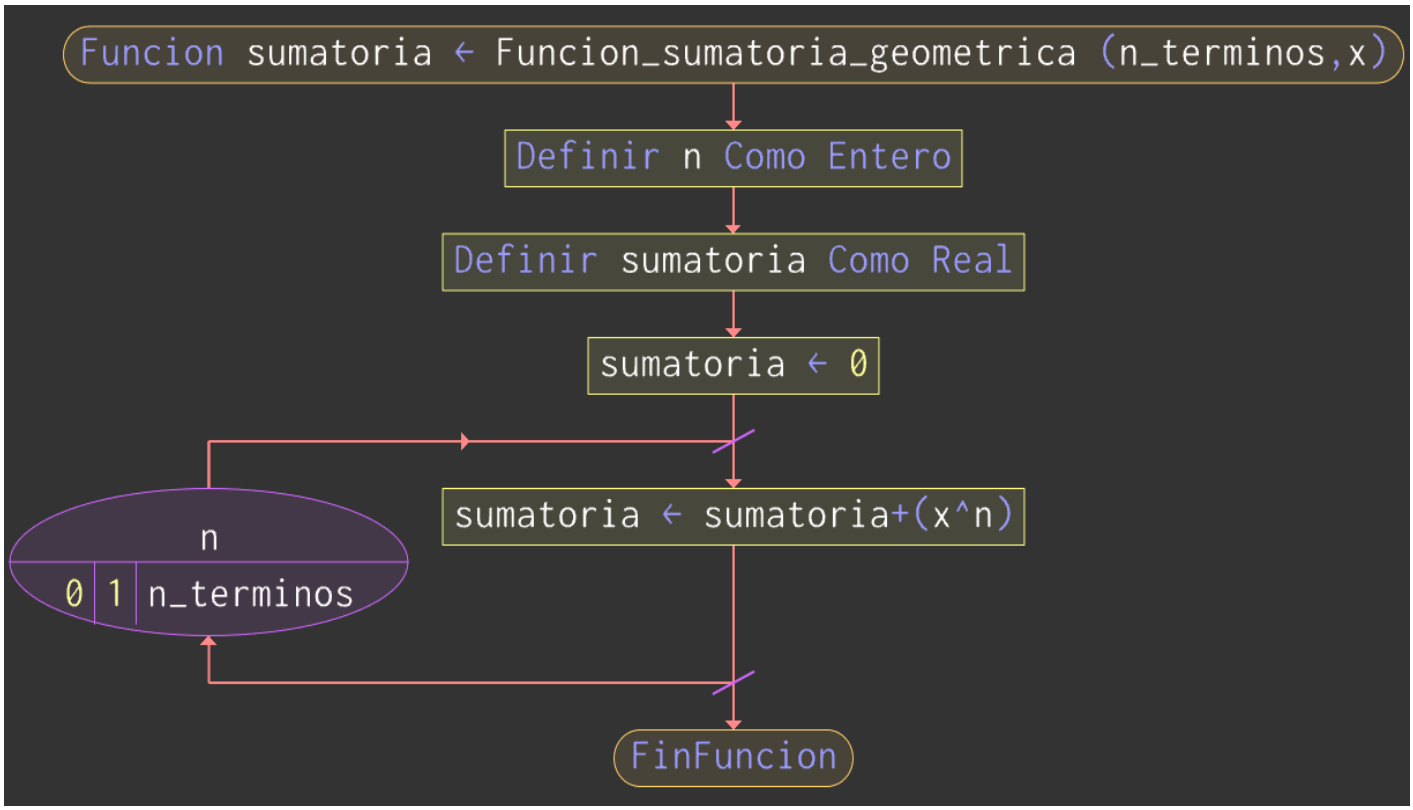
- **Función coseno**



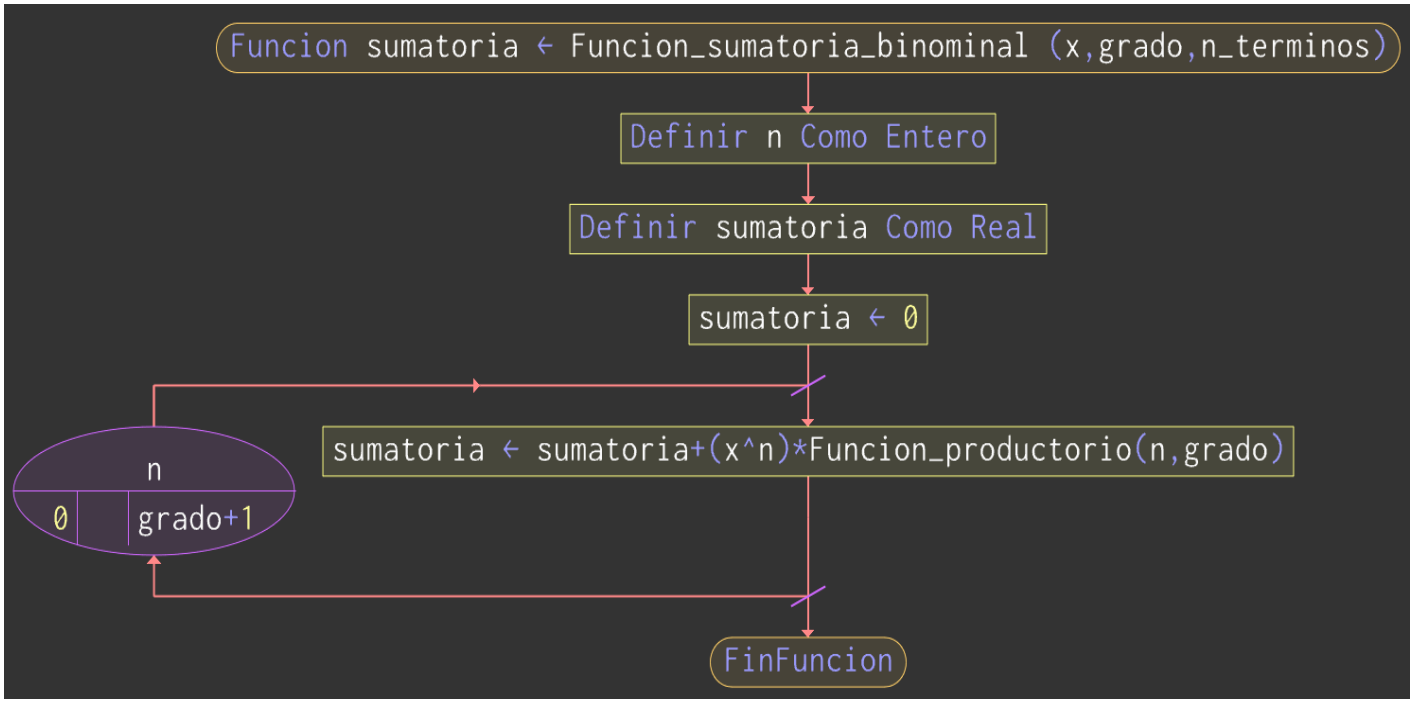
3. Función serie logaritmo natural



4. Función serie geométrica



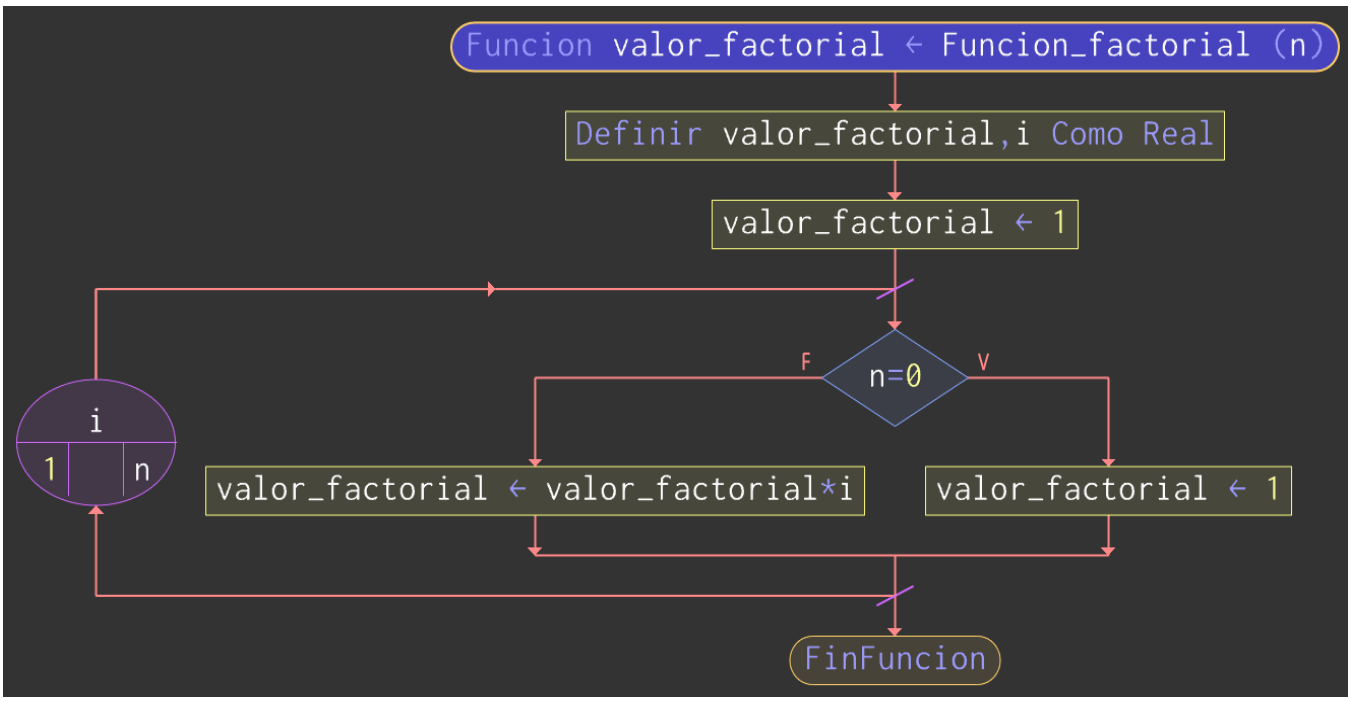
5. Función serie Binominal



Es necesario mencionar que esta última Funcion_sumatoria_binominal tiene dentro de su ciclo for(){} a la Funcion_productorio(n,grado), esto debido a que la fórmula de la serie binominal requiere el uso de dicha función.

- **Funcion_factorial (,)**

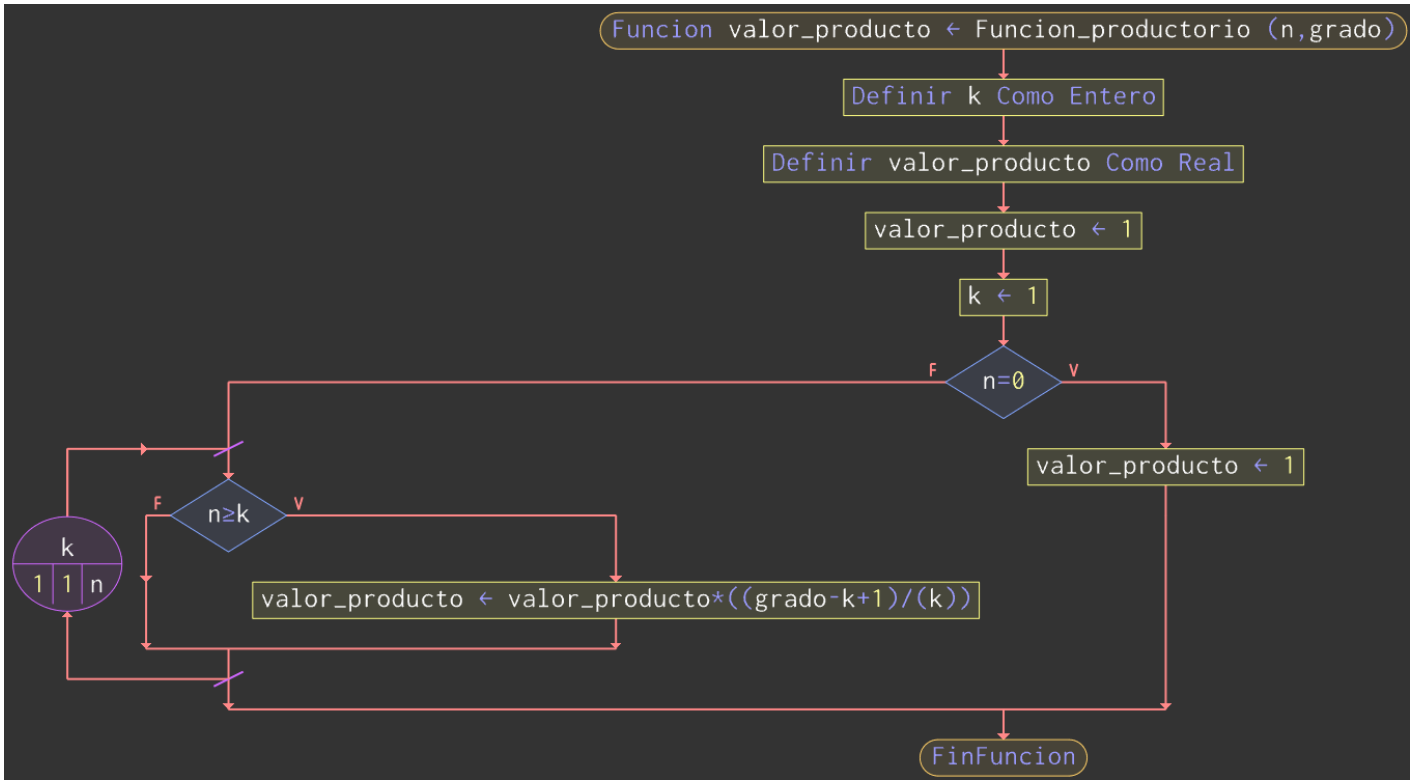
Talvez esta sea la función más recurrente al momento de ejecutar el programa final.



Depende de una variable “n” además dentro esta función se definen las variables de valor_factorial e i ,la primera guarda el valor acumulado del factorial que además será retornada por la función y la variable i cumple como contador dentro del ciclo for.
Además dentro de dicho ciclo es encuentra una condicional if() la cual cumple con una de las propiedades del factorial, es decir 0! = 1

- **Funcion_productorio (,)**

Esta función es usada para el momento en el que se desea obtener las operaciones de la serie binominal.



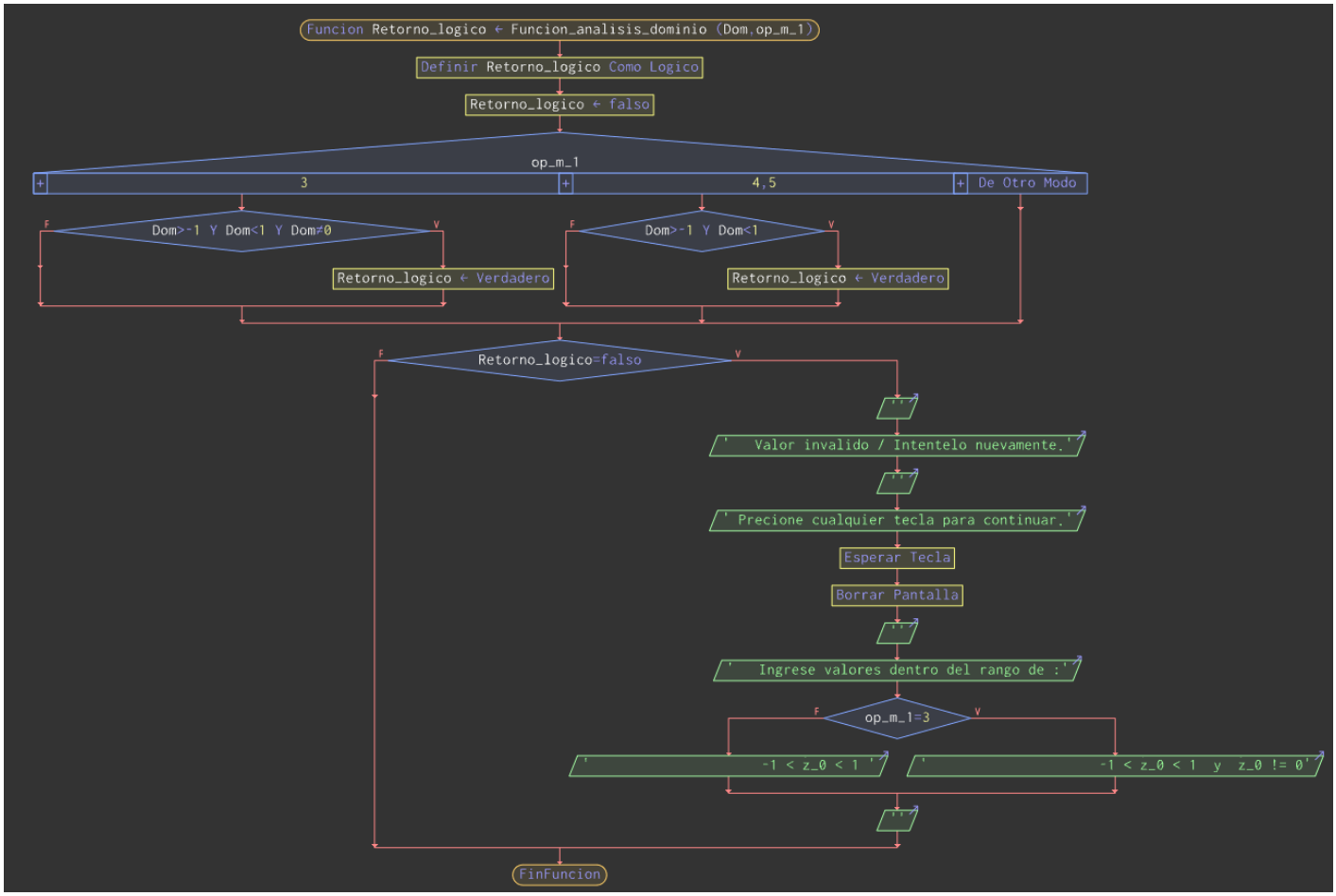
Cumple con la tarea matemática del productorio que la serie binominal necesita.

D. Funciones lógicas.

Para que sirve. –La única función lógica que se programó, fue la Funcion_analisis_dominio, esta función se activa al momento en el que el usuario necesite realizar las series de tipo logarítmica, serie geométrica y la serie binomial; Puesto que estas 3 tienen un radio en el disco de convergencia, el cual se vio en el marco teórico.

Básicamente esta función no dejara que el usuario ingrese un valor del punto de convergencia fuera de su respectivo radio de convergencia.

Dicha función depende de un valor de tipo flotante, el valor del punto ingresado, y de un valor entero el cual es el valor numérico del menu de opciones principal estos para este caso son los valores de { 3 (Serie logarítmica) , 4 (Serie geometrica) , 5 (Serie binomial) } es importante notar que la serie geométrica y la binomial comparten el mismo domino del punto de convergencia.

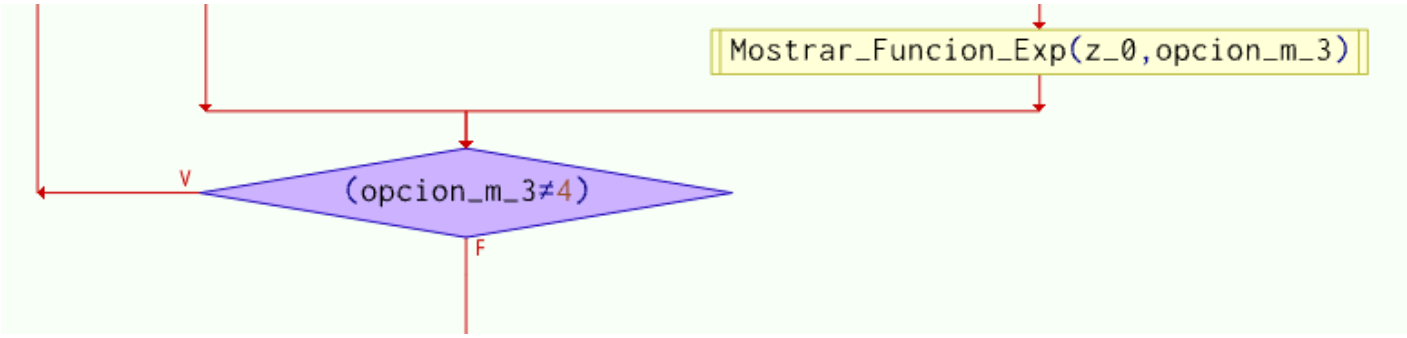


Para que el punto de convergencia ingresado por el usuario sea válido, este valor ingresa a una condicional switch() que depende de la opción seleccionada en el menú principal (Menú para elegir qué serie desea realizar) como esta función solo se activa para las series logarítmica, geométrica y binomial estas serán los posibles casos de la condicional switch; Sin embargo la serie geométrica y la serie binomial comparten el mismo radio de convergencia, por lo que para este caso se los agrupo en el mismo bloque de tareas, si el valor ingresado no cumple con la condicional if() encontrada dentro los posibles casos de switch() entonces la variable logia

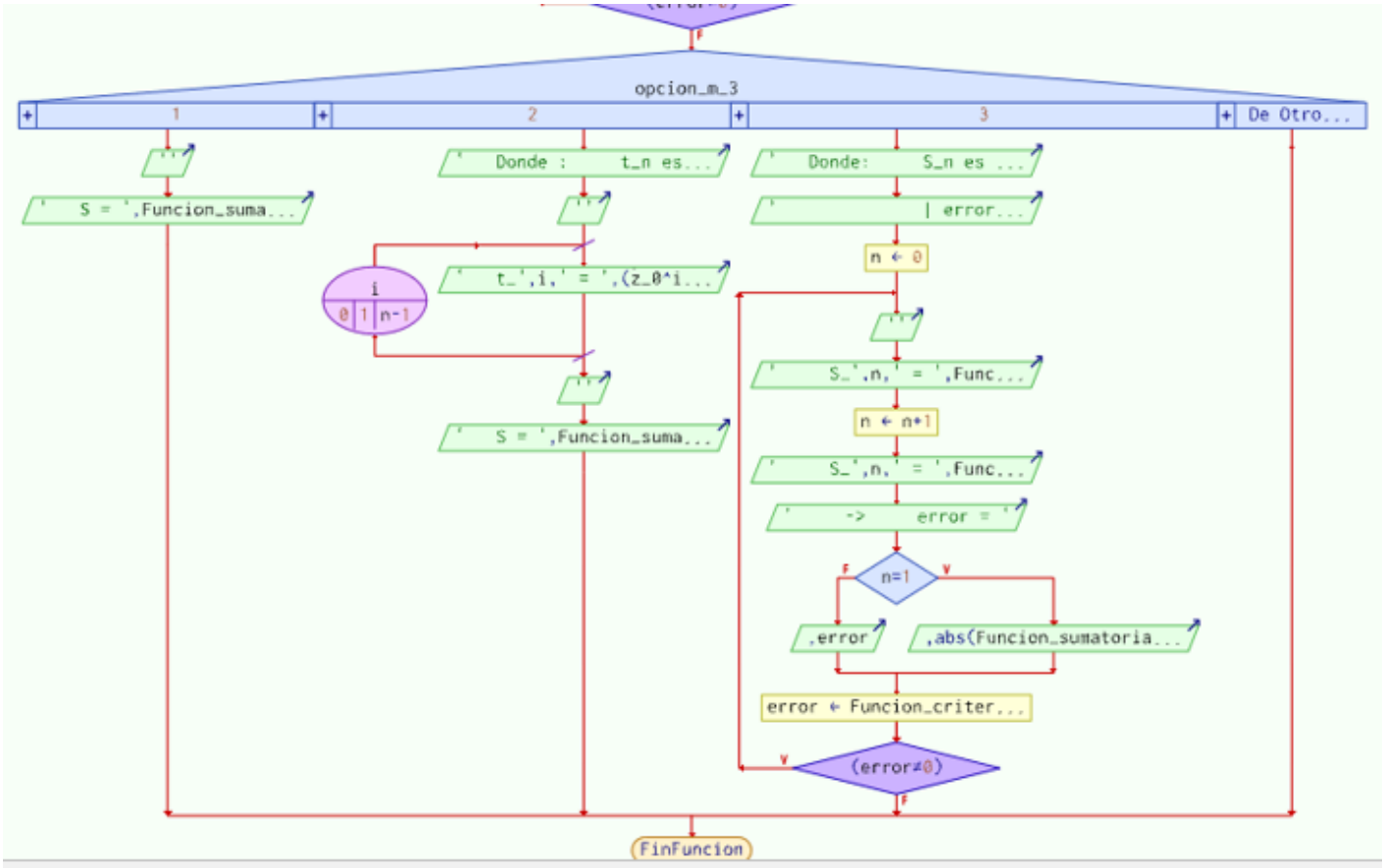
"Retorno_logico" no cambia conserva el valor lógico de falso; Sin embargo si el valor ingresado cumple con dicha condicional, el retorno lógico cambiara a verdadero.

Para la segunda parte de esta función_analisis_dominio, se encuentra una condicional if() que evalúa si el "Retorno_logico" cambio o no. Si fuera el caso de que "Retrono loico = True", termina la función; Caso contrario se mostrara en la consola que dicho valor no se encuentra dentro del radio de convergencia y deberá ingresar nuevamente un valor, esta vez también le mostrara el radio de convergencia aceptado y luego de esto la función finaliza.

Y estos son las funciones que llegarían a activarse para las distintas series que tiene el algoritmo, y se procederá a explicar desde el punto en el cual el subprograma Mostar sumatoria … sale de un ciclo do{}while (El cual evaluaba el error)

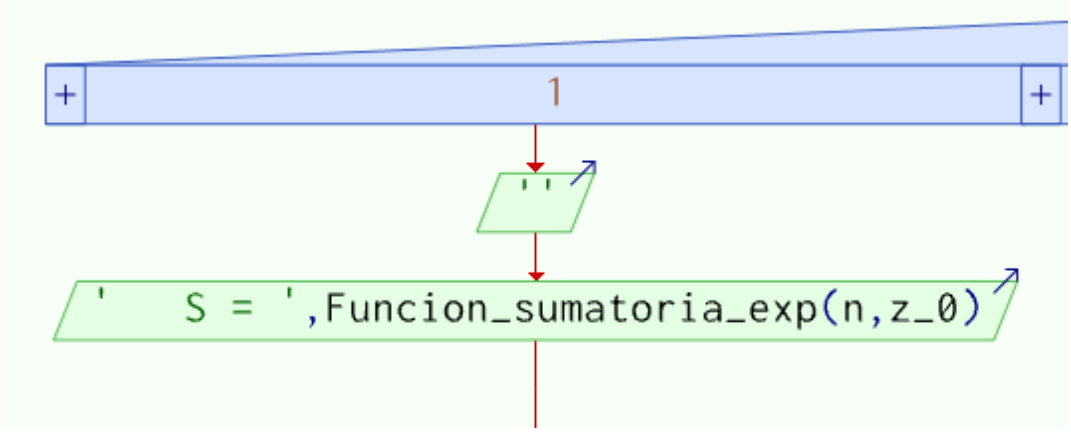


Para ingresar en un switch() como se ve a continuación:

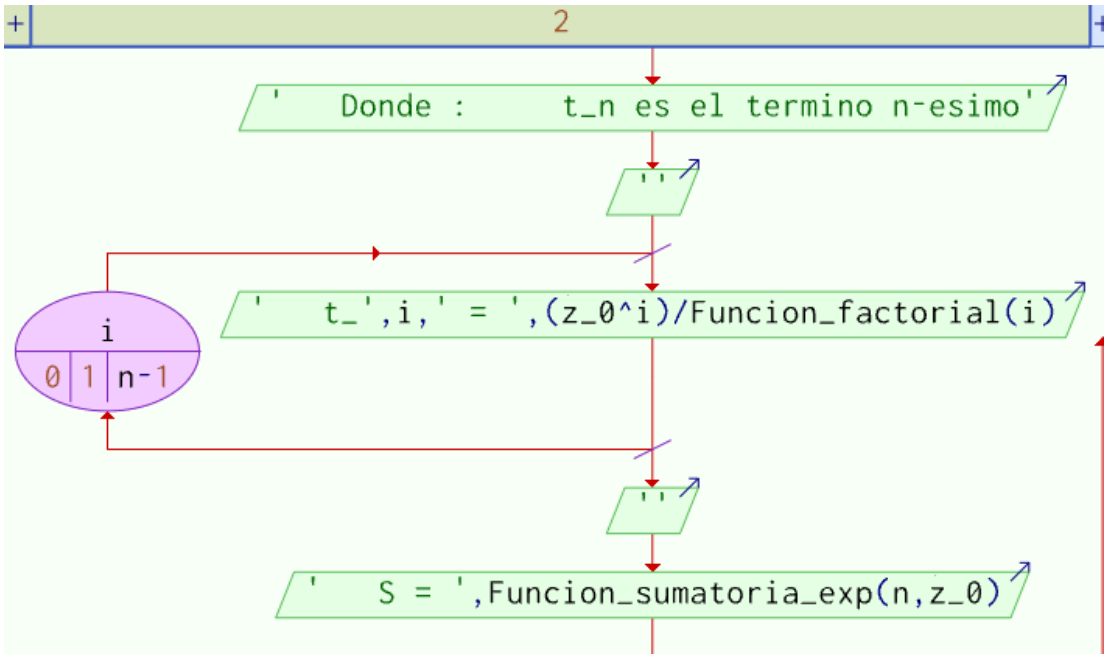


Ya generado la cantidad de términos que la serie contiene esto gracias al ciclo do{}while() ya cerrado finalmente el algoritmo ingresa a la condicional switch () la cual evalúa el valor de opción_m_3 la cual era el resultado del menú que pedía al usuario si quería ver el valor de la sumatoria final, los n-términos o el criterio que se utilizó para generar dicha sumatoria.

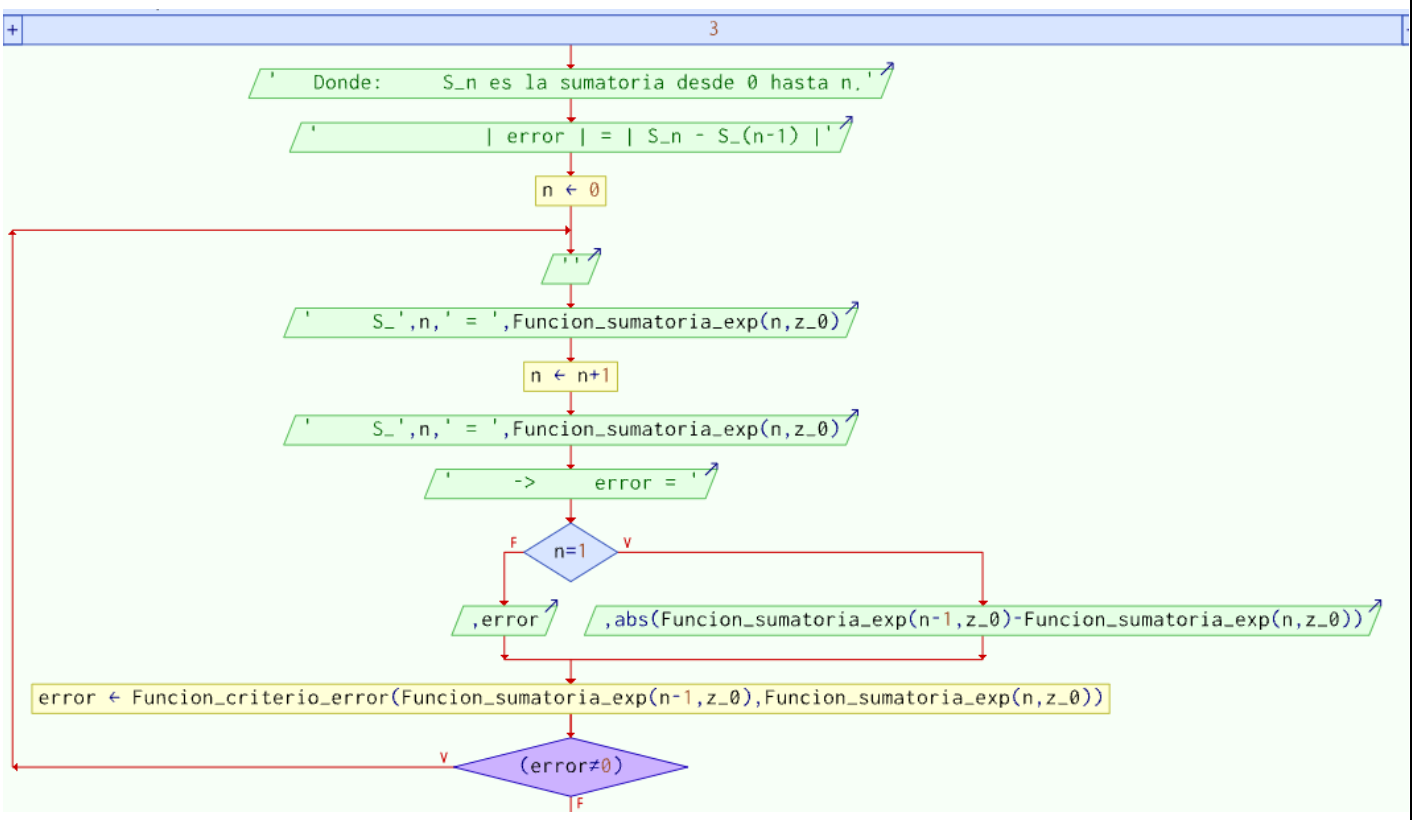
Para el 1er bloque, simplemente se llama a la Funcion_sumatoria_..., la cual para este punto tiene el valor final de dicha sumatoria



Para el 2do bloque, es utilizado un ciclo for(){} el cual inicia desde 0 hasta el valor que n tiene guardado, dentro del ciclo se realiza la tarea de generar y mostrar en la consola cada uno de los n termino esto con la fórmula de la serie y llamando a la Funcion_factorial.
Saliendo del ciclo el bloque se imprime el valor de la sumatoria.



Para el 3er bloque, nuevamente se le da el valor de 0 a la variable “n”, esto con el fin de generar las n-sumatorias y poder aplicar el criterio mediante un ciclo do{}while() que finalizara si el criterio tiende a cero.
Además, se mostrará en la consola cada paso realizado.



Finalmente el subprograma Mostrar_funcion_...() termina pero este se repetirá hasta que el usuario digite la opción de salir y retornamos al menú principal.

Para terminar los procesos del programa finalmente el usuario deberá elegir la opción de salir del menú principal.

Listado del código en el programa C++

A. Librerías

- **<iostream>**, para el ingreso y salida de datos.
- **<cmath>**, para realizar operaciones matemáticas de potencias o raíces, trigonométricas (seno, coseno, arco senos, etc.), etc.

- **<conio.h>**, para llamar a la función “getch(); “ que congela la consola hasta que el usuario digite una tecla.

B. Funciones

Nota. - Para que las capturas de pantalla lograran entrar en las páginas, en algunos casos, la 2da condicional switch, la cual imprimia en pantalla la elección de usuario (teoría o la resolución de las series).

- **int analiza_menu_1()**
  Evita que el usuario ingrese valores erróneos para el momento en el que está escogiendo entre las opciones del menú principal.

```
286 int analiza_menu_1() {
287     int opcion_1_n = 10;
288     char op_1_char[10];
289     cin >> op_1_char[0];
290     switch (op_1_char[0]) {
291     case '1':   opcion_1_n = 1;
292         break;
293     case '2':   opcion_1_n = 2;
294         break;
295     case '3':   opcion_1_n = 3;
296         break;
297     case '4':   opcion_1_n = 4;
298         break;
299     case '5':   opcion_1_n = 5;
300         break;
301     case '6':   opcion_1_n = 6;
302         cout << "\n    * Saliendo del programa *" << endl;
303         break;
304     default:    opcion_1_n = 0;
305         cout << "   Opcion invalida / Intentelo nuevamente." << endl;
306     }
307     cout << "\n Presione cualquier tecla para continuar." << endl;
308     getch();
309     system("cls");
310     if (opcion_1_n!=0) {
311         cout << endl;
312         switch (opcion_1_n) {
359     }
360     return opcion_1_n;
361 }
```

- **int analiza_2_opciones()**

  Para los menús de 2 posibles opciones.

```
363  // para menus de 2 opciones
364  int analiza_2_opciones() {
365      int opcion_2_n = 0;
366      char op_2_char[10];
367      cin >> op_2_char[0];
368      switch (op_2_char[0]) {
369      case '1':   opcion_2_n = 1;
370          break;
371      case '2':   opcion_2_n = 2;
372          break;
373      default:    opcion_2_n = 0;
374          cout << "   Opcion invalida / Intentelo nuevamente." << endl;
375      }
376      cout << "\n Presione cualquier tecla para continuar." << endl;
377      getch();
378      system("cls");
379      return opcion_2_n;
380  }
```

- **int analiza_4_opciones();**

  Para los menús de 4 posibles opciones.

```
381  // Analiza entres 4 opciones
382  int analiza_4_opciones() {
383      int opcion_4_n = 0;
384      char op_4_char[10];
385      cin >> op_4_char[0];
386      switch (op_4_char[0]) {
387      case '1':   opcion_4_n = 1;
388          break;
389      case '2':   opcion_4_n = 2;
390          break;
391      case '3':   opcion_4_n = 3;
392          break;
393      case '4':   opcion_4_n = 4;
394          break;
395      default:    opcion_4_n = 0;
396          cout << "   Opcion invalida / Intentelo nuevamente." << endl;
397      }
398      return opcion_4_n;
399  }
```

- **void menu_ver_resultados();**

  Mostrará en la pantalla el meno que se ve a continuación.

```
400  // menu ver resultados
401  void menu_ver_resultados() {
402      cout << "\n Presione cualquier tecla para continuar." << endl;
403      getch();
404      system("cls");
405      cout << "\n          * Menu *" << endl;
406      cout << "   Desea ver :" << endl;
407      cout << "   1. El valor de la sumatoria." << endl;
408      cout << "   2. Los n-terminos." << endl;
409      cout << "   3. Procedimiento." << endl;
410      cout << "   4. Salir" << endl;
411      cout << "   Digite su opcion. -> ";
412  }
```

Funciones tipo (mostrar_funcion_...). –

En esta sección las funciones llamaran a los subprogramas que realizaran distinto tareas matemáticas que generaran el valor final de la sumatoria, el valor de los n-términos, y más; También mostraran esto en la consola dependiendo lo que el usuario desee ver.

- **void mostrar_funcion_exp(float z_0, int opcion_m_3);**

```
417  void mostrar_funcion_exp(float z_0, int opcion_m_3) {
418      float error = 10,n = 0;// n+1 es la catidad de terminos
419      int i;
420      // corremos las funciones de sum exp que a su timpo depende de fact
421      // y consegimos el valor de n-terminos
422      do {
423          n = n+1;
424          error = funcion_criterio_error(funcion_sumatoria_exp(n-1,z_0),funcion_sumatoria_exp(n,z_0));
425      } while (error!=0);
426      switch (opcion_m_3) {
456  }
```

- **void mostrar_funcion_trigo(float z_0, int opcion_m_3, float opcion_m_4);**

```
458  // TRIGONOMETRICA
459  void mostrar_funcion_trigo(float z_0, int opcion_m_3, float opcion_m_4) {
460      float error = 10,n = 0;
461      int i;
462      if (opcion_m_4==1) {
463          // SENO (-1)^n*( (x)^(2*n+1)/Funcion_factorial(2*n+1)
464          do {
465              n = n+1;
466              error = pow((-1),n)*(pow((z_0),(2*n+1))/funcion_factorial(2*n+1));
467          } while (error!=0);
468          switch (opcion_m_3) {
491      } else {
492          // COSENO   (-1)^n*( (z_0)^(2*n)/Funcion_factorial(2*n) );
493          do {
494              n = n+1;
495              error = pow((-1),n)*(pow((z_0),(2*n))/funcion_factorial(2*n));
496          } while (error!=0);
497          switch (opcion_m_3) {
520      }
521  }
```

- **void mostrar_funcion_ln(float z_0, int opcion_m_3);**

```
523  // LOGARITMICA
524  void mostrar_funcion_ln(float z_0, int opcion_m_3) {
525      float error = 10,n = 0;
526      int i;
527      // n+1 es la catidad de terminos
528      cout<<"z_0=",z_0;
529      do {
530          error = (2/(2*n+1))*pow(((z_0-1)/(z_0+1)),(2*n+1));
531          n = n+1;
532      } while (error!=0);// el ternimo n-simo tiende a cero por valores negativos
533      switch (opcion_m_3) {
559  }
```

- **void mostrar_funcion_geo(float z_0, int opcion_m_3);**

```
561  // GEOMETRICA
562  void mostrar_funcion_geo(float z_0, int opcion_m_3) {
563      float error = 10,n = 0;// n+1 es la catidad de terminos
564      int i;
565      // corremos las funciones de sum exp que a su timpo depende de fact
566      // y consegimos el valor de n-terminos
567      do {
568          error = (pow(z_0,n));
569          n = n+1;
570      } while (error!=0);
571      switch (opcion_m_3) {
594  }
```

- **void mostrar_funcion_binominal(float z_0, float grado, int opcion_m_3);**

```
596  // BINOMINAL
597  void mostrar_funcion_binominal(float z_0, float grado, int opcion_m_3) {
598      float error = 10,n = 0;// n+1 es la catidad de terminos
599      int i;
600      // corremos las funciones de sum exp que a su timpo depende de fact
601      // y consegimos el valor de n-terminos
602      do {
603          error = (pow(z_0,n));
604          n = n+1;
605      } while (error!=0);
606      switch (opcion_m_3) {
628  }
```

Funciones tipo (función_sumatoria_...). –

En esta sección las funciones llamaran a los subprogramas función_factorial y función_productorio, para generar el valor de la sumatoria.

- **float funcion_sumatoria_exp(float n_terminos, float x);**

```
636 float funcion_sumatoria_exp(float n_terminos, float x) {
637     int n;
638     float sumatoria = 0;
639     for (n=0 ; n<=n_terminos ; n++) {
640         sumatoria = sumatoria+(pow(x,n))/funcion_factorial(n);
641     }
642     return sumatoria;
643 }
```

- **float funcion_sumatoria_seno(float x, float n_terminos);**

```
645 // * FUNCION SENO : sen x ; donde x pretencea los reales
646 // * sumatoria ( inicio n=0 , hasta n=n_terminos ) = (-1)^n*( (x)^(2*n+1)/(2*n+1)! )
647 float funcion_sumatoria_seno(float x, float n_terminos) {
648     int n;
649     float sumatoria = 0;
650     for (n=0 ; n<=n_terminos ; n++) {
651         sumatoria = sumatoria+pow((-1),n)*(pow((x),(2*n+1))/funcion_factorial(2*n+1));
652     }
653     return sumatoria;
654 }
```

- **float funcion_sumatoria_coseno(float x, float n_terminos);**

```
656 // * FUNCION COSENO : cos x ; donde x pretencea los reales
657 // * sumatoria ( inicio n=0 , hasta n=n_terminos ) = (-1)^n*( (x)^(2*n)/(2*n)! )
658 float funcion_sumatoria_coseno(float x, float n_terminos) {
659     int n;
660     float sumatoria = 0;
661     for (n=0 ; n<=n_terminos ; n++) {
662         sumatoria = sumatoria+pow((-1),n)*(pow((x),(2*n))/funcion_factorial(2*n));
663     }
664     return sumatoria;
665 }
```

- **float funcion_sumatoria_ln(float n_terminos, float x);**

```
667 // * * * LAS FUCIONES QUE SIGUEN TIENEN RESTRICCIONES PARA X * * *
668 // * FUNCION LOGARITMO NATURAL : ln(x)  ; -1<x<1 y x<>0
669 // * sumatoria ( inicio n=0 , hasta n=n_terminos ) = [ 1/(2*n+1) ]*[ (x-1)
670 float funcion_sumatoria_ln(float n_terminos, float x) {
671     int n;
672     float sumatoria = 0;
673     for (n=0 ; n<=n_terminos ; n++) {
674         sumatoria = sumatoria+2*((1)/(2*n+1))*pow(((x-1)/(x+1)),(2*n+1));
675     }
676     return sumatoria;
677 }
```

- **float funcion_sumatoria_geometrica(float n_terminos, float x);**

```
678
679  // * FUNCION SERIE GEOMETRICA : (1-x)^-1  ;  -1<x<1
680  // NOTA X puede ser igual a la unidad
681  // * sumatoria ( inicio 0 , hasta n=n_terminos )= x^n
682  float funcion_sumatoria_geometrica(float n_terminos, float x) {
683      int n;
684      float sumatoria =0;
685      for (n=0 ; n<=n_terminos ; n++) {
686          sumatoria = sumatoria+(pow(x,n));
687      }
688      return sumatoria;
689  }
```

- **float funcion_sumatoria_binominal(float x, float grado, int n_terminos);**

```
691  // * FUNCION SERIE BINOMIAL : (1-x)^-1  ;  -1<x<1
692  // NOTA X puede ser igual a la unidad
693  // * sumatoria ( inicio 0 , hasta n=n_terminos )= x^n
694  float funcion_sumatoria_binominal(float x, float grado, int n_terminos) {
695      int n;
696      float sumatoria =0;
697      for (n=0 ; n<=grado+1 ; n++) {
698          sumatoria = sumatoria+(pow(x,n))*funcion_productorio(n,grado);
699      }
700      return sumatoria;
701  }
```

- **float funcion_criterio_error(float termino_0, float termino_1);**

```
703  float funcion_criterio_error(float termino_0, float termino_1) {
704      float error = 10;
705      error = abs(termino_0-termino_1);
706      return error;
707  }
```

- **float funcion_factorial(float n);**

```
712 float funcion_factorial(float n) {
713     float i;
714     float valor_factorial =1;
715     for (i=1 ; i<=n ; i++) {
716         if (n==0) {
717             valor_factorial = 1;
718         } else {
719             valor_factorial = valor_factorial*i;
720         }
721     }
722     return valor_factorial;
723 }
```

- **float funcion_productorio(float n, float grado);**

```
726 float funcion_productorio(float n, float grado) {
727     int k = 1;
728     float valor_producto =1;
729     if (n==0) {
730         valor_producto = 1;
731     } else {
732         for (k=1;k<=n;k++) {
733             if (n>=k) {
734                 valor_producto = valor_producto*((grado-k+1)/(k));
735             }
736         }
737     }
738     return valor_producto;
739 }
```

- **bool funcion_analisis_dominio(float dom, int op_m_1);**

```
744 bool funcion_analisis_dominio(float dom, int op_m_1) {
745     bool retorno_logico = false;
746     switch (op_m_1) {
747     case 3:
748         if (dom>-1 && dom<1 && dom!=0) {
749             retorno_logico = true;
750         }
751         break;
752     case 4:
753         if (dom>-1 && dom<1) {
754             retorno_logico = true;
755         }
756         break;
757     case 5:
758         if (dom>-1 && dom<1) {
759             retorno_logico = true;
760         }
761         break;
762     }
763     if (retorno_logico==false) {
764         cout << "\n   Valor invalido / Intentelo nuevamente." << endl;
765         cout << "\n Presione cualquier tecla para continuar." << endl;
766         getch();
767         system("cls");
768         cout << "   Ingrese valores dentro del rango de :" << endl;
769         if (op_m_1==3) {
770             cout << "                    -1 < z_0 < 1  y  z_0 != 0" << endl;
771         } else {
772             cout << "                    -1 < z_0 < 1 " << endl;
773         }
774         cout << endl;
775     }
776     return retorno_logico;
777 }
```

Capturas de pantalla de la ejecución del programa. –

- Probando el menú inicial:

  Ingresando valores erróneos al menú:

L1_RMM_E1_v4.cpp

```
1  #inclu
2  #inclu
3  #inclu
4  using
5  // Dec
6  int an
7  int an
8  int an
9  void m
10 void m
11 void m
12 void m
13 void m
14 void m
15 float
16 float
17 float
18 float funcion_sumatoria_ln(float n_terminos, float x);
19 float funcion_sumatoria_geometrica(float n_terminos, float x);
20 float funcion_sumatoria_binominal(float x, float grado, int n_terminos);
21 float funcion_criterio_error(float termino_0, float termino_1);
22 float funcion_factorial(float n);
23 float funcion_productorio(float n, float grado);
24 bool funcion_analisis_dominio(float dom, int op_m_1);
```

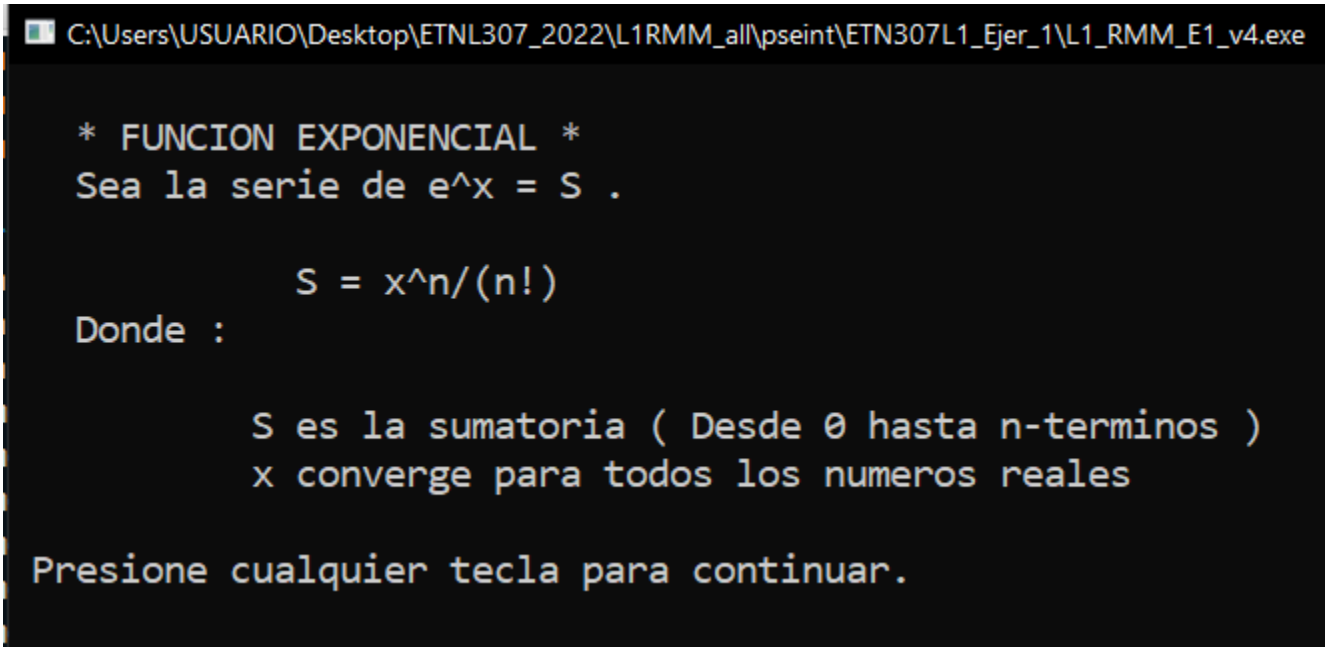
```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_1\L1_RMM_E1_v4.exe

  * Series de Taylor y Maclaurin *
          Menu
  1. Funcion exponencial.
  2. Funciones trigonometricas.
  3. Funcione logaritmica.
  4. Funcione geometrica.
  5. Funcione binomica.
  6. Salir.

  Digite su opcion. -> a
  Opcion invalida / Intentelo nuevamente.

Presione cualquier tecla para continuar.
```

2da opción errónea.

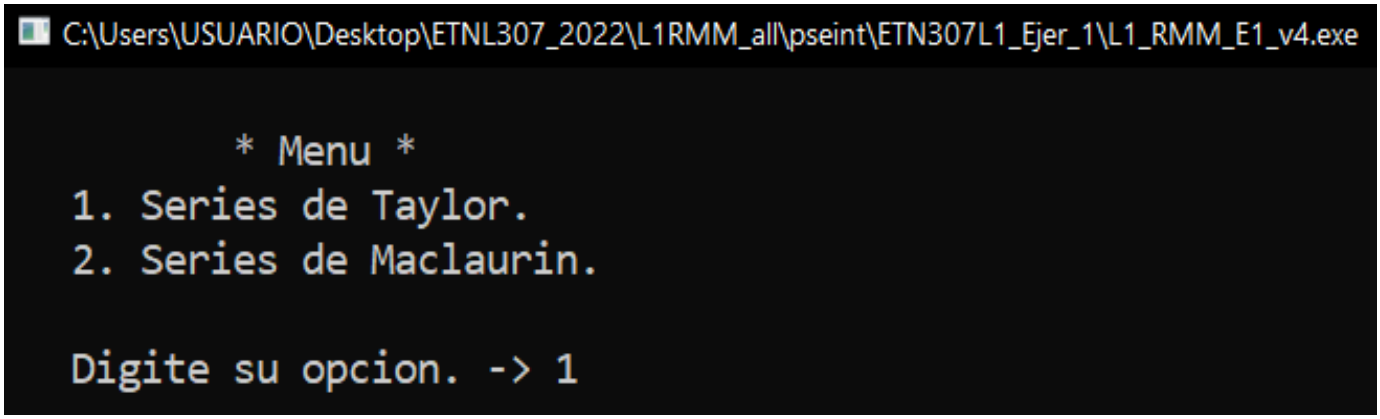
```
  Digite su opcion. -> 0
  Opcion invalida / Intentelo nuevamente.

Presione cualquier tecla para continuar.
```

Todo esto gracias a un ciclo do{}while.

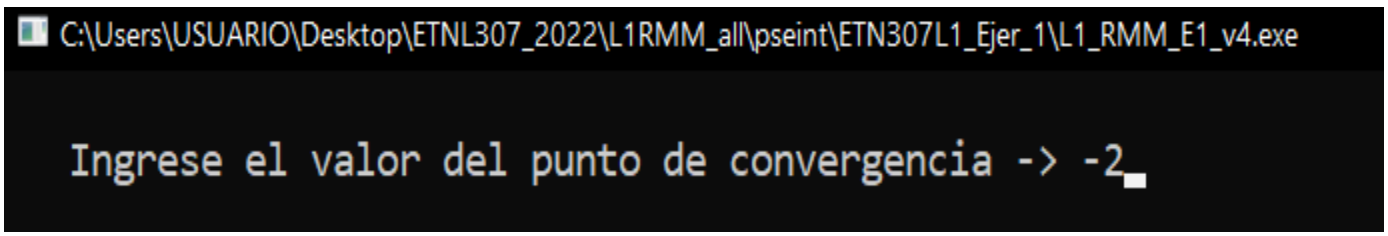
- Ingresando a la Serie exponencial.



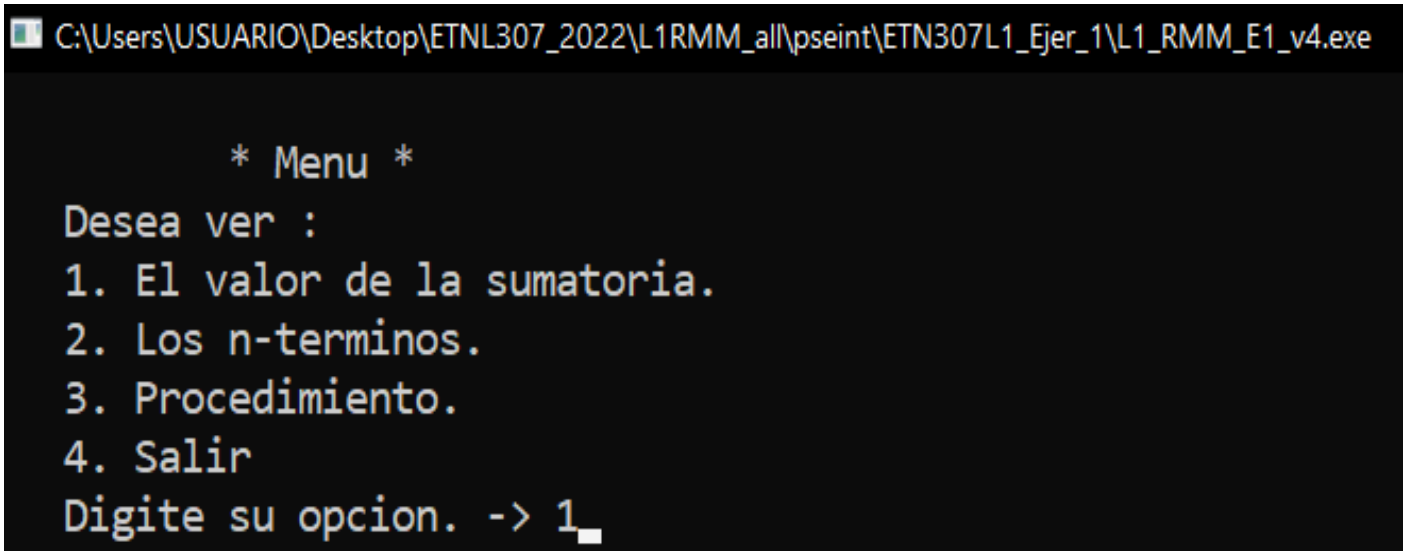
Presionando cualquier tecla:



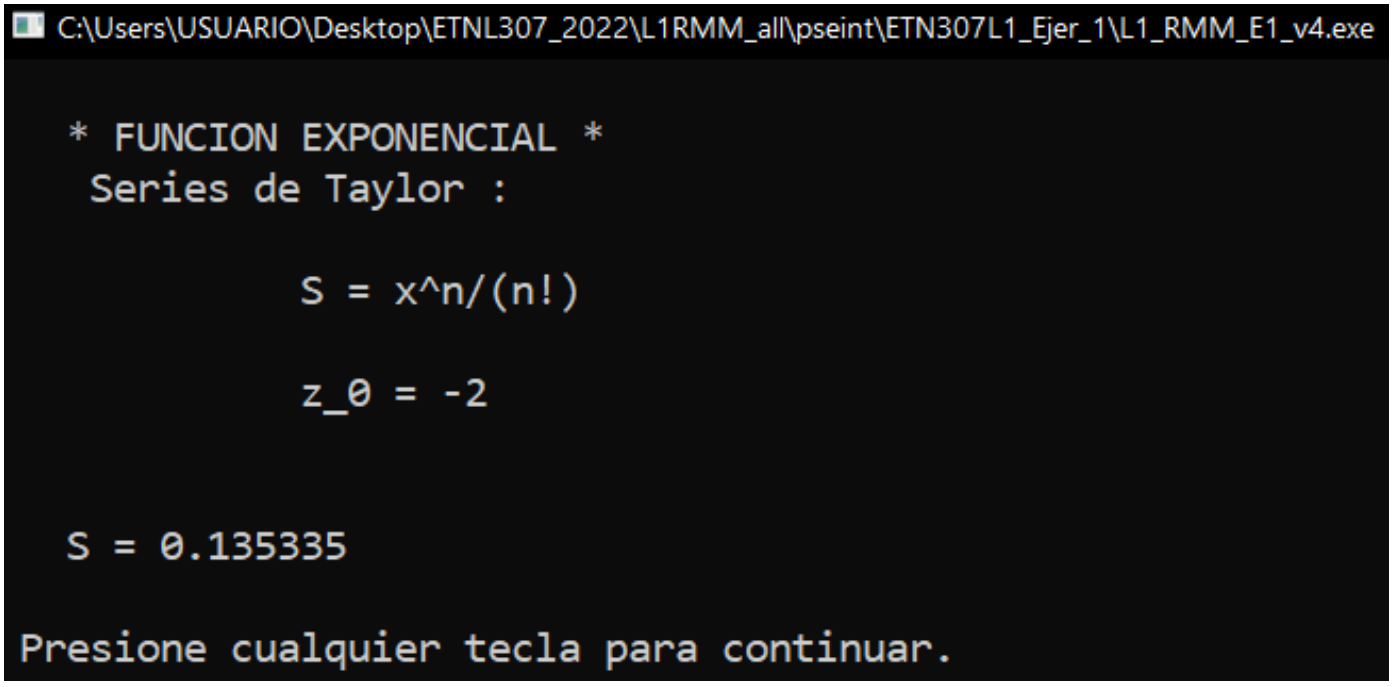
Luego de elegir la 1ra opción:



Ingresando el valor de convergencia = -2, y luego presionando enter.



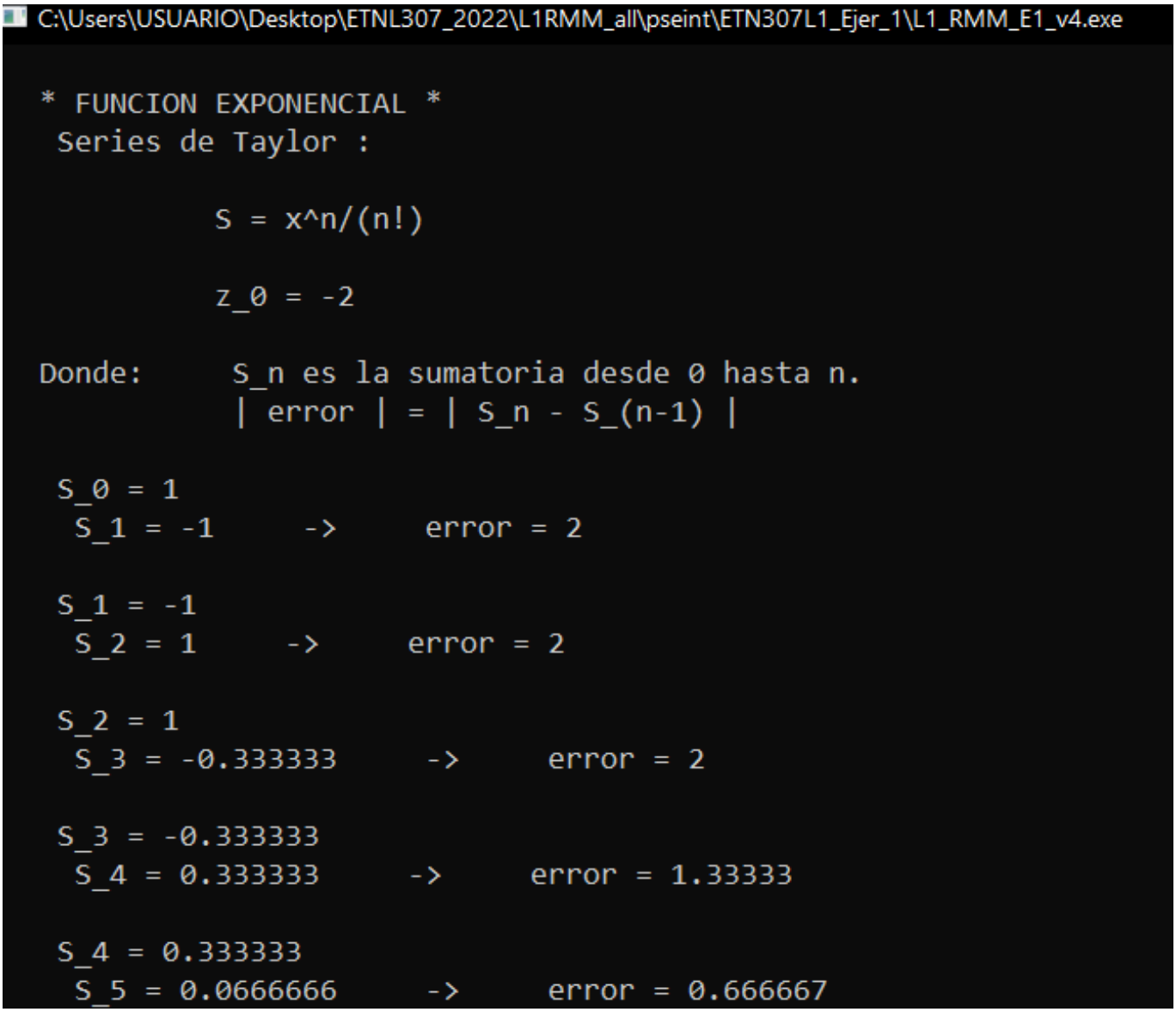
Para la 1ra opción.



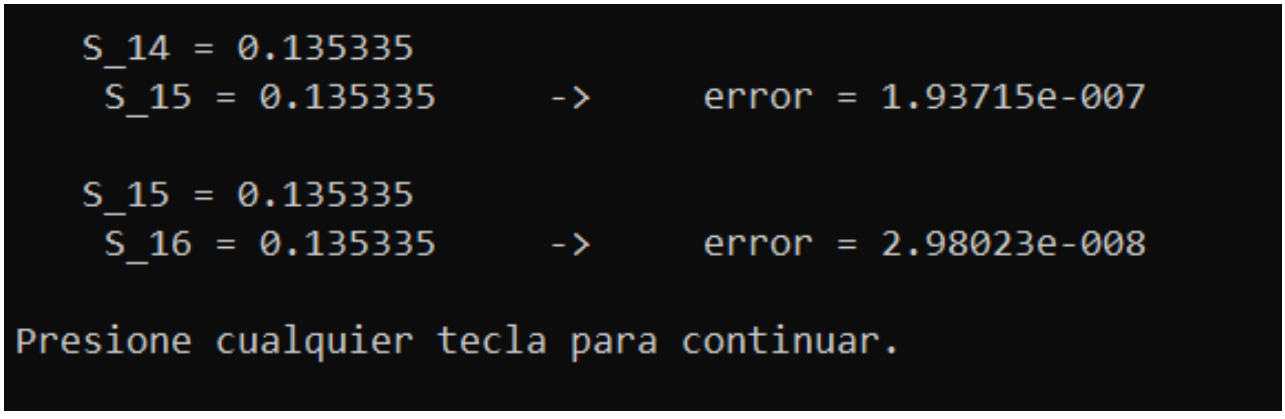
Para la 2da opción.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_1\L1_RMM_

* FUNCION EXPONENCIAL *
 Series de Taylor :

          S = x^n/(n!)

          z_0 = -2

Donde :     t_n es el termino n-esimo

t_0 = 1
t_1 = -2
t_2 = 2
t_3 = -1.33333
t_4 = 0.666667
t_5 = -0.266667
t_6 = 0.0888889
t_7 = -0.0253968
t_8 = 0.00634921
t_9 = -0.00141093
t_10 = 0.000282187
t_11 = -5.13067e-005
t_12 = 8.55112e-006
t_13 = -1.31556e-006
t_14 = 1.87937e-007
t_15 = -2.50582e-008

S = 0.135335
```

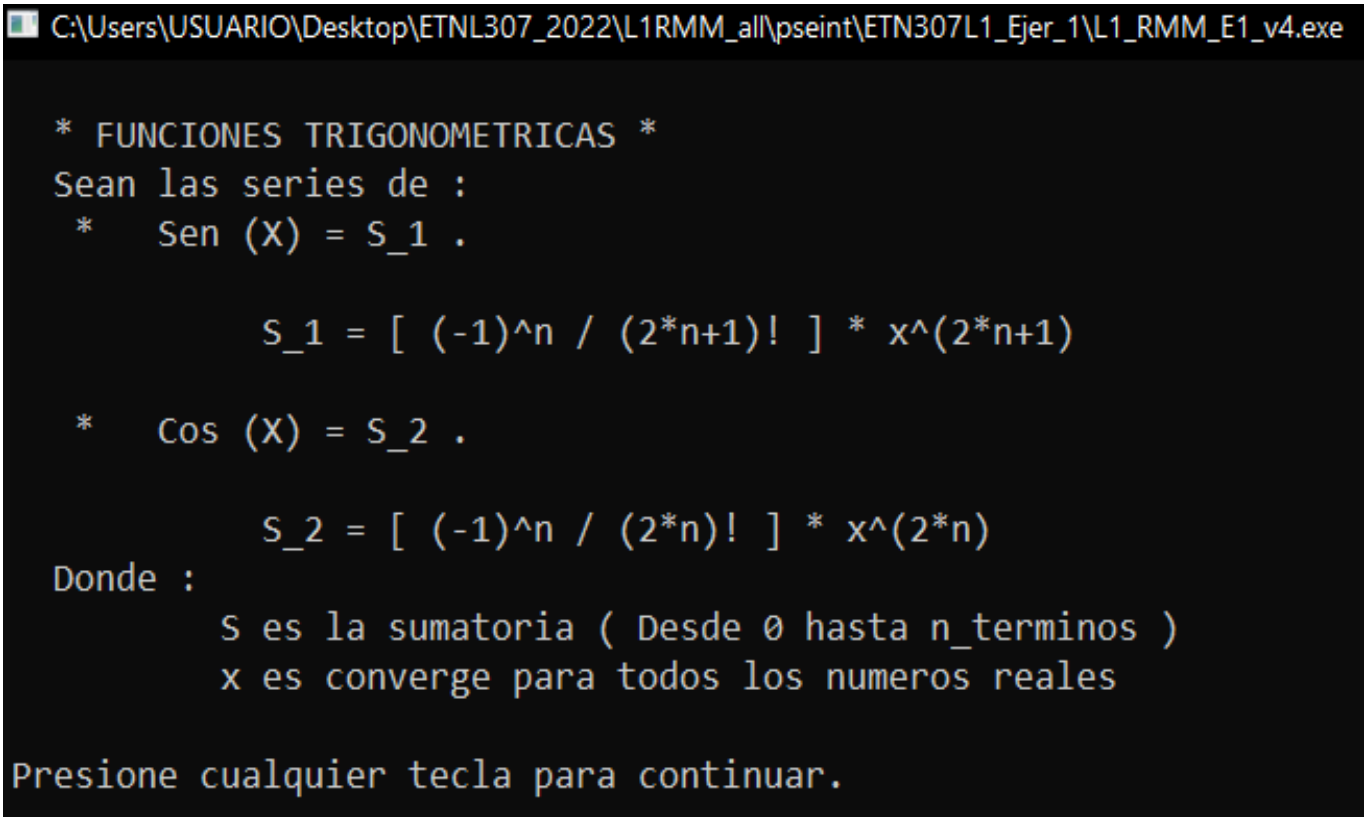
Para la 3ra opción.



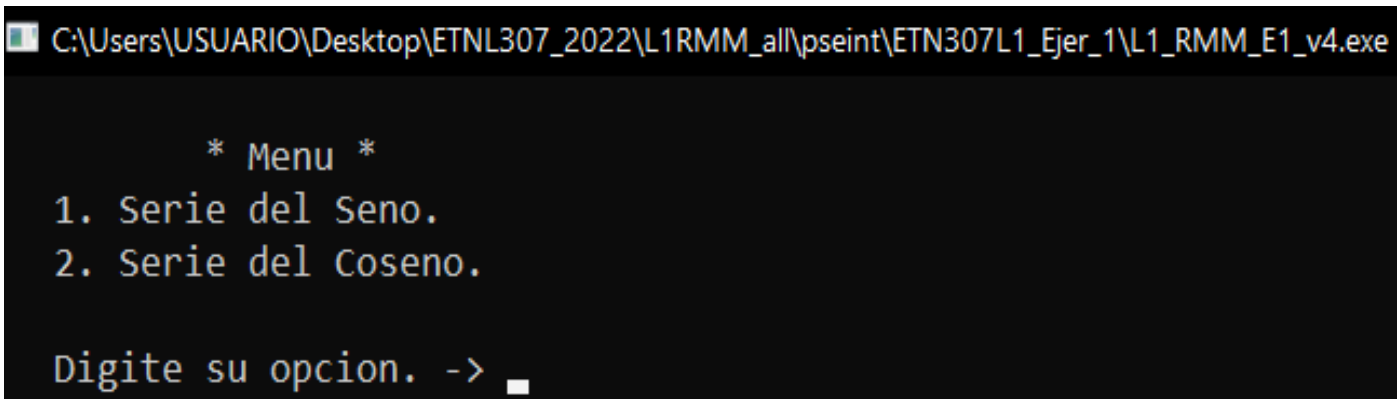
Recortando la imagen.



- Ingresando a la Serie trigonométrica.



Luego de presionar cualquier tecla, ingresará al menú para seleccionar entre la serie de Taylor y Maclaurin para este caso se escogerá la serie de Taylor, luego aparecerá.



Escogiendo la serie del seno, luego preguntara que punto de convergencia desea y para este caso serán igual a 80 grado; luego ingresando en el menú de mostrar resultados.

Para la opción de ver el valor de la sumatoria.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_1\L1_RMM_E1_v4.exe

  * FUNCIONES TRIGONOMETRICAS *
   Series de Taylor :
   *   Sen (X) = S_1 .

            S_1 = [ (-1)^n / (2*n+1)! ] * x^(2*n+1)

            z_0 = 80 ‖ , z_0 = 1.39626 [rad]


     S = 0.984808

Presione cualquier tecla para continuar.
```

Para la opción de ver los n términos.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_1\L1_RMM_E1_v4.exe

  * FUNCIONES TRIGONOMETRICAS *
   Series de Taylor :
   *   Sen (X) = S_1 .

            S_1 = [ (-1)^n / (2*n+1)! ] * x^(2*n+1)

            z_0 = 80 ‖ , z_0 = 1.39626 [rad]

  Donde :    t_n Termino n-esimo

  t_0 = 1.39626
  t_1 = -0.453681
  t_2 = 0.0442237
  t_3 = -0.00205277
```

Cortando la imagen.

```
  t_14 = 1.80931e-027
  t_15 = -3.79285e-030
  t_16 = 7.00223e-033

    S = 0.984808

Presione cualquier tecla para continuar.
```

Para la opción de ver el procedimiento.

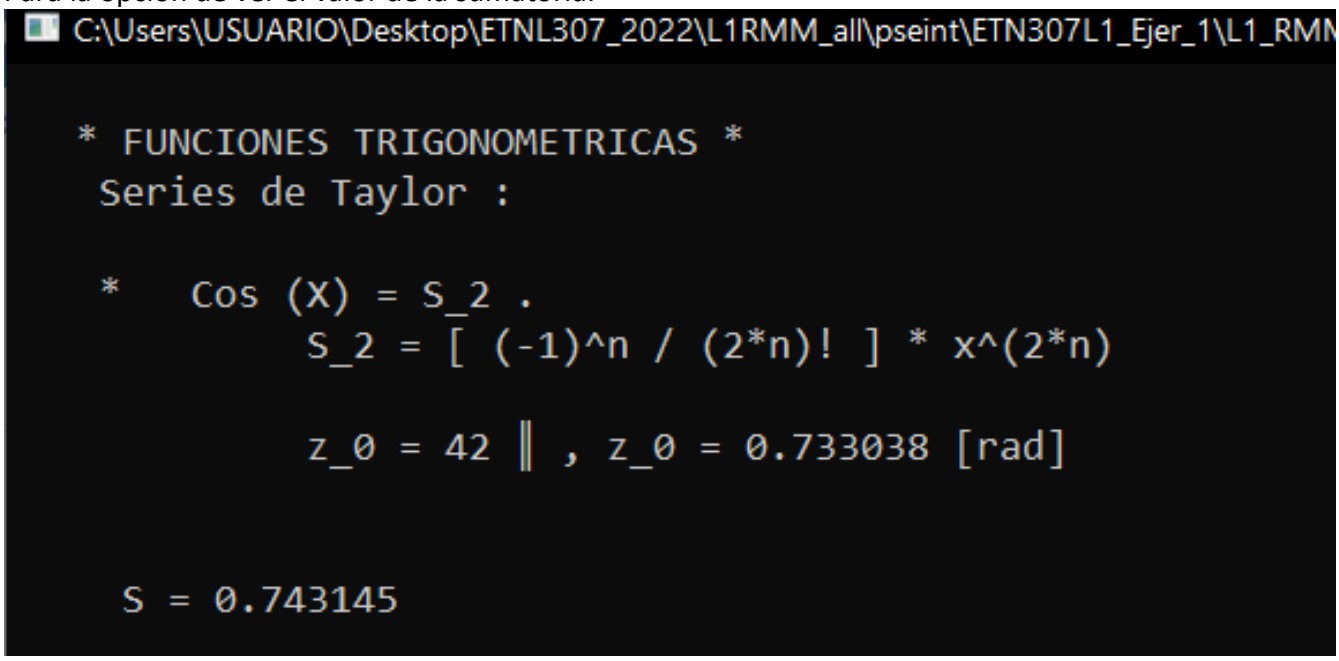
```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_1\L1_RMM_E1_v4.exe

* FUNCIONES TRIGONOMETRICAS *
 Series de Taylor :
 *   Sen (X) = S_1 .

          S_1 = [ (-1)^n / (2*n+1)! ] * x^(2*n+1)

          z_0 = 80 ‖ , z_0 = 1.39626 [rad]

 Donde:     t_n es el termino n-esimo.
                t_n -> 0

   t_0 = 1.39626

   t_1 = -0.453681

   t_2 = 0.0442237
```

Cortando la imagen.

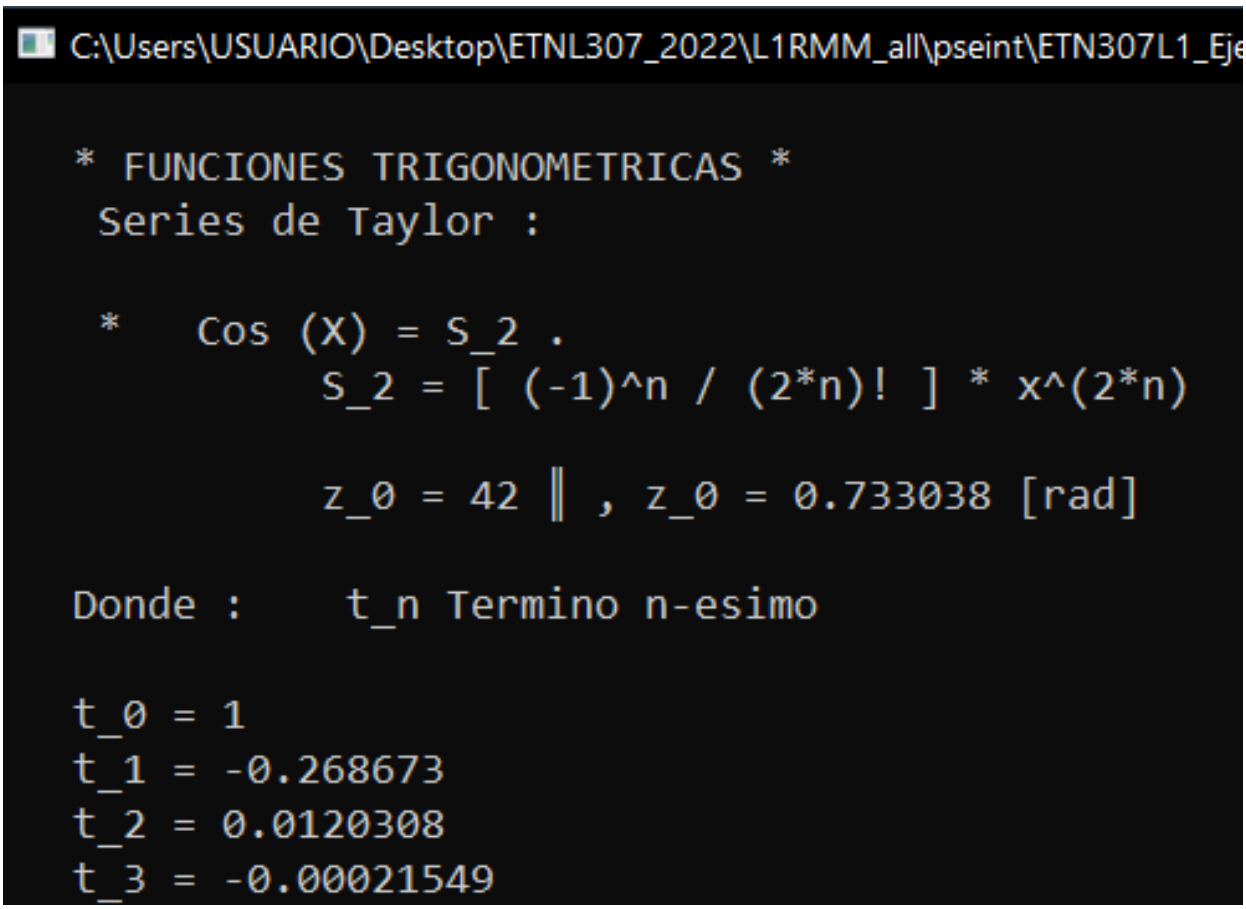
```
    t_15 = -3.79285e-030

    t_16 = 7.00223e-033

        t_17 = 0

Presione cualquier tecla para continuar.
```

Escogiendo la serie del coseno, luego preguntara que punto de convergencia desea y para este caso serán igual a 42 grados; luego ingresando en el menú de mostrar resultados.

Para la opción de ver el valor de la sumatoria.



Para la opción de ver los n-términos.



Recortando la imagen.

```
t_15 = -3.38954e-037
t_16 = 1.83604e-040
t_17 = -8.82818e-044

  S = 0.743145

Presione cualquier tecla para continuar.
```

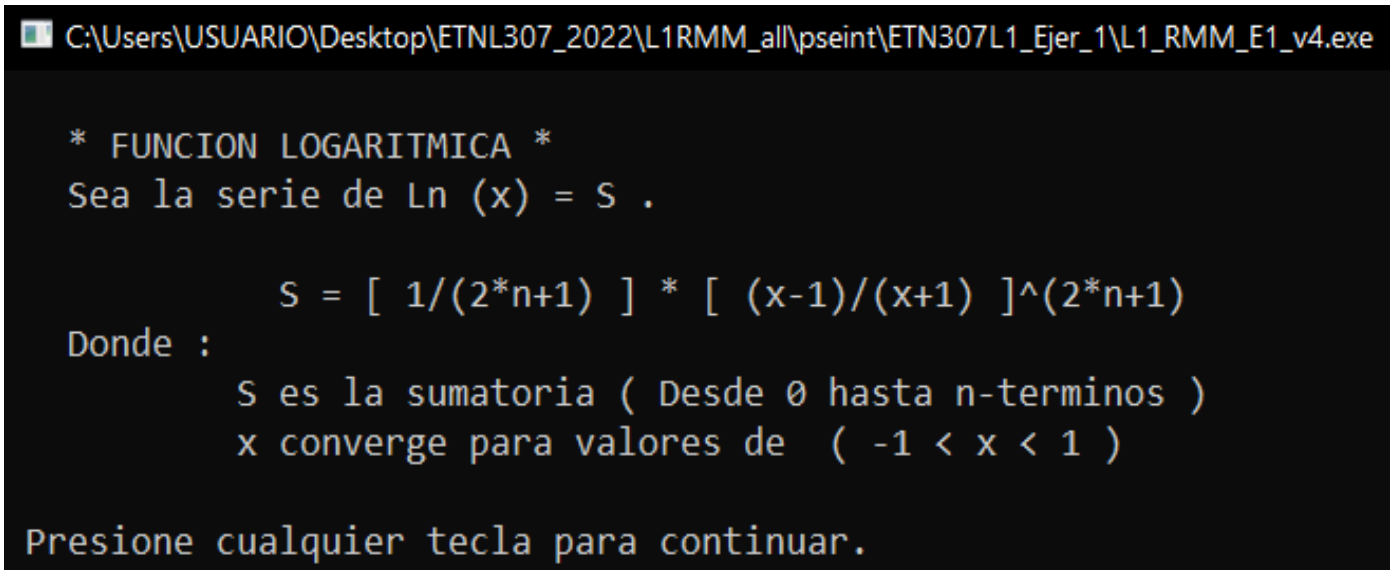
Para la opción de ver el procedimiento.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_1\L1_RMM_E1_v4.exe

* FUNCIONES TRIGONOMETRICAS *
 Series de Taylor :

 *   Cos (X) = S_2 .
          S_2 = [ (-1)^n / (2*n)! ] * x^(2*n)

          z_0 = 42 ‖ , z_0 = 0.733038 [rad]

Donde:     t_n es el termino n-esimo.
                 t_n -> 0

  t_0 = 1

  t_1 = -0.268673

  t_2 = 0.0120308
```

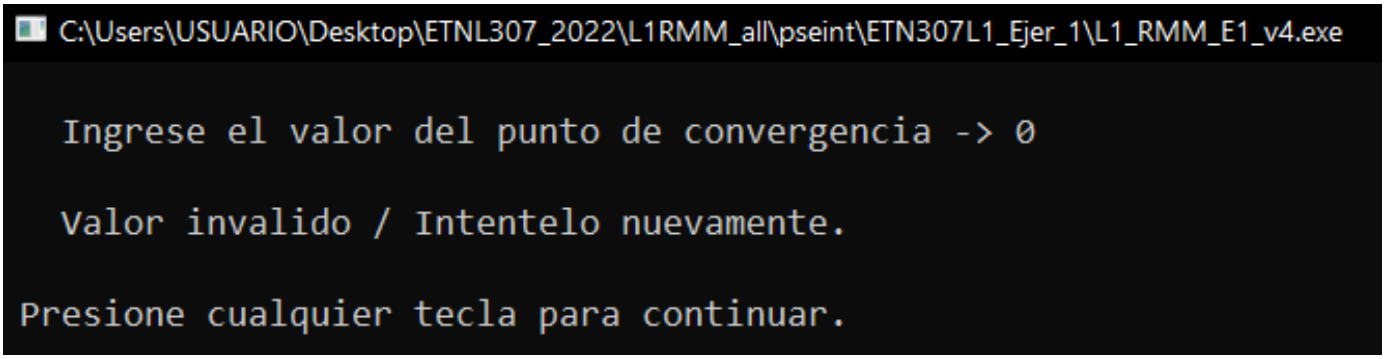
Recortando la imagen.

```
  t_16 = 1.83604e-040

  t_17 = -8.82818e-044

      t_18 = 0

Presione cualquier tecla para continuar.
```

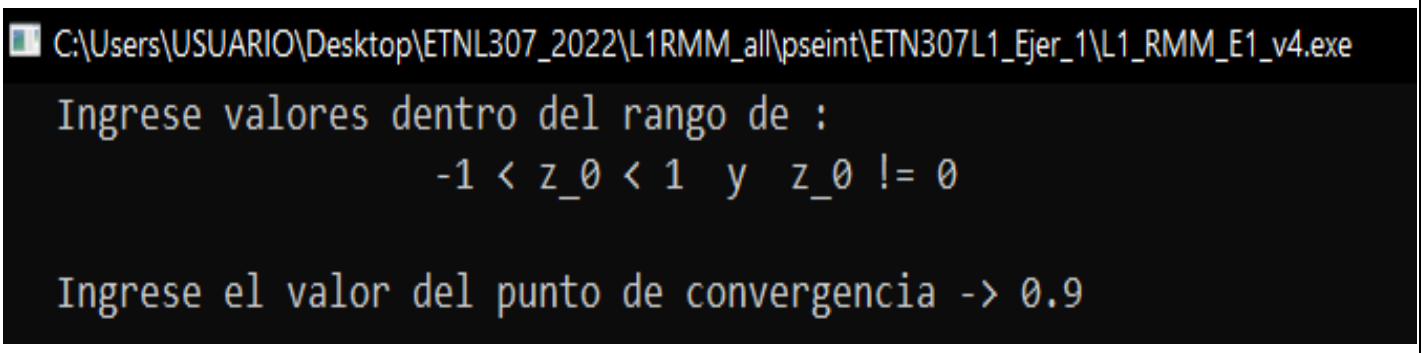
- Ingresando a la serie logarítmica.



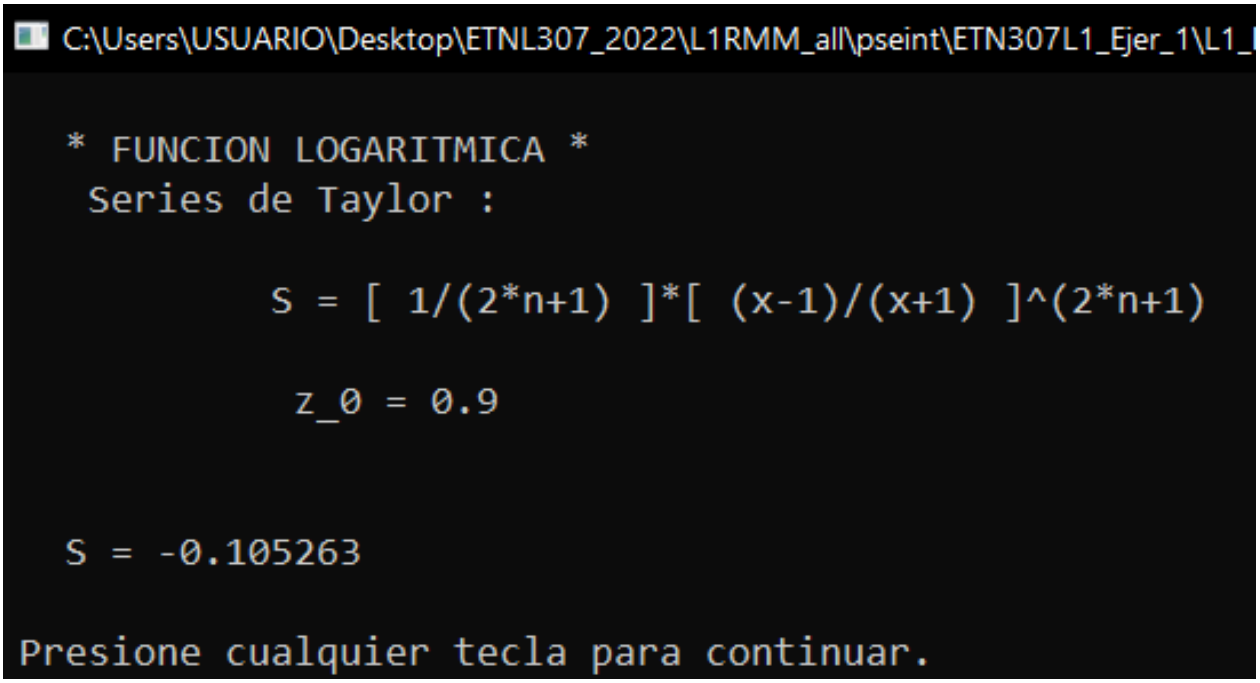
Ingresando valores erróneos para el punto de convergencia.



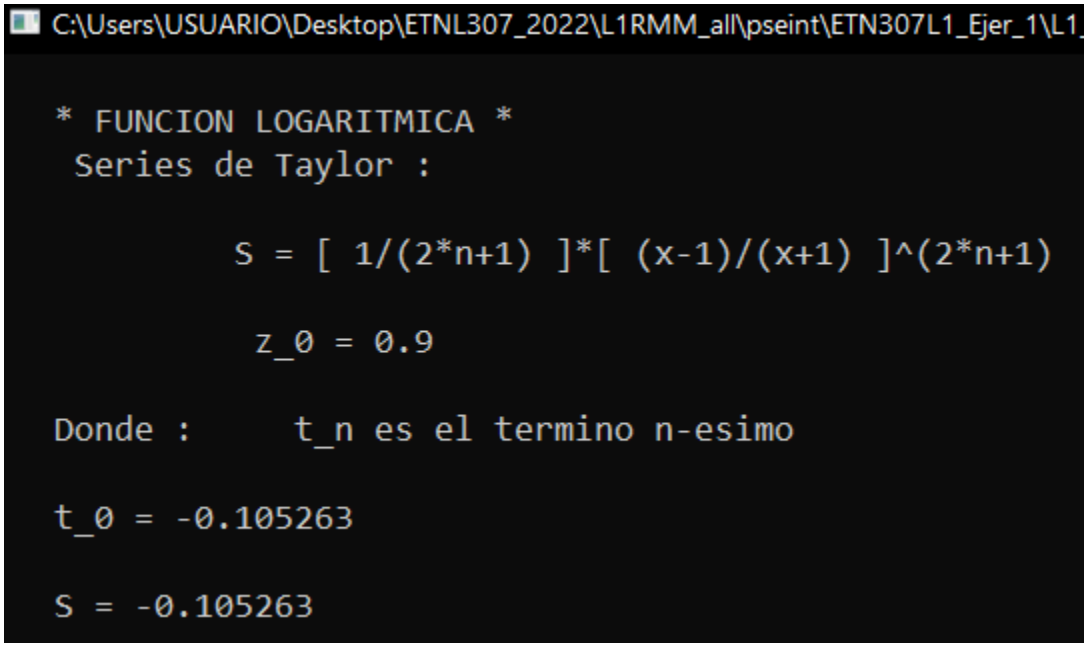
Presionando cualquier tecla, y luego ingresando un valor correcto.



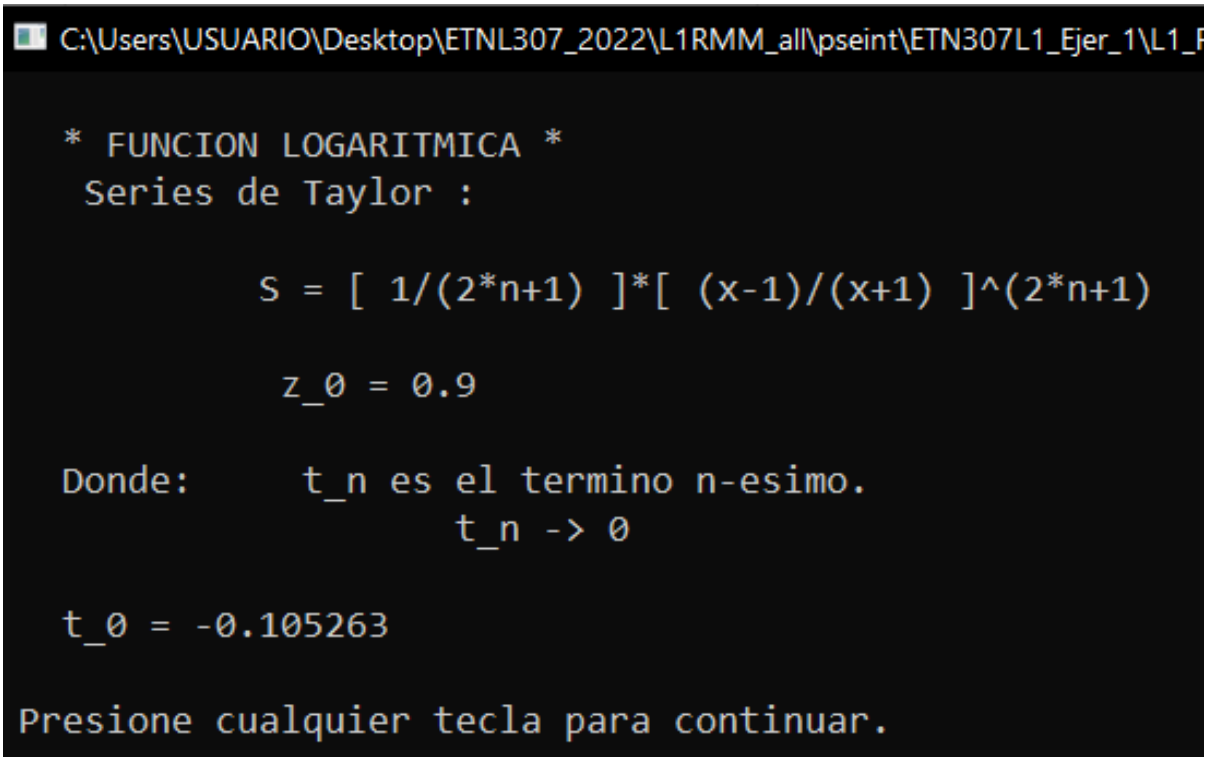
Ingresando a la opción de ver el valor de la sumatoria.



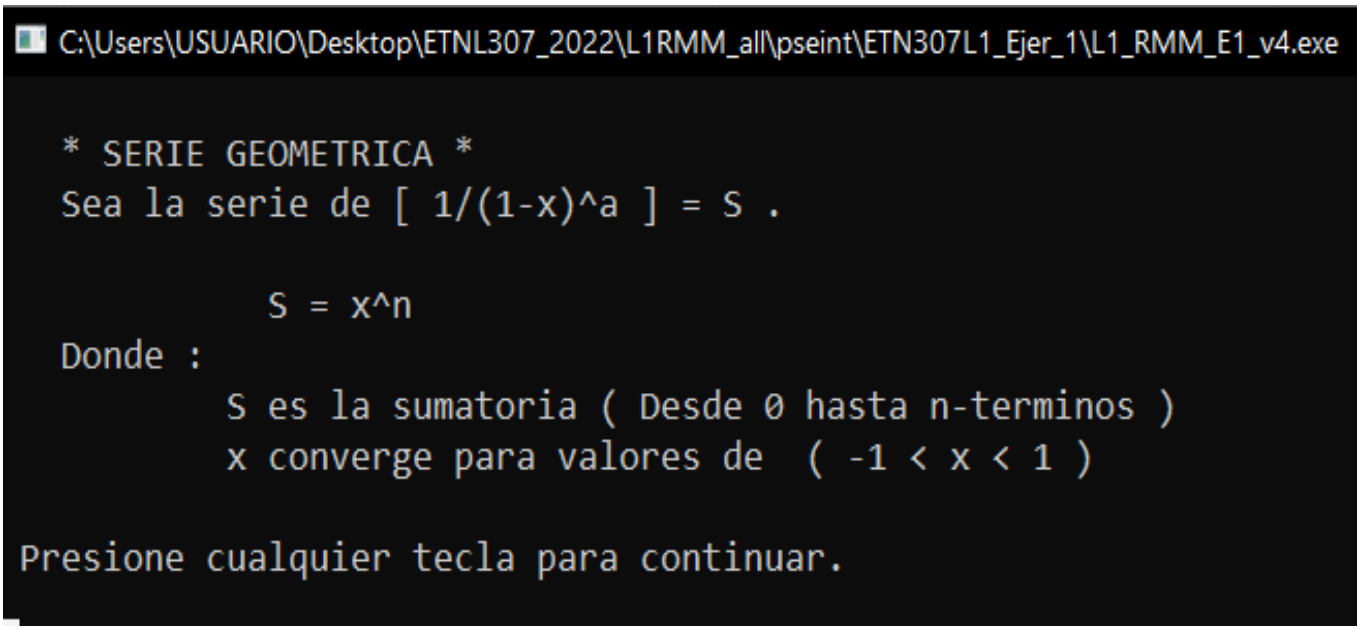
Ingresando a la opción de ver los n-términos.



Ingresando a la opción de ver el procedimiento.

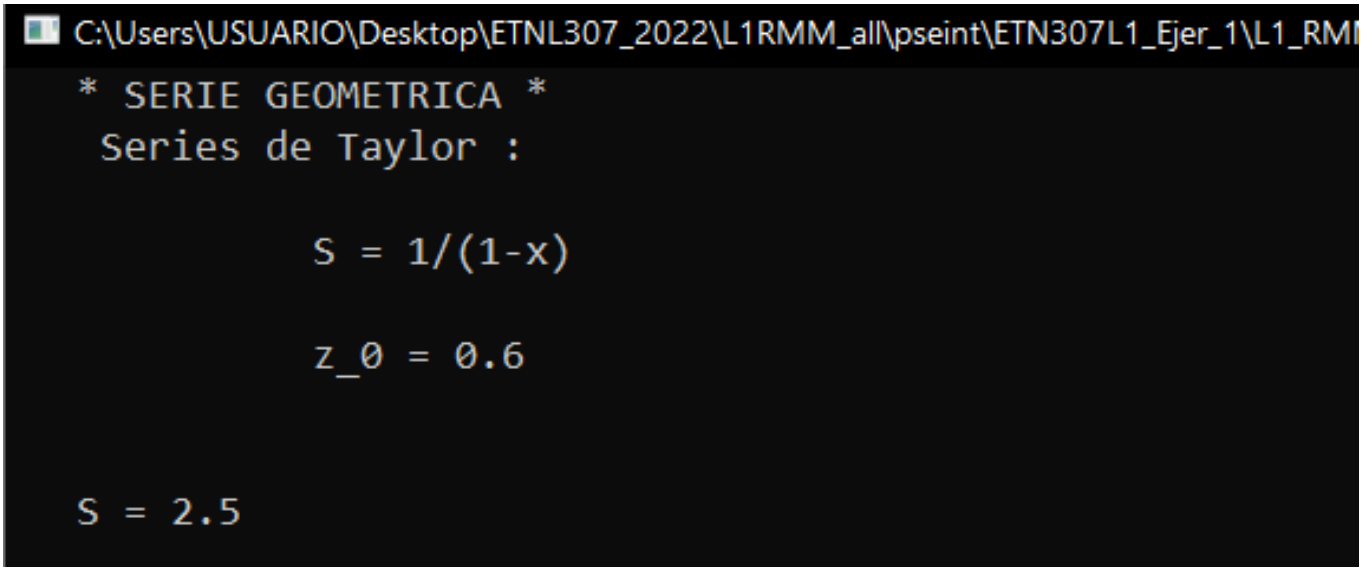


- Ingresando a la serie geométrica.

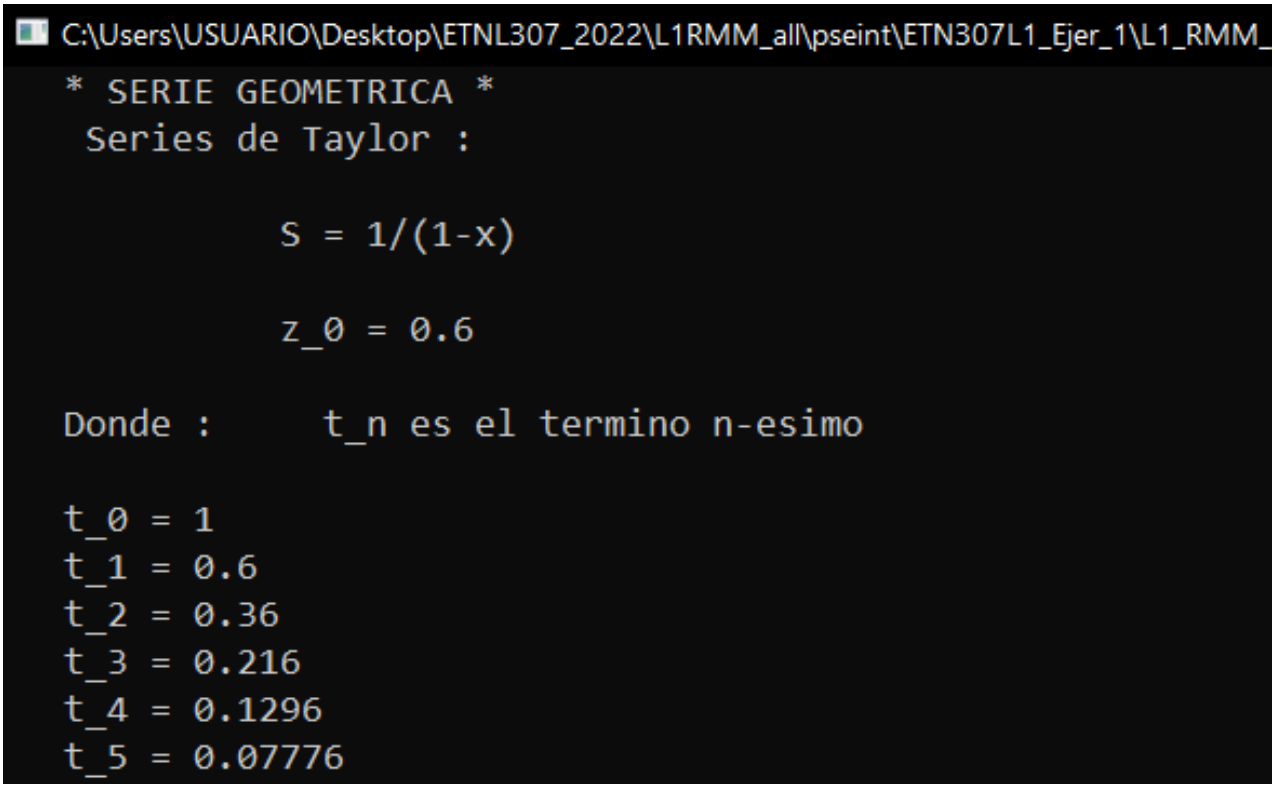


Escogiendo a la serie de Taylor para luego ingresar el valor del punto de convergencia igual a 0,6.

Ingresando a la opción de ver el valor de la sumatoria.



Ingresando a la opción de ver los n-términos.



Recortando la imagen.

```
t_202 = 1.4013e-045
t_203 = 1.4013e-045
t_204 = 0

S = 2.5
```

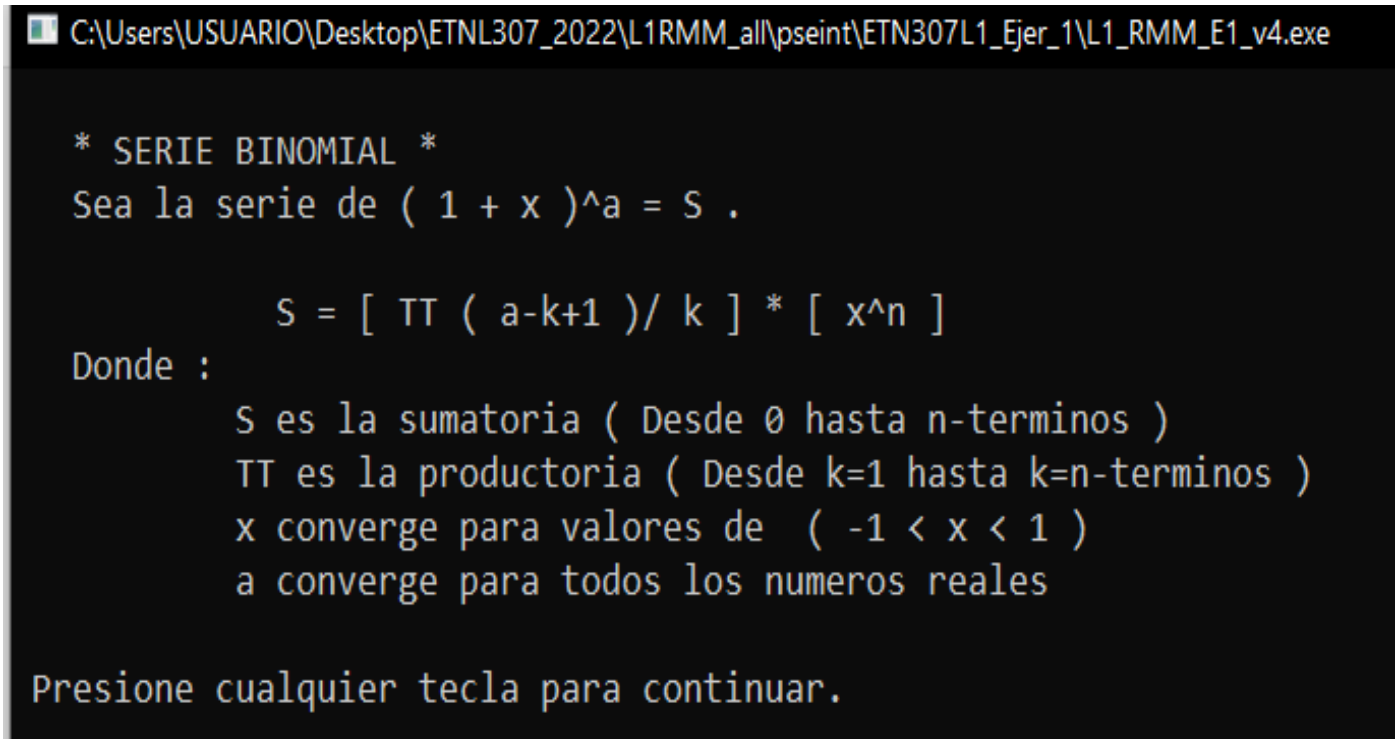
Ingresando a la opción de ver el procedimiento.

```
Seleccionar C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_1
 * SERIE GEOMETRICA *
  Series de Taylor :

           S = 1/(1-x)

           z_0 = 0.6

 Donde:     t_n es el termino n-esimo.
                  t_n -> 0

   t_0 = 1

   t_1 = 0.6

   t_2 = 0.36
```

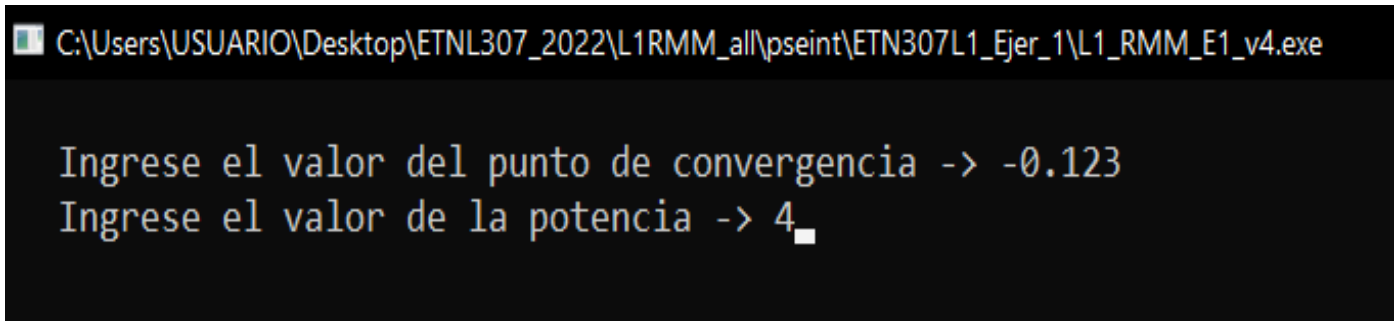
Recortando la imagen.

```
   t_202 = 1.4013e-045

   t_203 = 1.4013e-045

       t_204 = 0
```

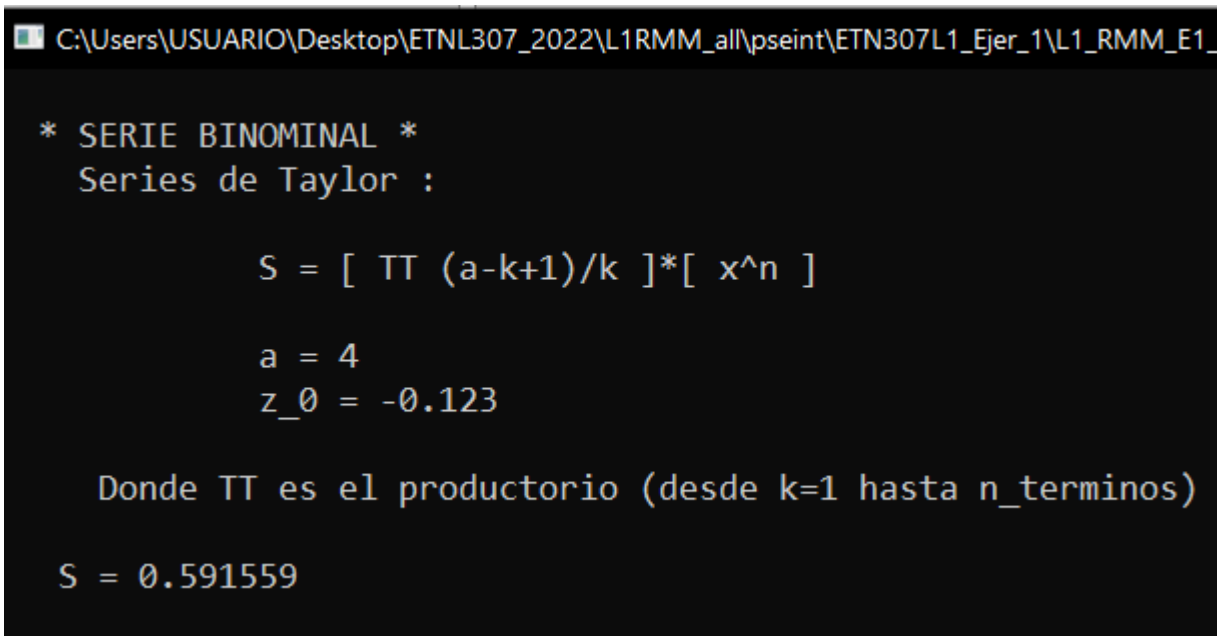
- Ingresando a la serie binomial



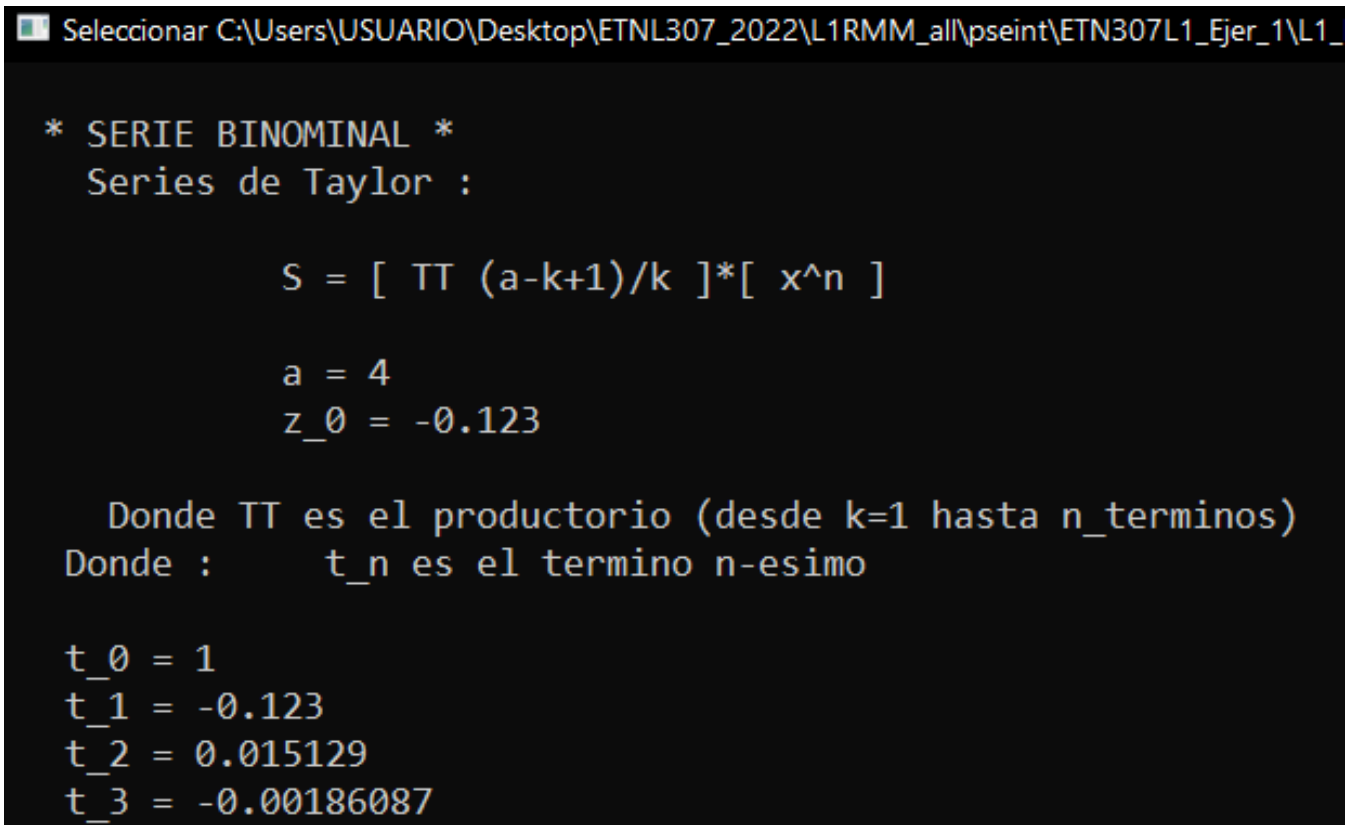
Escogiendo a la serie de Taylor para luego ingresar el valor del punto de convergencia igual a -0,123 además una potencia igual a 4.



Ingresando a la opción de ver el valor de la sumatoria.



Ingresando a la opción de ver los n-términos.



Recortando la imagen.

```
t_48 = 2.10195e-044
t_49 = -2.8026e-045
t_50 = 0

S = 0.591559
```

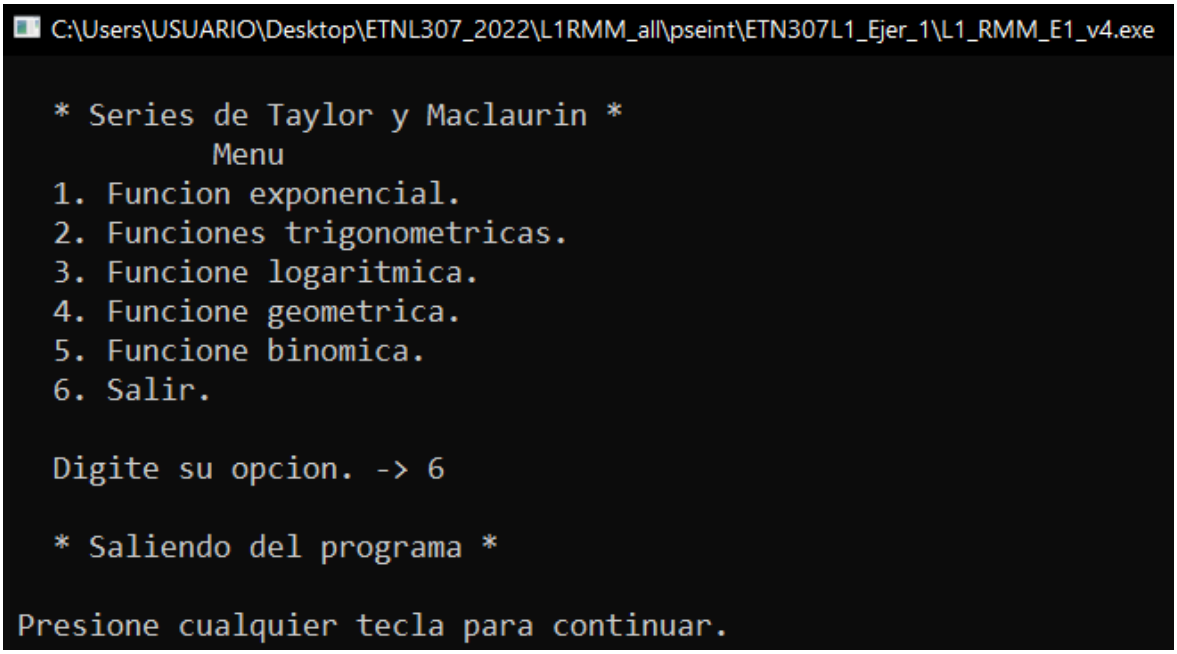
Ingresando a la opción de ver el procedimiento.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_1\L1_RMM_E1_v4.exe

* SERIE BINOMINAL *
  Series de Taylor :

           S = [ TT (a-k+1)/k ]*[ x^n ]

           a = 4
           z_0 = -0.123

   Donde TT es el productorio (desde k=1 hasta n_terminos)
 Donde:     t_n es el termino n-esimo.
                  t_n -> 0
   t_0 = 1
   t_1 = -0.123
   t_2 = 0.015129
   t_3 = -0.00186087
```

Recortando la imagen.

```
t_47 = -1.68156e-043
t_48 = 2.10195e-044
t_49 = -2.8026e-045
```

- Finalmente escogiendo salir de menú principal.



Conclusiones. -

El algoritmo realizo bien la mayoría de las tareas, a excepción de la serie de la función logarítmica ya que en esta los valores que el programa entregaba fueron algo cercanos, pero en ningún punto de convergencia probado entrego un valor correcto.

Observaciones. -

Para el caso de la serie logarítmica talvez la razón por la que no funciona correctamente sea por las restricciones de su radio de convergencia ya que además de encontrarse entre -1<z_0<1, también debía evitarse el punto con el valor igual a '0'.

Notando que el punto de convergencia era evaluado por la función_análisis_dominio, se realizaron distintos cambios para poder obtener el valor correcto de la sumatoria, sin embargo, ningún cambio fue exitoso.

Cabe aclarar que la función_analisia_dominio funciono con éxito para los subprogramas que realizaban las series geométrica y binomial (ambas con un radio del punto de convergencia).

- **ECUACION DE 2DO GRADO**

Marco teórico. –

Una ecuación de 2do grado también llamada ecuación cuadrática tiene la forma:

$$ax^2 + bx + c = 0\ , \quad a \neq 0$$

Dónde: x es la variable.
a es el coeficiente cuadrático (distinto de cero).
b el coeficiente lineal.
c es el término independiente.
a, b y c pertenecen a los números reales

- Demostración:

  Lo primero es dividir la entre el coeficiente cuadrático.

  $$\frac{ax^2}{a} + \frac{b}{a}x + \frac{c}{a} = 0$$

  Simplificando.

  $$x^2 + \frac{b}{a}x + \frac{c}{a} = 0$$

  Despejamos el termino independiente

  $$x^2 + \frac{b}{a}x = -\frac{c}{a}$$

  Sumamos en ambos miembros b^2/(4*a^2)

  $$x^2 + \frac{b}{a}x + \frac{b^2}{4a^2} = \frac{b^2}{4a^2} - \frac{c}{a}$$

  Se puede observar un cuadrado perfecto la parte derecha de la ecuación

  $$(x + \frac{b}{2a})^2 = \frac{b^2-4ac}{4a^2}$$

  Despejando x se obtiene

  $$x + \frac{b}{2a} = \frac{\pm\sqrt{b^2-4ac}}{2a}$$

  Finalmente, la ecuación será:

  $$x = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

- Cabe destacar el análisis de la discriminante la cual es:

$$b^2-4ac$$

i. Discriminante positiva resultara en dos soluciones reales distintas.

ii. Discriminante igual a cero resultara en dos soluciones reales pero iguales.

iii. Discriminante negativa resultara en dos soluciones imaginarias conjugadas.

$$Discriminante = \Delta \quad \rightarrow \quad \begin{cases} \Delta > 0 \\ \Delta = 0 \\ \Delta < 0 \end{cases}$$

❖ <u>Ejemplos prácticos.</u> –

i. Si $\Delta > 0$

Sea la ecuación:

$$1x^2 + 3x - 4 = 0$$

Resuelto por factorización tiene como raíces:

$$x_1 = 1 \quad ; \quad x_2 = -4$$

Otro ejemplo:

$$1x^2 + 10x + 1 = 0$$

Resuelto por la formula general cuadrática.

$$x_1 = -0{,}1010205144 \quad ; \quad x_2 = -9{,}898978486$$

ii. Si $\Delta = 0$

Sea la ecuación:

$$1x^2 + 1x + 0.25 = 0$$

Resuelto por la formula general cuadrática.

$$x_{1,2} = -0{,}5$$

Otro ejemplo:

$$0.25x^2 + 1x + 1 = 0$$

Resuelto por la formula general cuadrática.

$$x_{1,2} = -2$$

iii. Si $\Delta < 0$

Sea la ecuación:

$$1x^2 + 2x + 5 = 0$$

Resuelto por la formula general tiene como raíces imaginarias conjugadas:

$$x_{1,2} = -1 + 2i$$

Otro ejemplo:

$$1x^2 + 1x + 10 = 0$$

Resuelto por la formula general tiene como raíces imaginarias conjugadas:

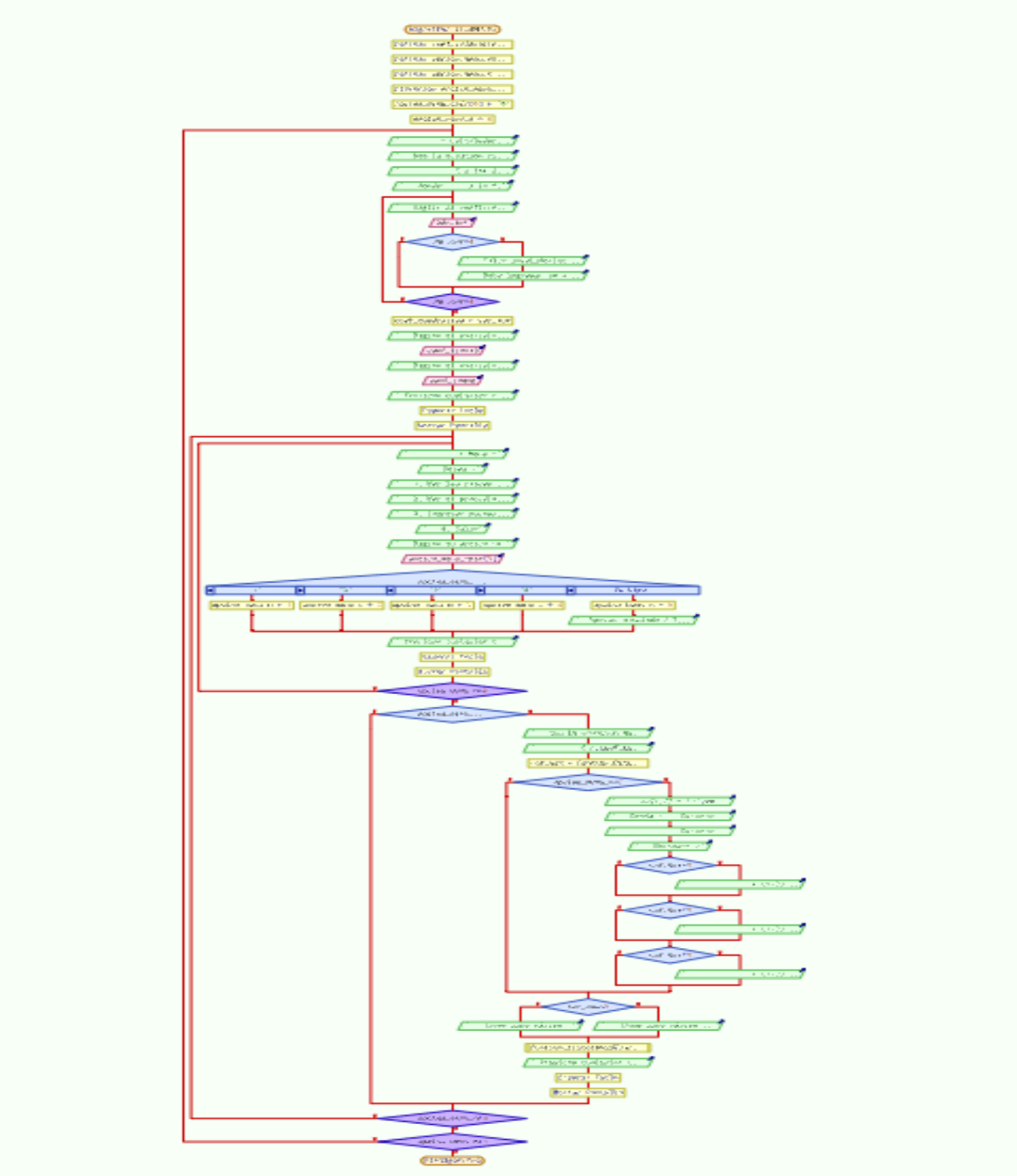
$$x_{1,2} = -0{,}5 \pm 3{,}122498999i$$

Diagramas de flujo. –

Para dibujar los diagramas de flujo de los algoritmos se empleó el programa “PSeInt”, y como estos están conformados por funciones se muestran de la siguiente manera; Además para poder visualizar de manera detallada se realizarán distintos acercamientos a cada diagrama de flujo.

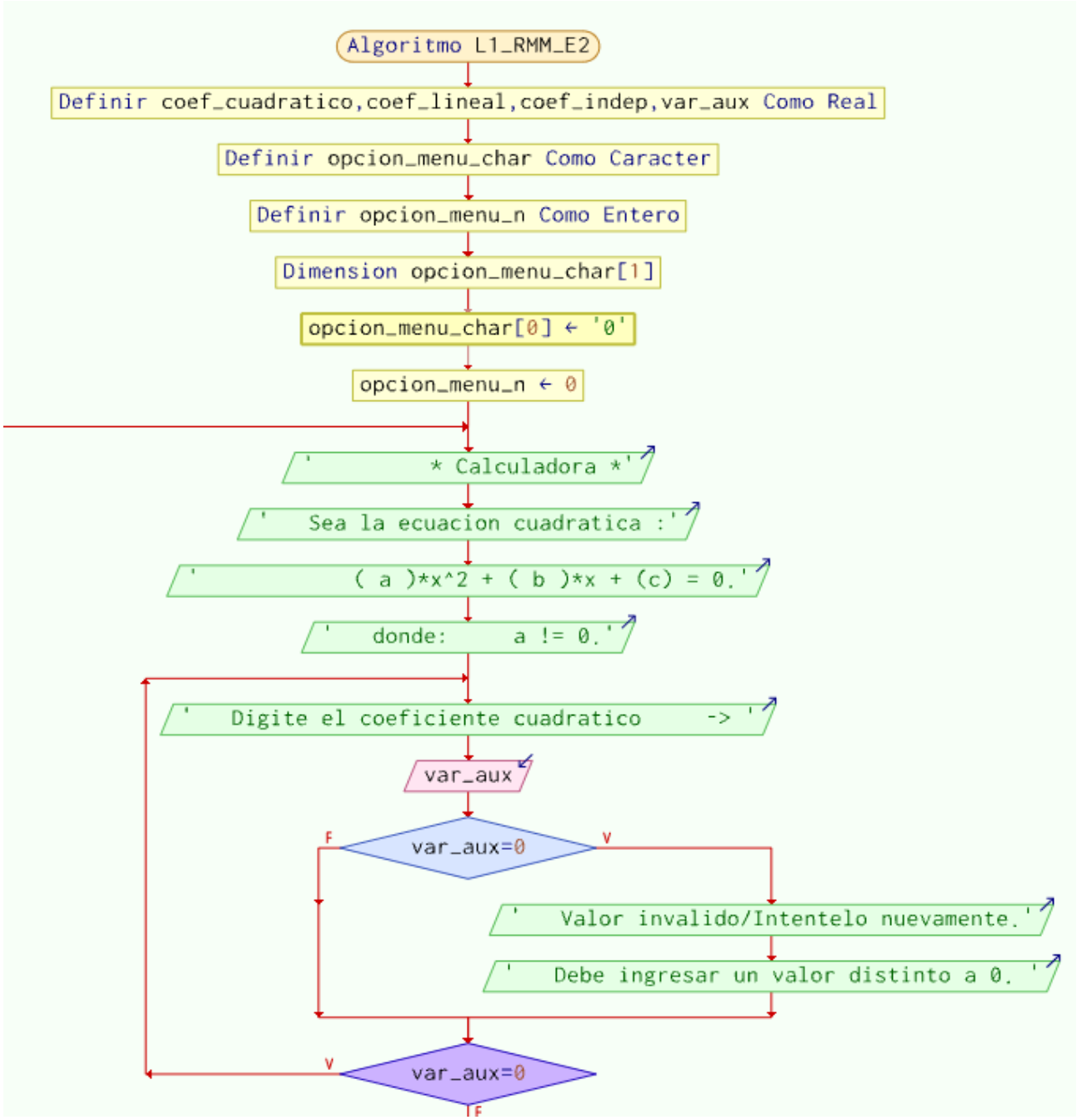
- Series de Taylor y Maclaurin (L1_RMM_EJ_2 es el nombre del archivo)

- Diagrama principal



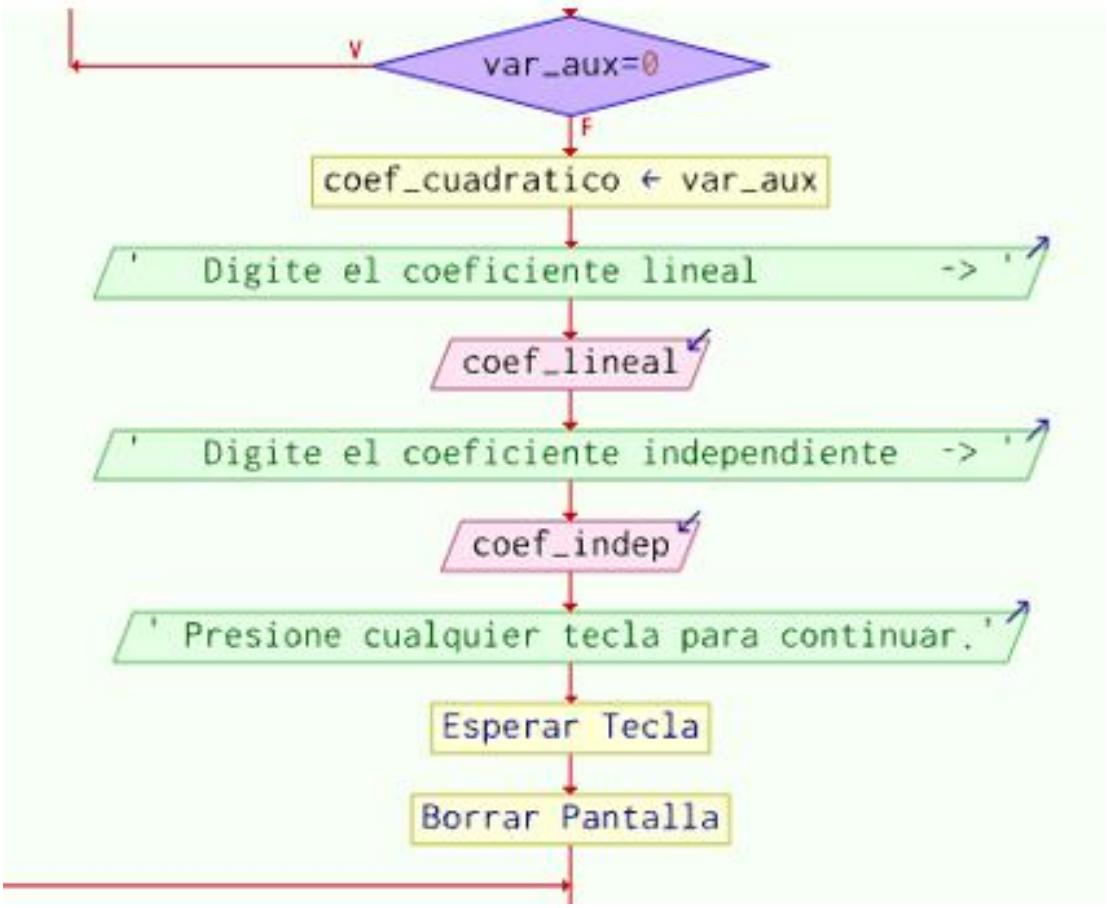
Zoom (superior del diagrama de flujo principal).

- Inicio del algoritmo: el algoritmo se llama L1_RMM_EJ_2
- Definir variables: Opcion_menu_char para que el usuario ingrese cualquier valor y el programa no tropiece con opciones letras u opciones que no se encuentran en le menú, opicon_menu_n para cuando si existe la opción ingresada tome un valor numérico entero.
- Se inicia un ciclo do{}while() que termina si y solo si el usuario digita la opción de salir.
- Impresión en la pantalla: Muestra el menú de inicio y pide ingresar su opción.



- Se inicia otro ciclo do{}while() para evaluar que el coeficiente cuadrático no tenga el valor de cero.
- Se ingrese el valor de una variable auxiliar, para la tarea ya mencionada.
- Condicional if(){}else{} , para mostrar en la consola un mensaje(si el valor del coef. Cuadrático fuera cero)
- Condicional While(), para terminar el proceso repetitivo si el valor del coef. Cuadrático fuera de 0

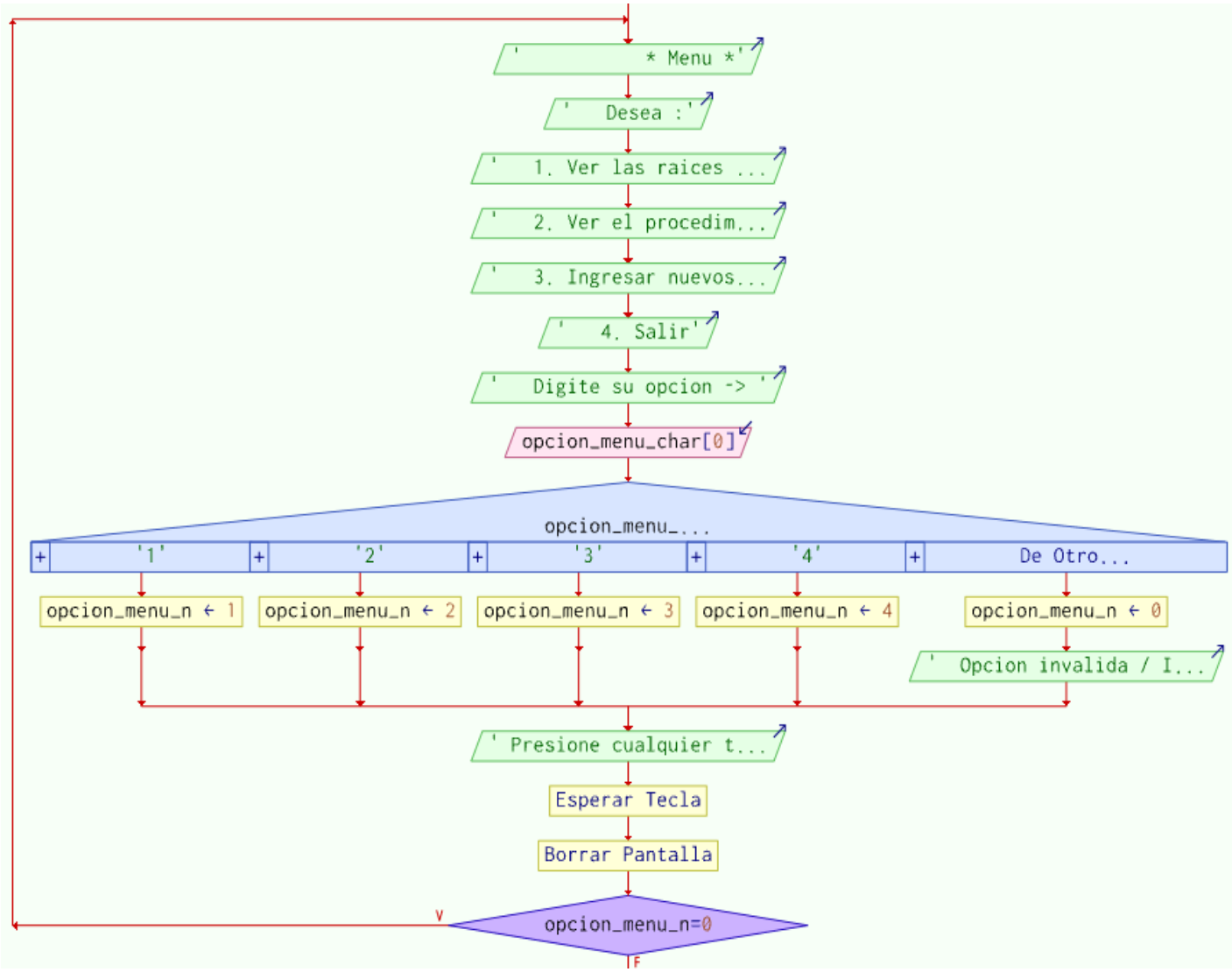
Zoom (fuera de la condicional while).



- Si el valor ingresado anteriormente fuera distinto de cero, se guardara en la variable coef_cuadratico.
- Se piden los valores de los demás coeficientes, como también en cada variable se le guarda el valor que le corresponde
- Se mostrará un mensaje, luego se limpia la consola.
- Se inicia dos ciclos do{}while(), para que se retorne continuamente al menú.

Zoom (menú).

- Se mostrará el menú, para luego pedir la opción que se desea ejecutar(guardando la opción en una variable tipo char).

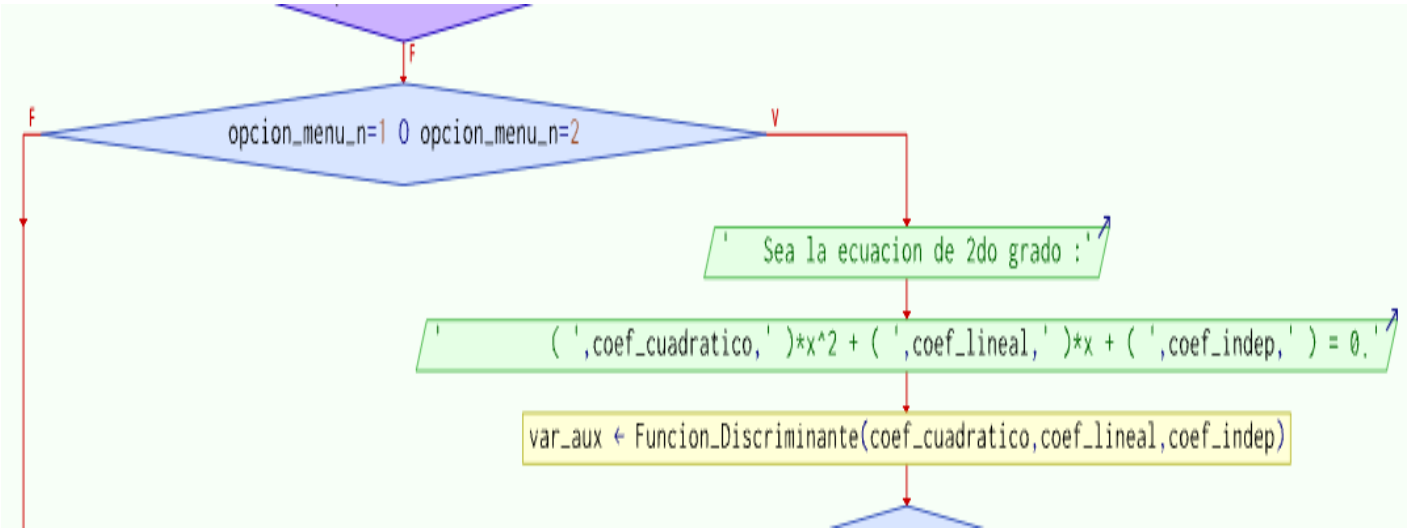


- Condicional switch, como se ve en el diagrama, si es ingresado una opción invalida (letras, etc.) la variable opción_menu_n tomara el valor de 0; Sin embargo, si el valor ingreso si está dentro de las opciones, la variable numérica tomara el valor de la opción requerida.

- Si fuera el caso de 'opción_menu_n = 0' la condicional while evitaría que el usuario pase del menu.

Zoom (condicional if).

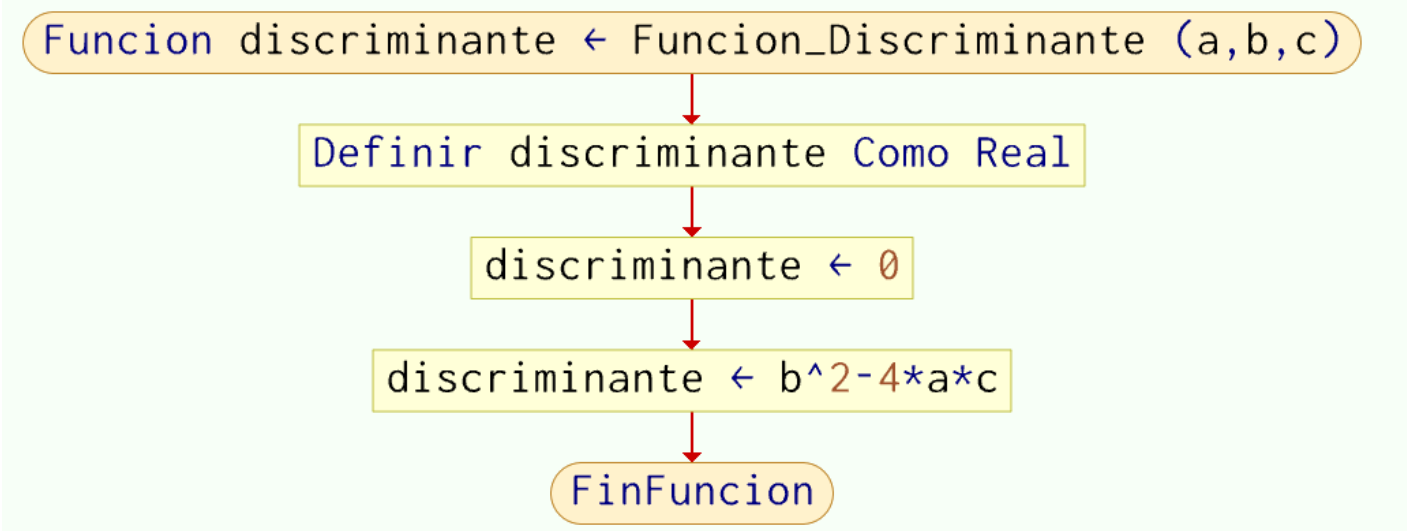
- La condicional if(){}else{}, que se ve en esta parte del diagrama mostrara o no el procedimiento.
- Cabe aclara que si fuera el caso (opción_menu_n = 2), la variable numérica var_aux tomaría el valor retornado por la Función _Discriminante.



- Antes de continuar será necesario notar que la función_Discriminante depende de los 3 valores, ingresados ya que esta tendrá por formula:

$$Discriminante = b^2 - 4 * a * c$$

- Funcion_Discriminante, es llamada y por ello mostraremos si propio diagrama de flujo.

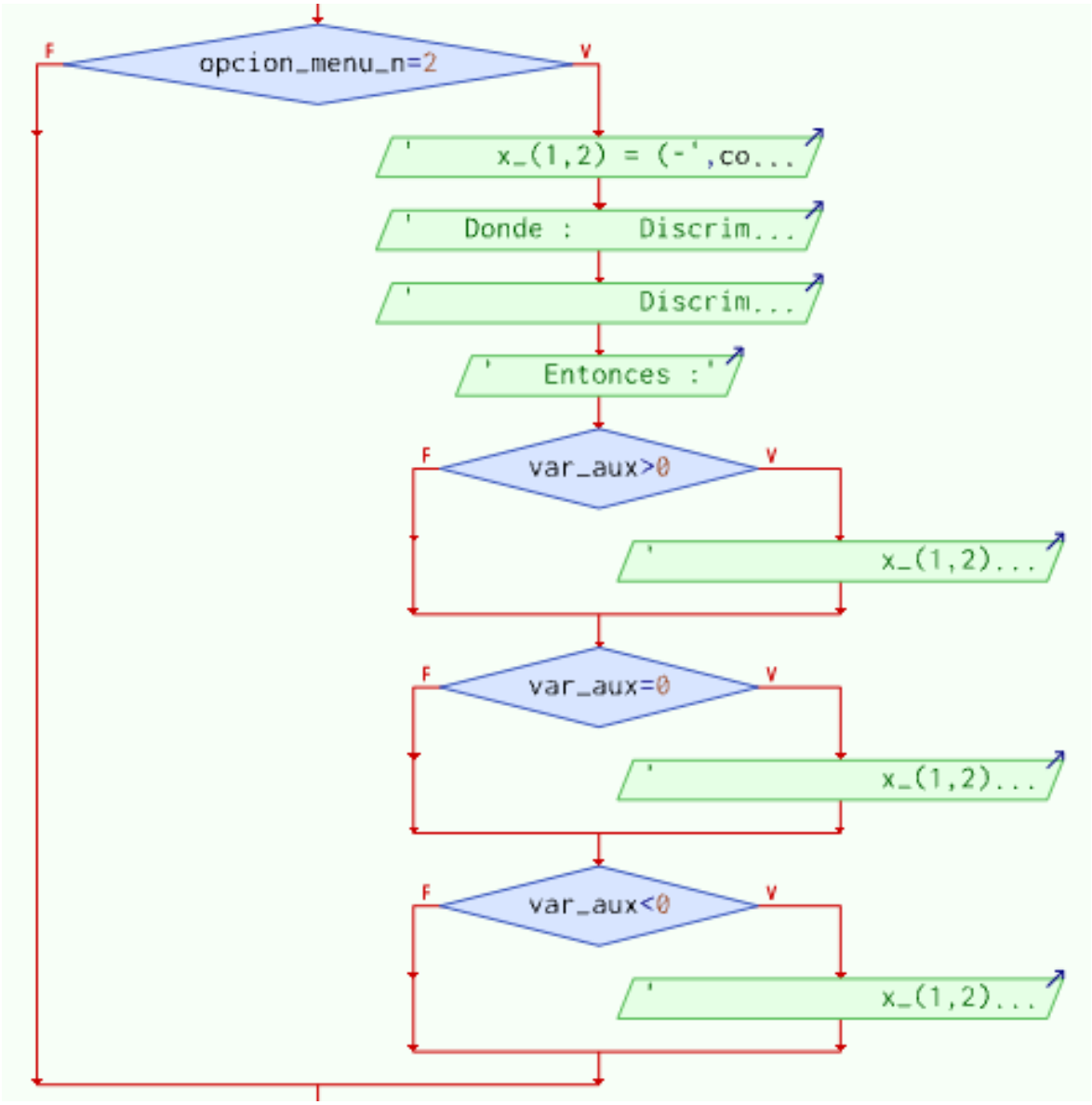


- Realiza la tarea matemática de la formula ya mencionada, para luego devolver el valor calculado.

Saliendo de la Función y retornando al diagrama principal en la parte que continua.

Zoom (condicional if).

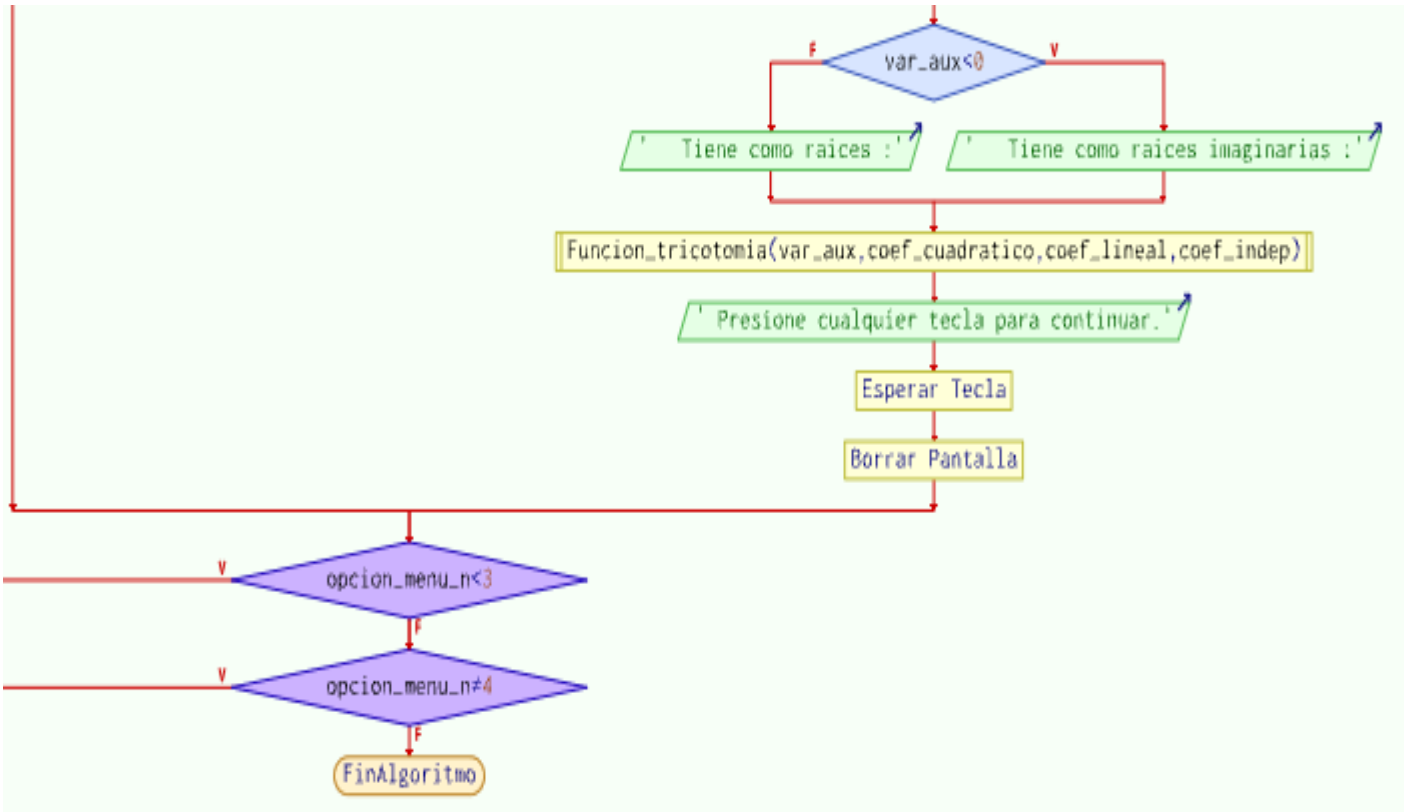
- La condicional if(){}else{}, para mostrar o no el procedimiento para luego considerar si se trata de una discriminante positiva, negativa o nula(esto con sus respectivos condicionales if{}).



- Básicamente mostrar como de efectúan las operaciones de la formula cuadrática con los valores ingresados.

Zoom (condicional if).

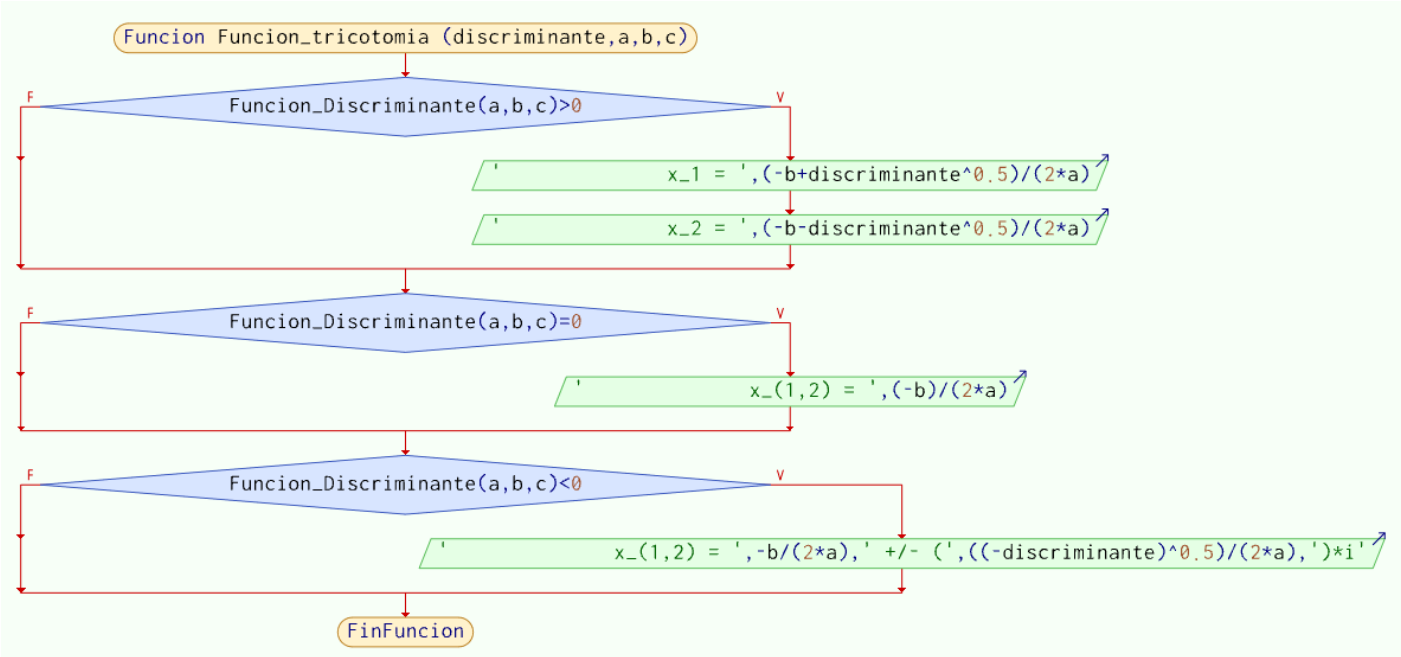
- Se ingresa a otra condicional if{} para imprimir en la consola, si las raíces son reales o son imaginarias conjugadas; para luego llamar a la Funcion_tricotomia.



- Antes de continuar será necesario notar que la función tricotomía depende de los 3 valores ingresados y del valor de la discriminante ya que para realizar las operaciones finales estos serán utilizados para los distintos casos de la tricotomía de la discriminante.

$$Discriminante = \Delta \quad \rightarrow \quad \begin{cases} \Delta > 0 \\ \Delta = 0 \\ \Delta < 0 \end{cases}$$

- Funcion_Discriminante, es llamada y por ello mostraremos si propio diagrama de flujo.



Para terminar los procesos del programa finalmente el usuario deberá elegir la opción de salir del menú.

Listado del código en el programa C++

C. Librerías

- **<iostream>**, para el ingreso y salida de datos.
- **<cmath>**, para realizar operaciones matemáticas de potencias o raíces, trigonométricas (seno, coseno, arco senos, etc.), etc.
- **<conio.h>**, para llamar a la función “getch(); “ que congela la consola hasta que el usuario digite una tecla.

D. Funciones

- **int funcion_discriminante(),** depende de las 3 variables ingresadas y calcula el valor de la determinante.

```
101 float funcion_discriminante(float a, float b, float c) {
102     float discriminante;
103     discriminante = 0;
104     discriminante = pow(b,2)-4*a*c;
105     return discriminante;
106 }
```

- **Void funcion_tricotomia(),** depende de las 3variables originales y también del valor de la determinante con el se aplican distintos condicionales para efectuar la terea matemática requerida.

```
108 void funcion_tricotomia(float discriminante, float a, float b, float c) {
109     if (funcion_discriminante(a,b,c)>0) {
110         cout << "            x_1 = " << (-b+pow(discriminante,0.5))/(2*a) << endl;
111         cout << "            x_2 = " << (-b-pow(discriminante,0.5))/(2*a) << endl;
112     }
113     if (funcion_discriminante(a,b,c)==0) {
114         cout << "            x_(1,2) = " << (-b)/(2*a) << endl;
115     }
116     if (funcion_discriminante(a,b,c)<0) {
117         cout << "            x_(1,2) = " << -b/(2*a) << " +/- (" << (pow((-discriminante),0.5))/(2*a) << ")*i" << endl;
118     }
119 }
```

- Ingresando al programa.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_2\L1_RMM_E2_v5.exe

      * Calculadora *

Sea la ecuacion cuadratica :

        ( a )*x^2 + ( b )*x + (c) = 0.

donde:     a != 0.

Digite el coeficiente cuadratico     -> 0
Valor invalido/Intentelo nuevamente.
Debe ingresar un valor distinto a 0.

Digite el coeficiente cuadratico     -> 1
Digite el coeficiente lineal         -> 3
Digite el coeficiente independiente  -> 4
```

El programa inicia mostrando la forma de una ecuación de 2do grado, para luego aclarar que el valor de 'a' debe ser distinto de 0 y finalmente pide al usuario ingresar un valor para a; Si el usuario ingresara el valor de '0' para el coef. Cuadrático, se mostrará el mensaje "valor invalido/Inténtelo nuevamente." Y aclarara que el valor ingresado para el coeficiente 'a' debe ser dittinto de '0'

Luego de que se ingrese algún valor valido para 'a', se precede a pedir los coeficientes restantes (b y c).

Presionando cualquier tecla para continuar:

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_2\L1_RMM_E2_v5.exe

        * Menu *

  Desea :
  1. Ver las raices resultantes.
  2. Ver el procedimiento.
  3. Ingresar nuevos valores
  4. Salir

  Digite su opcion ->
```

Se muestra un menú, es cual será recurrente hasta que el usuario elija la opción 3 o la 4.

Luego de elegir la 1ra opción:

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_2\L1_RMM_E2_v5.exe
  Sea la ecuacion de 2do grado :

        ( 1 )*x^2 + ( 3 )*x + ( 4 ) = 0.

  Tiene como raices imaginarias :

            x_(1,2) = -1.5 +/- (1.32288)*i


Presione cualquier tecla para continuar.
```

Para la 2da opción:

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_2\L1_RMM_E2_v5.exe
  Sea la ecuacion de 2do grado :

        ( 1 )*x^2 + ( 3 )*x + ( 4 ) = 0.

    x_(1,2) = (-3) + (+/-)[ (3)^2 - 4*(1)*(4) ]^(0.5) / (1*2)

  Donde :    Discriminante = (3)^2 - 4*(1)*(4)
             Discriminante = -7
  Entonces :
             x_(1,2) = (-3) (+/-) [ (7)^0.5 ]*i / (1*2)

  Tiene como raices imaginarias :

             x_(1,2) = -1.5 +/- (1.32288)*i


Presione cualquier tecla para continuar.
```

Ahora se probará al programa para que resuelva los ejemplos vistos anteriormente, con las distintas posibles soluciones y en la opción de ver desarollo.

i. Si $\Delta > 0$

Sea la ecuación:

$$1x^2 + 3x - 4 = 0$$

Resuelto por factorización tiene como raíces:

$$x_1 = 1 \;\; ; \;\; x_2 = -4$$

En el programa.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_2\L1_RMM_E2_v5.exe
  Sea la ecuacion de 2do grado :

        ( 1 )*x^2 + ( 3 )*x + ( -4 ) = 0.

    x_(1,2) = (-3) + (+/-)[ (3)^2 - 4*(1)*(-4) ]^(0.5) / (1*2)

  Donde :    Discriminante = (3)^2 - 4*(1)*(-4)
             Discriminante = 25
  Entonces :
             x_(1,2) = (-3) + (+/-)[25]^(0.5) / (1*2)

  Tiene como raices :

             x_1 = 1
             x_2 = -4


Presione cualquier tecla para continuar.
```

Otro ejemplo:

$$1x^2 + 10x + 1 = 0$$

Resuelto por la formula general cuadrática.

$$x_1 = -0{,}1010205144 \;\; ; \;\; x_2 = -9{,}898978486$$

En el programa.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_2\L1_RMM_E2_v5.exe
  Sea la ecuacion de 2do grado :

        ( 1 )*x^2 + ( 10 )*x + ( 1 ) = 0.

    x_(1,2) = (-10) + (+/-)[ (10)^2 - 4*(1)*(1) ]^(0.5) / (1*2)

  Donde :    Discriminante = (10)^2 - 4*(1)*(1)
             Discriminante = 96
  Entonces :
             x_(1,2) = (-10) + (+/-)[96]^(0.5) / (1*2)

  Tiene como raices :

             x_1 = -0.101021
             x_2 = -9.89898


Presione cualquier tecla para continuar.
```

ii. Si $\Delta = 0$

Sea la ecuación:

$$1x^2 + 1x + 0.25 = 0$$

Resuelto por la formula general cuadrática.

$$x_{1,2} = -0{,}5$$

En el programa.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_2\L1_RMM_E2_v5.exe
  Sea la ecuacion de 2do grado :

        ( 1 )*x^2 + ( 1 )*x + ( 0.25 ) = 0.

    x_(1,2) = (-1) + (+/-)[ (1)^2 - 4*(1)*(0.25) ]^(0.5) / (1*2)

  Donde :    Discriminante = (1)^2 - 4*(1)*(0.25)
             Discriminante = 0
  Entonces :
             x_(1,2) = (-1) / (1*2)

  Tiene como raices :

             x_(1,2) = -0.5


Presione cualquier tecla para continuar.
```

Otro ejemplo:

$$0.25x^2 + 1x + 1 = 0$$

Resuelto por la formula general cuadrática.

$$x_{1,2} = -2$$

En el programa.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_2\L1_RMM_E2_v5.exe
  Sea la ecuacion de 2do grado :

        ( 0.25 )*x^2 + ( 1 )*x + ( 1 ) = 0.

    x_(1,2) = (-1) + (+/-)[ (1)^2 - 4*(0.25)*(1) ]^(0.5) / (0.25*2)

  Donde :    Discriminante = (1)^2 - 4*(0.25)*(1)
             Discriminante = 0
  Entonces :
             x_(1,2) = (-1) / (0.25*2)

  Tiene como raices :

             x_(1,2) = -2


Presione cualquier tecla para continuar.
```

iii. Si $\Delta < 0$

Sea la ecuación:

$$1x^2 + 2x + 5 = 0$$

Resuelto por la formula general tiene como raíces imaginarias conjugadas:

$$x_{1,2} = -1 \pm 2i$$

En el programa.

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_2\L1_RMM_E2_v5.exe
  Sea la ecuacion de 2do grado :

        ( 1 )*x^2 + ( 2 )*x + ( 5 ) = 0.

    x_(1,2) = (-2) + (+/-)[ (2)^2 - 4*(1)*(5) ]^(0.5) / (1*2)

  Donde :    Discriminante = (2)^2 - 4*(1)*(5)
             Discriminante = -16
  Entonces :
             x_(1,2) = (-2) (+/-) [ (16)^0.5 ]*i / (1*2)

  Tiene como raices imaginarias :

             x_(1,2) = -1 +/- (2)*i


Presione cualquier tecla para continuar.
```

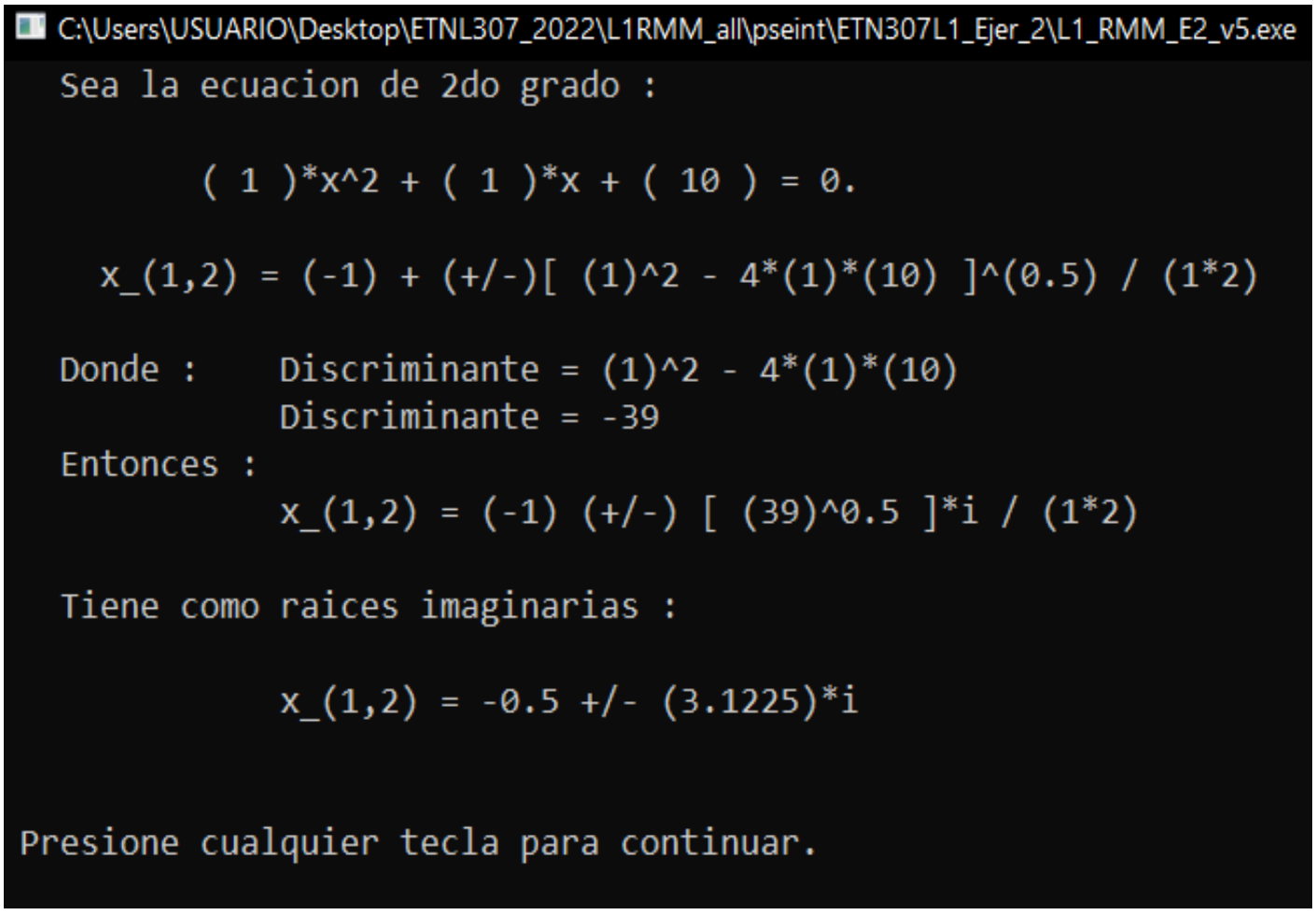
Otro ejemplo:

$$1x^2 + 1x + 10 = 0$$

Resuelto por la formula general tiene como raíces imaginarias conjugadas:

$$x_{1,2} = -0{,}5 \pm 3{,}122498999i$$

En el programa.



Conclusiones. -

El algoritmo realizo bien todos los casos que se le presento.

Observaciones. -

Los decimales que el programa maneja varía entre 5 a 6 decimales, se deberá tomar esto en cuenta para futuros proyectos si fueran necesario.

• ECUACION DE 3ER GRADO

Marco teórico. –

La ecuación general de tercer grado es:

$$ax^3 + bx^2 + cx + d = 0$$

Dónde: x es la variable.
a es el coeficiente cubico (distinto de cero)
b es el coeficiente cuadrático (distinto de cero).
c el coeficiente lineal.
d es el término independiente.
a, b, c y d pertenecen a los números reales.

Para la resolución de ecuaciones cubicas se empleará el método de Cardano-Tartaglia.

- Demostración:

  Dividiendo a de la ecuación general

$$x^3 + \frac{b}{a}x^2 + \frac{c}{a}x + \frac{d}{a} = 0$$

  Para mayor facilidad de operaciones se partirá de la ecuación

$$x^3 + ax^2 + bx + c = 0 \qquad (1)$$

  Realizando el cambio de variable

$$x = t - \frac{a}{3} \qquad (2)$$

Se elimina el termino cuadrático y se obtiene la forma reducida

$$t^3 + pt + q = 0, \qquad (3)$$

Donde:

$$p = \frac{3b - a^2}{3}, \qquad q = \frac{2a^3 - 9ab + 27c}{27}$$

Realizando otro cambio de variable

$$t = u + v, \qquad (4)$$

Elevando este último al cubo se obtiene

$$(u+v)^3 = u^3 + v^3 + 3u^2v + 3uv^2 = u^3 + v^3 + 3uv(u+v), \qquad (5)$$

Que nos da la identidad

$$(u+v)^3 - 3uv(u+v) - u^3 - v^3 = 0. \qquad (5)$$

Por lo tanto, si encontramos los calores de u, v tales que

$$p = -3uv, \qquad q = -u^3 - v^3, \qquad (6)$$

Despejando v de la primera ecuación

$$v = -p/3u.$$

Reemplazando está en la ec. (4) obtenemos

$$t = u - \frac{p}{3u} \qquad (7)$$

También podemos reemplazar v en la 2da ecuación y obtenemos

$$u^3 + q - (p/3u)^3 = 0$$

Multiplicando por u^3

$$u^6 + qu^3 - (p/3)^3 = 0. \qquad (8)$$

- Nótese que esta ecuación tiene una forma similar a una ecuación cuadrática, entonces podemos verla de la siguiente manera

$$u^2 - tu - p/3 = 0.$$

(esto solo para tomar prestado el análisis de la discriminante)

Reemplazando la ec. (6) en la ec. (5) obtenemos:

$$(u+v)^3 - 3uv(u+v) + q = 0 \qquad (9)$$

Así pues, concluimos que resolver la ecuación (3) es equivalente a resolver la ecuación (8), en el sentido de que las soluciones de la ecuación (3) son las que se obtienen a partir de las soluciones de la ecuación (8) a través de la ecuación (7). Ahora bien, en la ecuación (8) u es cuadrática con en u^3, luego sus soluciones son:

$$u^3 = \frac{-q \pm \sqrt{q^2 + 4(p/3)^3}}{2} = -\frac{q}{2} \pm \sqrt{(q/2)^2 + (p/3)^3}$$

O visto de una manera más sencilla:

$$u^3 = -q/2 \pm \sqrt{\Delta}. \qquad (10)$$

Donde la discriminante la podemos escribir de la siguiente manera

$$\Delta = \left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3$$

Combinando las 2 raíces cuadradas de la discriminante con las 3 raíces cubicas, nos salen las 6 raíces de la ecuación (8), pero la ecuación (3) solo puede tener 3 raíces. Ello se debe a que, cada valor de t, hay 2 valores de u que cumplen con la ecuación (7). Dado uno de ellos, el otro está determinado por la relación

$$uu' = -p/3, \qquad (11)$$

Que se deduce de que u u' son raíces de la ecuación

$$u^2 - tu - p/3 = 0.$$

Fijando un u positivo que cumpla con la ecuación (10), con signo positivo y una raíz cuadrada prefijada, y sea u negativo una raíz de la ecuación (10) con signo negativo (y de la misma raíz cuadrada). Entonces

$$(u_+u_-)^3 = (q/2)^2 - (q/2)^2 - (p/3)^3 = -(p/3)^3,$$

Luego

$$u_+u_- = -(p/3)\omega,$$

Donde omega es la solución de la raíz cubica de la unidad. Por consiguiente

$$u_+(\omega^2 u_-) = -(p/3).$$

Cambiando u negativo por (omega)^2*u_, concluimos que, para cada u positivo elegido arbitrariamente, existe un u negativo elegido adecuadamente tal que u positivo y u negativo determinan la misma raíz t de la ecuación (3). En definitiva, vemos que las soluciones de (3) son de la forma de la ecuación (7), con

$$u = \sqrt[3]{-q/2 + \sqrt{\Delta}},$$

donde en esta expresión hay que entender que la raíz cuadrada del determinante es fija (elegida arbitrariamente de entre las dos posibilidades) y que al variar la elección de la raíz cubica recorremos las distintas raíces de la ecuación (3). Finalmente observamos que, de la segunda ecuación de (6), se sigue que

$$v^3 = -q/2 - \sqrt{\Delta},$$

Despejando el cubo obtenemos

$$v = \sqrt[3]{-q/2 - \sqrt{\Delta}},$$

- Análisis de los casos posibles
  - i. **Caso de una raíz triple** $(p = q = 0)$

    En este caso en particular la forma de la ecuación cubica será de la forma:

    $$(x - \alpha)^3 = x^3 - 3\alpha x^2 + 3\alpha^2 x - \alpha^3 = 0,$$

    Notando resulta una raíz triple de alpha.

    Donde en la fórmula de Cardano se reduce a

    $$x = -a/3$$

- Cubicas en el caso clásico

  Esta depende de la discriminante y sus casos son:

  i. **Si** $\Delta = 0$

  Todas sus raíces son reales, y almenos dos de ellas son iguales.

  P y q son distintos de cero, entonces la ecuación tiene como raíces

$$x = -\frac{3q}{2p} - \frac{a}{3}, \quad y \quad x = -\frac{4p^2}{9q} - \frac{a}{3}.$$

  ii. **Si** $\Delta > 0$

  Se tiene una raíz real y dos imaginarias.

  Una raíz real viene dada por

$$x = \sqrt[3]{-q/2 + \sqrt{\Delta}} + \sqrt[3]{-q/2 - \sqrt{\Delta}} - a/3,$$

  donde las raíces cubicas u y v son reales. Las otras dos raíces son imaginarias, y vienen dadas por

$$x = -\frac{u+v}{2} - \frac{a}{3} \pm \frac{\sqrt{3}}{2}(u-v)\,i.$$

  iii. **Si** $\Delta < 0$

  La ecuación tiene tres raíces reales y simples.

  Estas raíces vienen dadas por

$$x = 2\sqrt{-\frac{p}{3}}\cos\frac{\theta + 2k\pi}{3} - \frac{a}{3},$$

  Donde k=0,1,2 y el ángulo 0 < theta < pi está determinado por

$$\cos\theta = \frac{-q/2}{\sqrt{-(p/3)^3}}.$$

❖ Ejemplos prácticos. –

i. Caso de una raíz triple $(p = q = 0)$

Sea la ecuación:

$$1x^3 + 3x^2 + 3x + 1 = 0$$

Resuelto por factorización tiene como raíces

$$x_{1,2,3} = -1$$

i. Si $\Delta > 0$

Sea la ecuación:

$$1x^3 + 2x^2 + 3x + 4 = 0$$

- Valores calculados:

$$p = \frac{3 * 2 - 1^2}{3} \rightarrow \quad p = 1.6666667$$

$$q = \frac{2 * 2^3 - 9 * 2 * 3 + 27 * 4}{27} \rightarrow \quad q = 2{,}5925925926$$

$$\Delta = 7{,}4074074074$$

Resuelto por factorización tiene como raíces:

$$x_1 = -1.6506291914$$

$$x\_2 = -0.1746854043 \pm 1{,}5468688872i$$

Otro ejemplo ecuación:

$$1x^3 - 3x^2 + 9x - 5 = 0$$

Valores calculados:

$$p = \frac{3 * 9 - 3^2}{3} \quad \rightarrow \quad p = 6$$

$$q = \frac{2 * 9^3 - 9 * (-3) * 9 + 27 * (-5)}{27} \rightarrow \quad q = 2$$

$$\Delta = 36$$

Resuelto por factorización tiene como raíces:

$$x_1 = 0{,}6725199979$$

$$x\_2 = 1{,}163740001 \pm 2{,}465853273i$$

i. Si $\Delta = 0$

Sea la ecuación:

$$27x^3 + 27x^2 - 72x + 28 = 0$$

- Valores calculados:

$$p = -3$$

$$q = 2$$

$$\Delta = 0$$

Resuelto por factorización tiene como raíces:

$$x_1 = -2.3333333333$$

$$x_{2,3} = 0.666666666$$

ii. Si $\Delta < 0$

Sea la ecuación:

$$x^3 + 5x^2 - 8x - 42 = 0$$

- Valores calculados:

$$p = -16{,}3\overline{33}$$

$$q = -19, \overline{407}$$

$$\Delta = -268, , \overline{888}$$

Resuelto por factorización tiene como raíces:

$$x_1 = 2{,}8729833462$$

$$x_2 = -4{,}8729833462$$

$$x_3 = -3$$

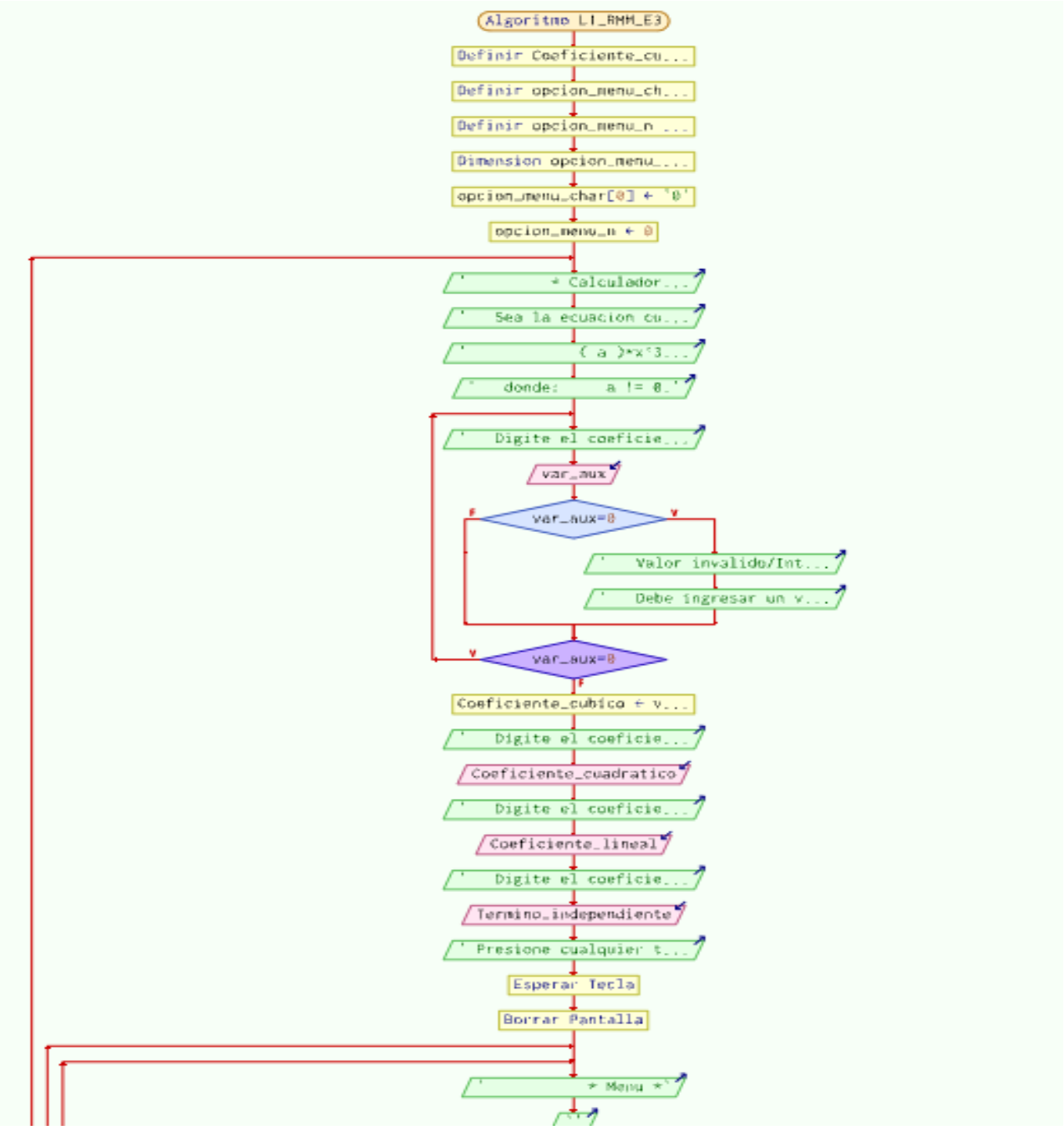
Diagrama de flujo. –

Para dibujar los diagramas de flujo de los algoritmos se empleó el programa “PSeInt”, y como estos están conformados por funciones se muestran de la siguiente manera; Además para poder visualizar de manera detallada se realizarán distintos acercamientos a cada diagrama de flujo.

- Series de Taylor y Maclaurin (L1_RMM_EJ_3 es el nombre del archivo)

- Diagrama principal (parte superior)

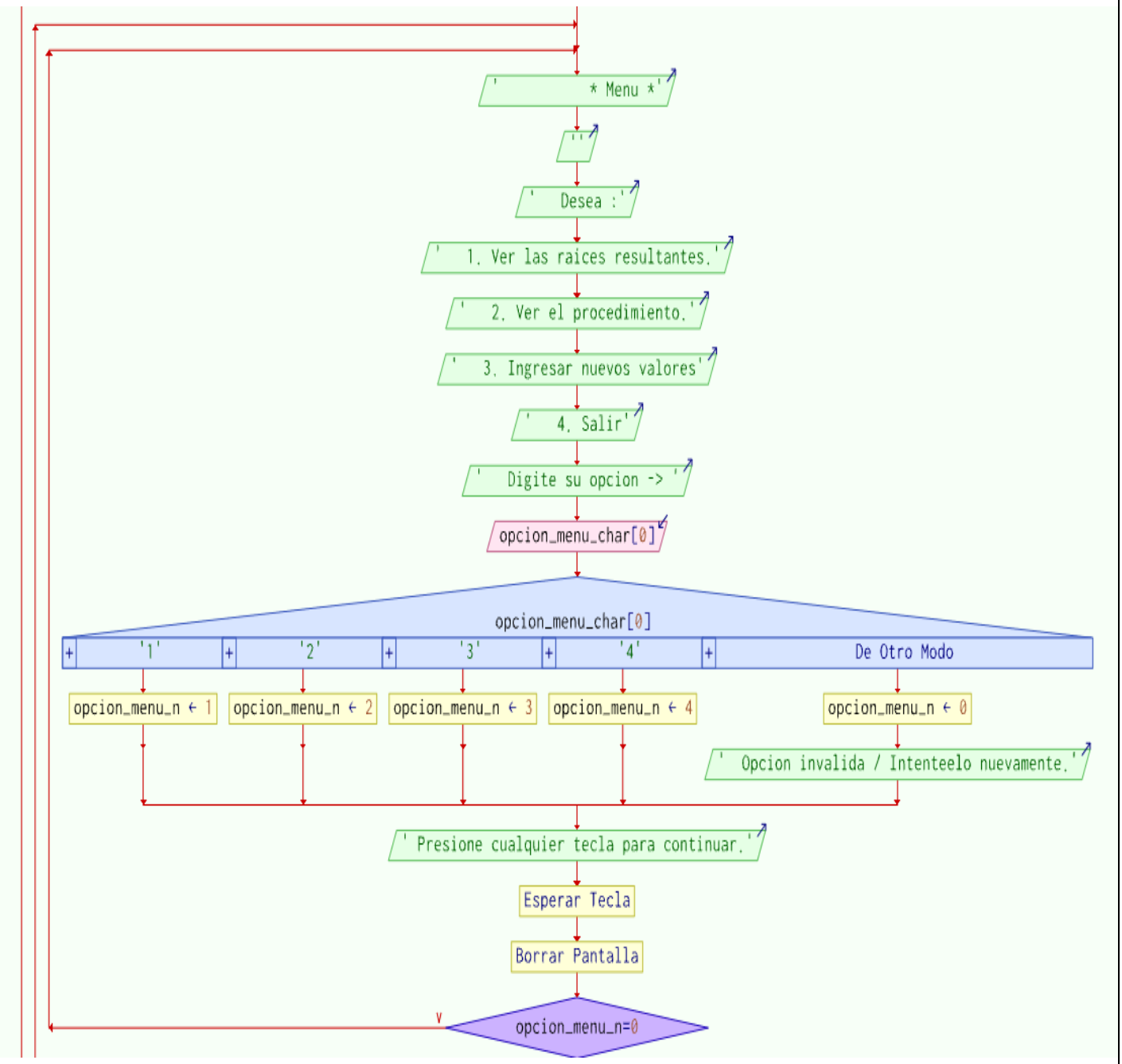
Zoom a la parte superior del diagrama de flujo



Similar al algoritmo anterior, se declaran variables que se usuran en determinados puntos del programa.

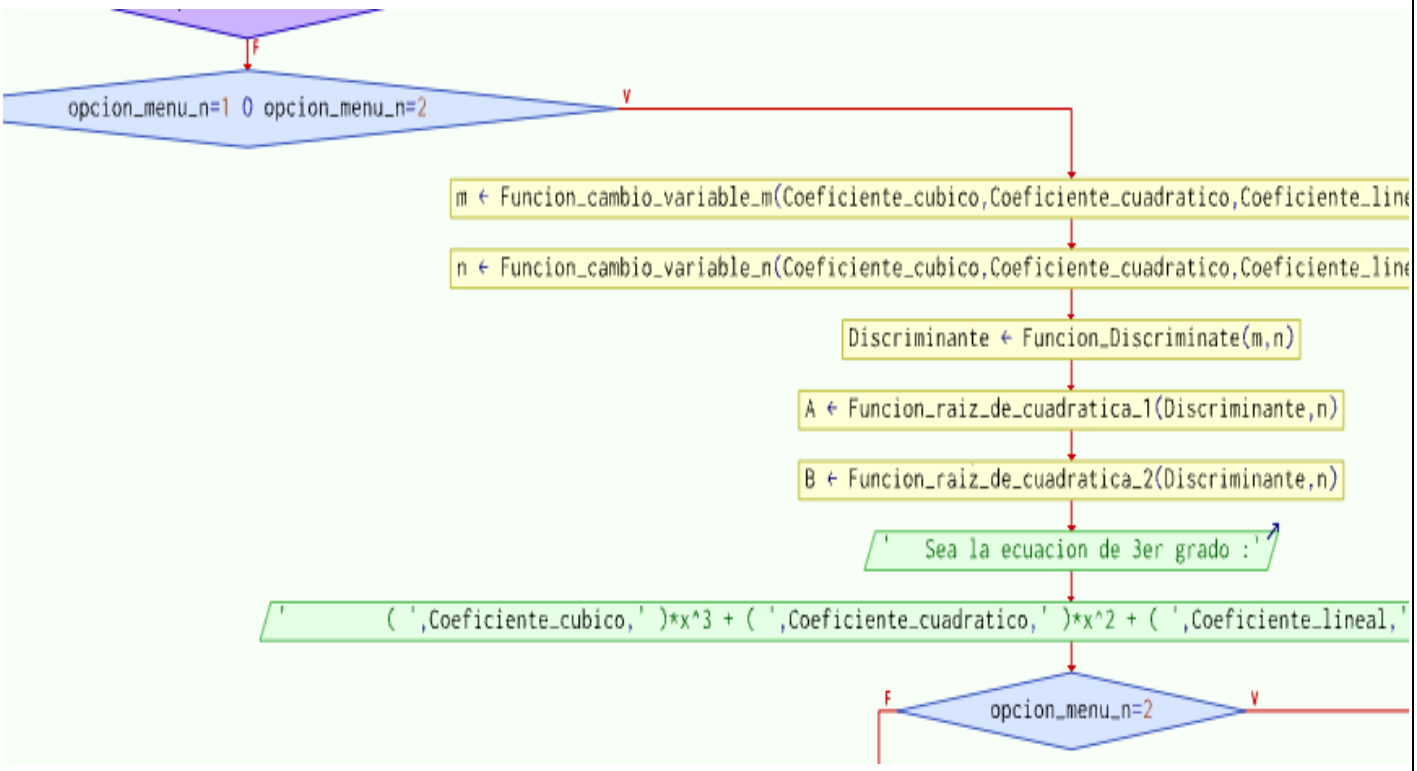
También consta de un ciclo do{}while() que impedirá que el programa llegara a cerrarse, sí que el usuario no lo deseara; Otro ciclo do{}while() que impide que el valor para el coeficiente cubico sea cero.
Se ingresan los valores de los demás coeficientes, se borra la pantalla y finalmente ingresamos al menú.

Zoom a la parte del menú



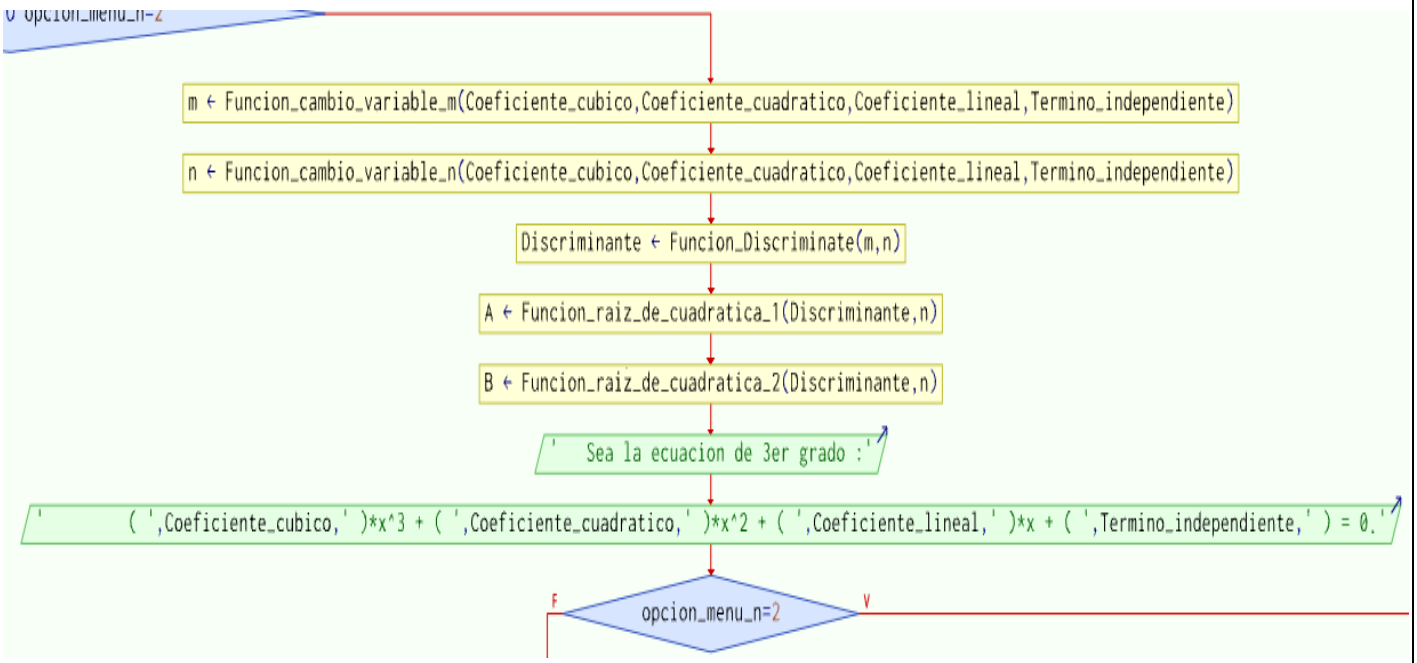
En esta parte del se elija que el usuario ingrese un valor erróneo para la selección de opciones, guardando dicho valor para luego pasar por un chico do{}while() para continuar o repetir hasta que el valor ingresado sea correcto.

Zoom a la parte del menú



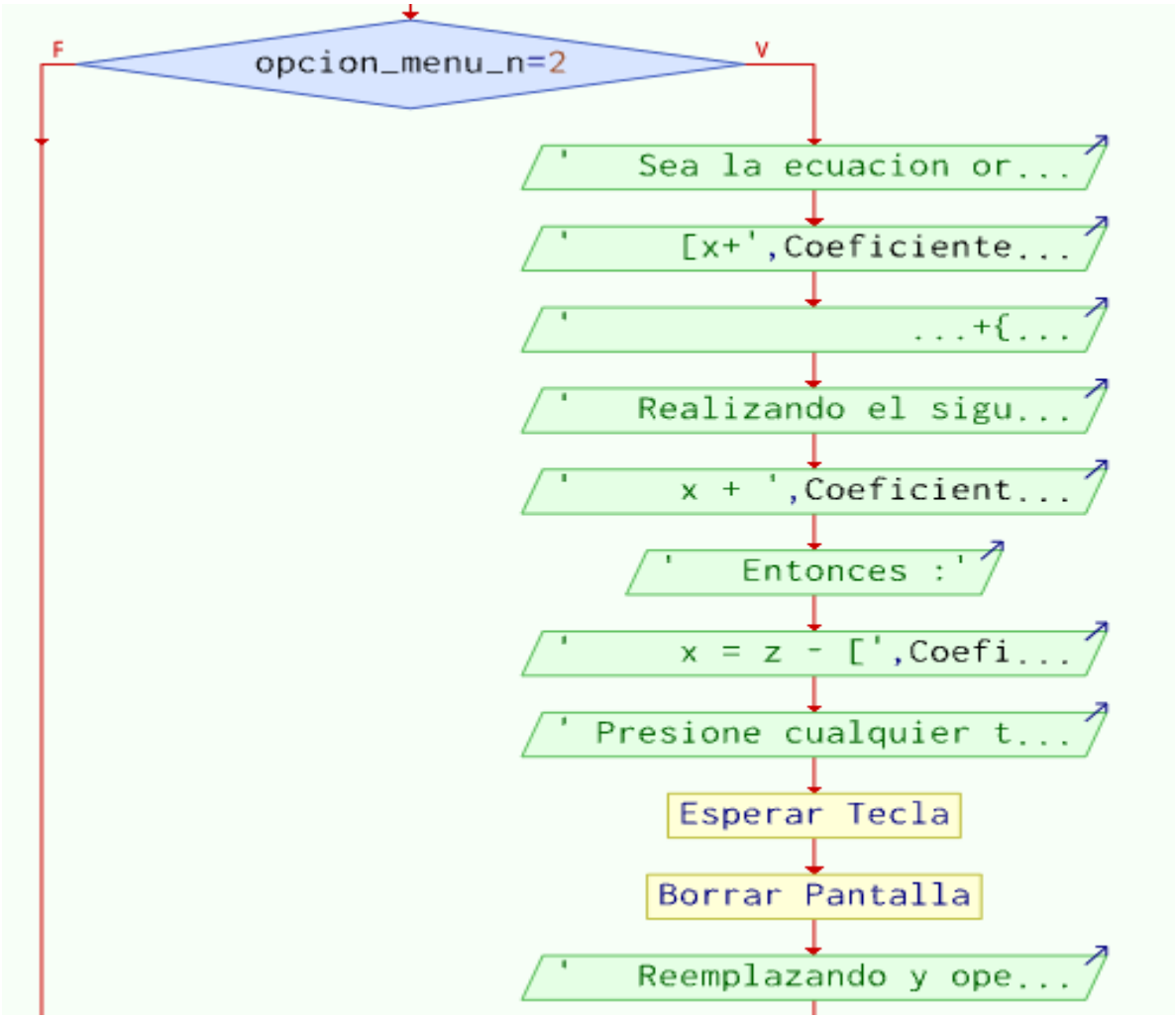
El valor ingresado finalmente sele del ciclo, para ser evaluado en una condiciones if(){}else{} para iniciar con los cálculos para resolver la ec. De 3er grado, o saltarlos y reiniciar el programa o simplemente cerrar dicho programa.

Si fuera el caso de iniciar con los cálculos:

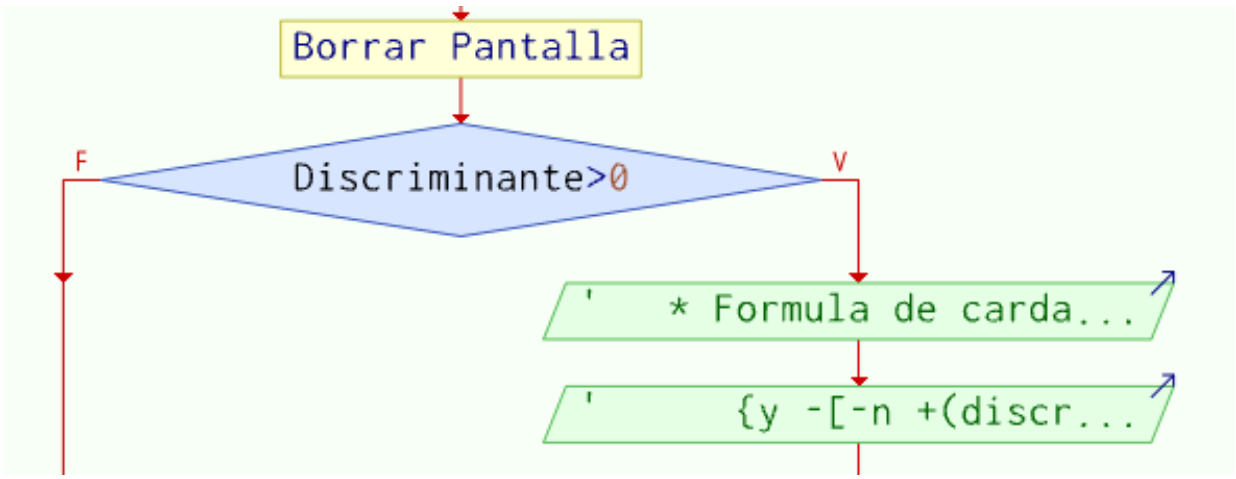


Realiza los cálculos que ya se vieron en la parte de marco teórico para este caso, se ingresa en una segunda condicional para mostrar en pantalla los procedimientos o simplemente mostrar los resultados; luego de terminar con el diagrama principal se mostrar con más detalle las funciones y operaciones se realizaron en esta parte.

Zoom a la parte de mostrar procedimiento

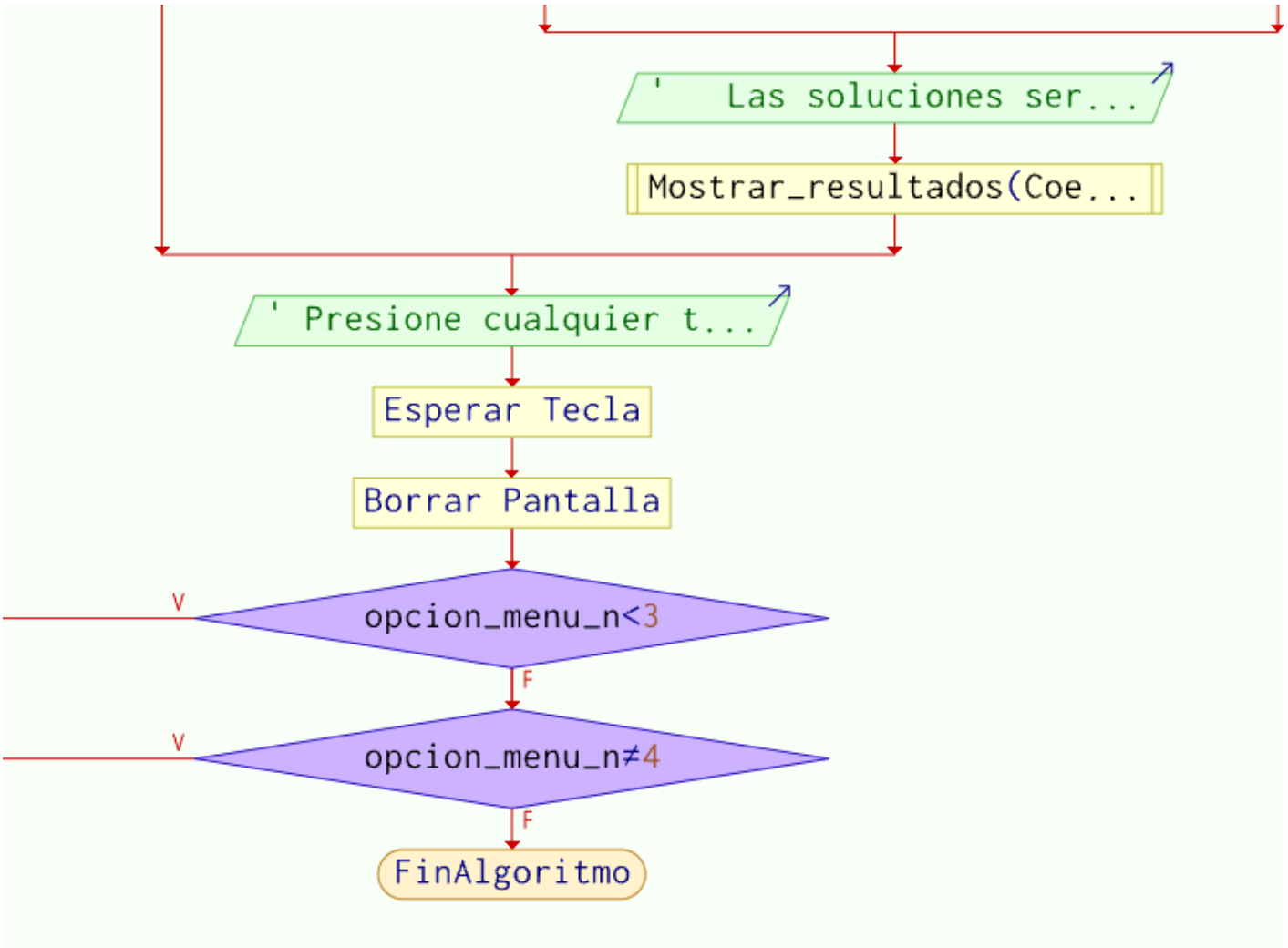


Recortando la imagen



El algoritmo ingresa en distintas condicionales if{}else{}, esto simplemente para mostrar el caso correspondiente de la discriminante, para finalmente mostrar los resultados de las raíces.

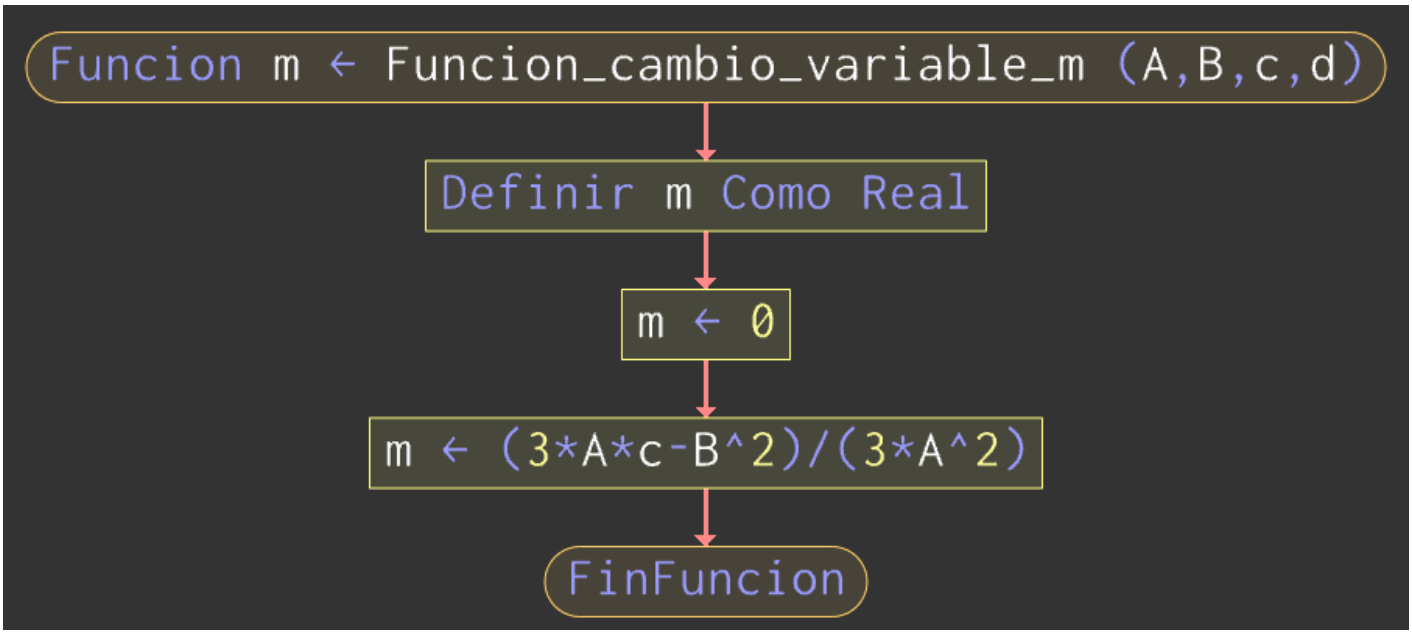
Zoom a la parte de mostrar resultados



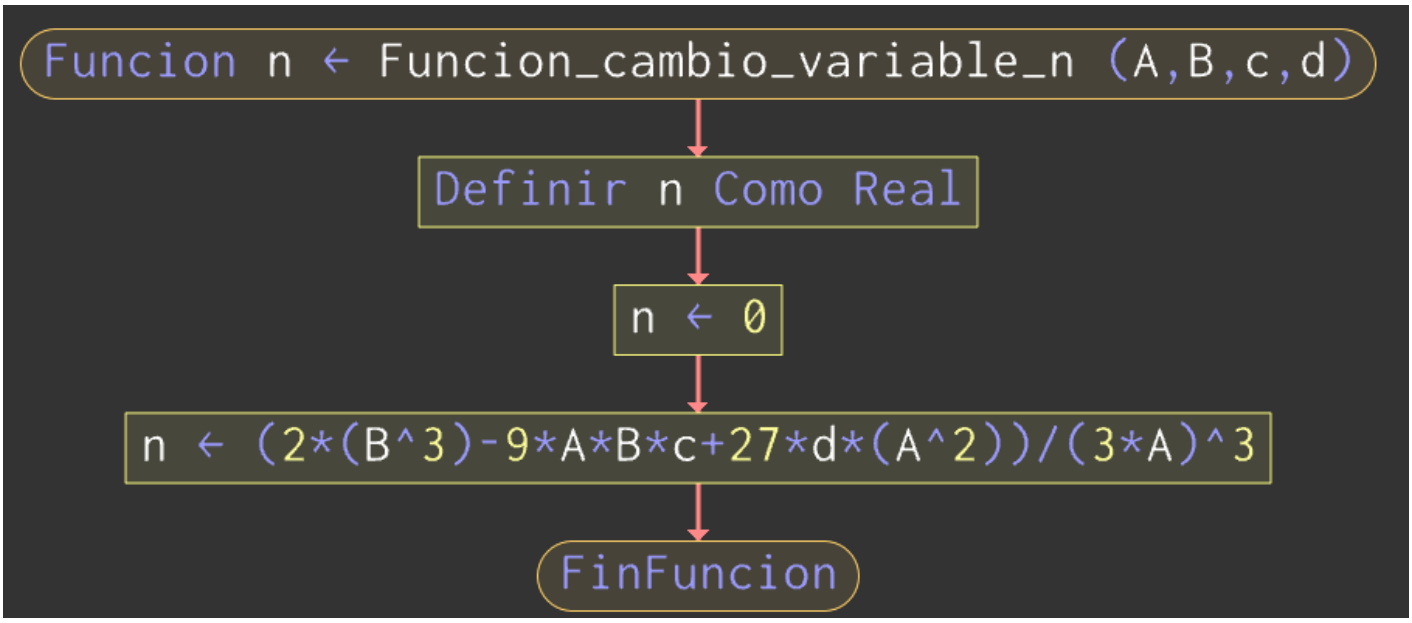
Finalmente, el algoritmo ingresa a 2 condicionales while, estos impedirán que el programa se cierre si el usuario no lo deseara.

El primer While retornara a la sección de menú ver resultados, mientras que el 2do retorna la a sección de ingresar datos.

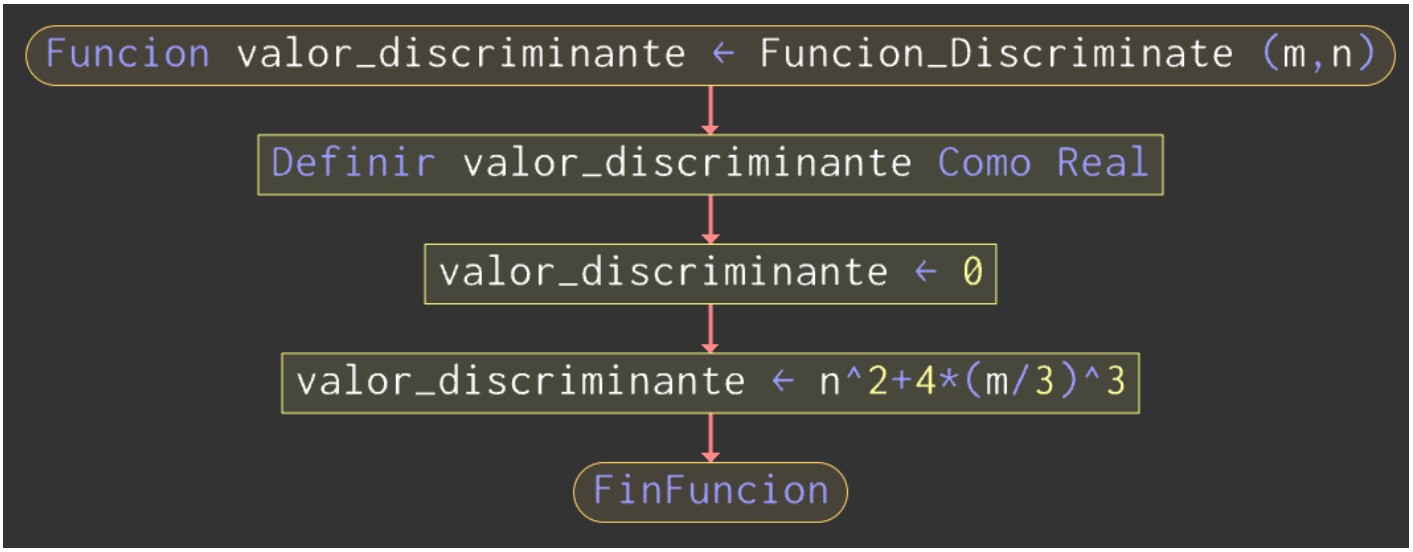
- **Funcion_cambio_variable_m(,)**



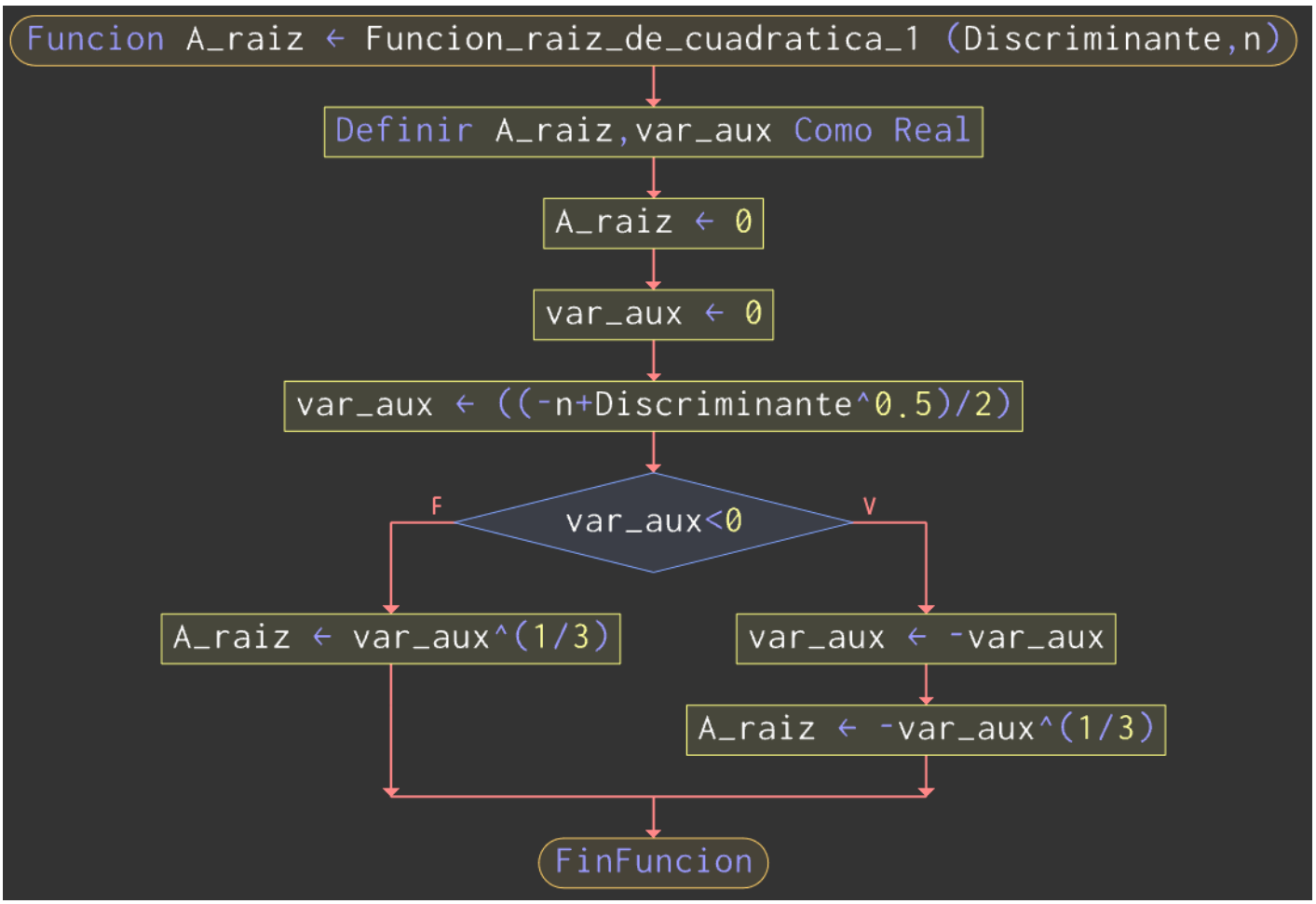
- **Funcion_cambio_variable_n(,)**



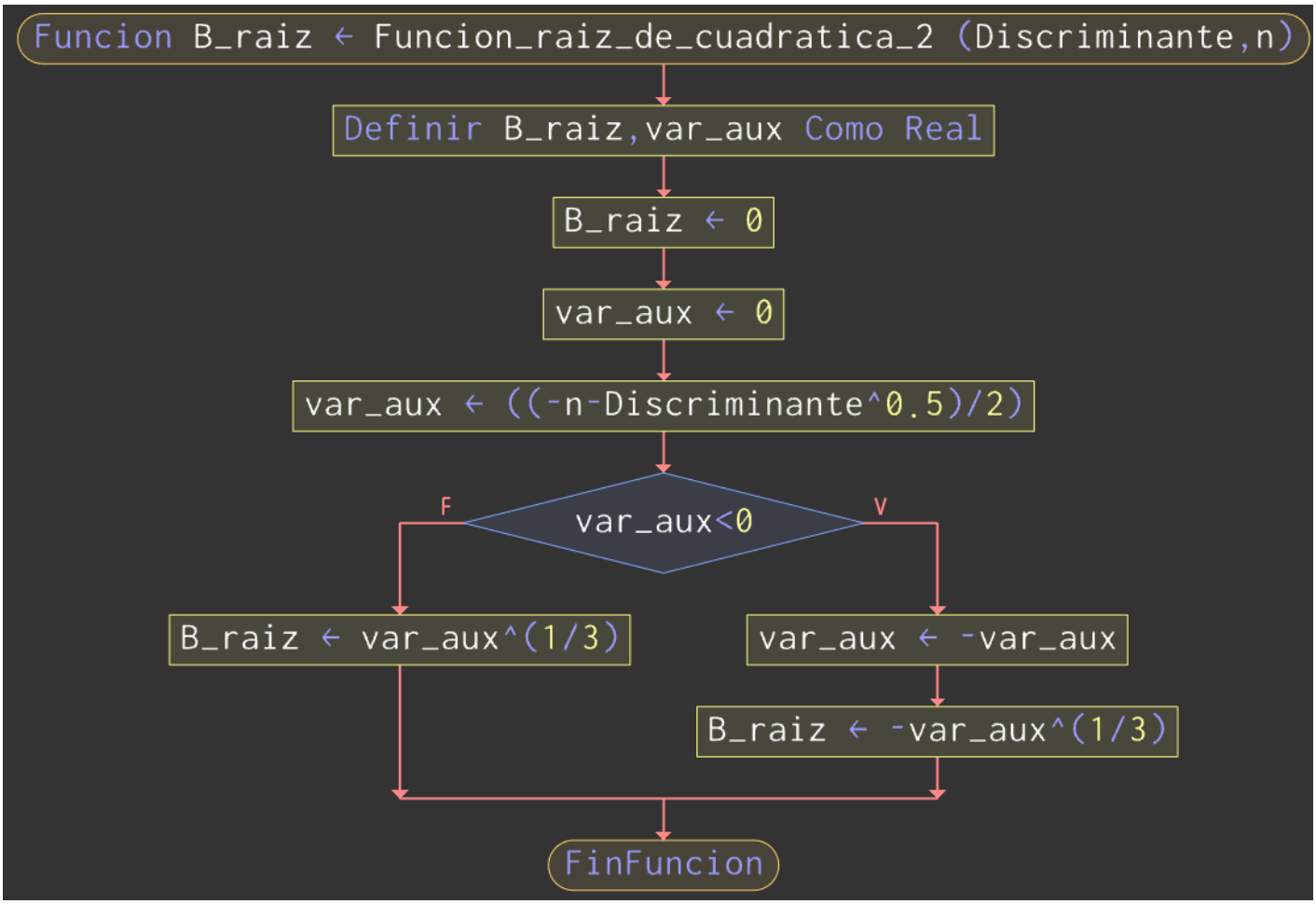
- **Funcion_valor_discriminante(,)**



- **Funcion_raiz_1(,)**



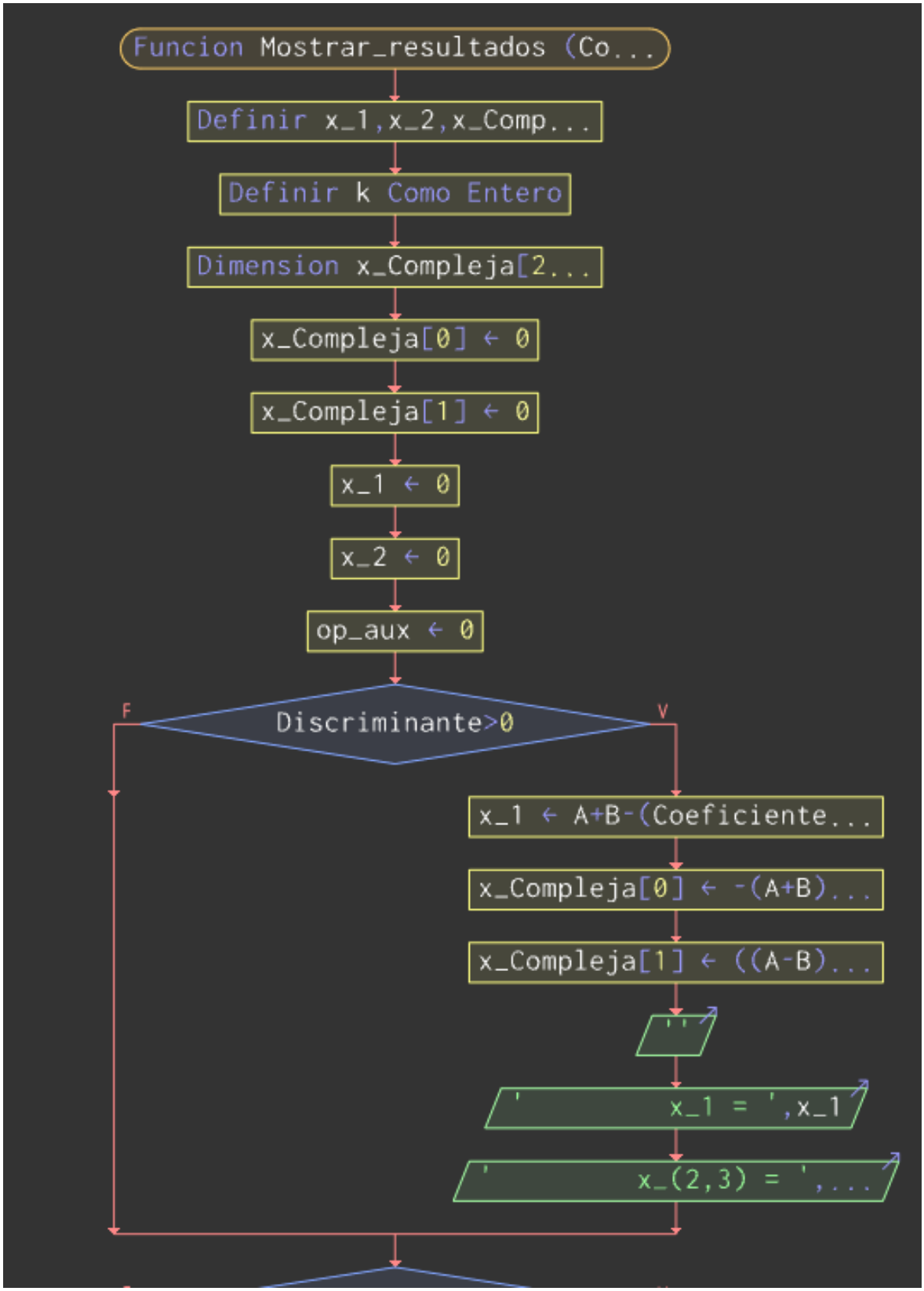
- **Funcion_raiz_2(,)**



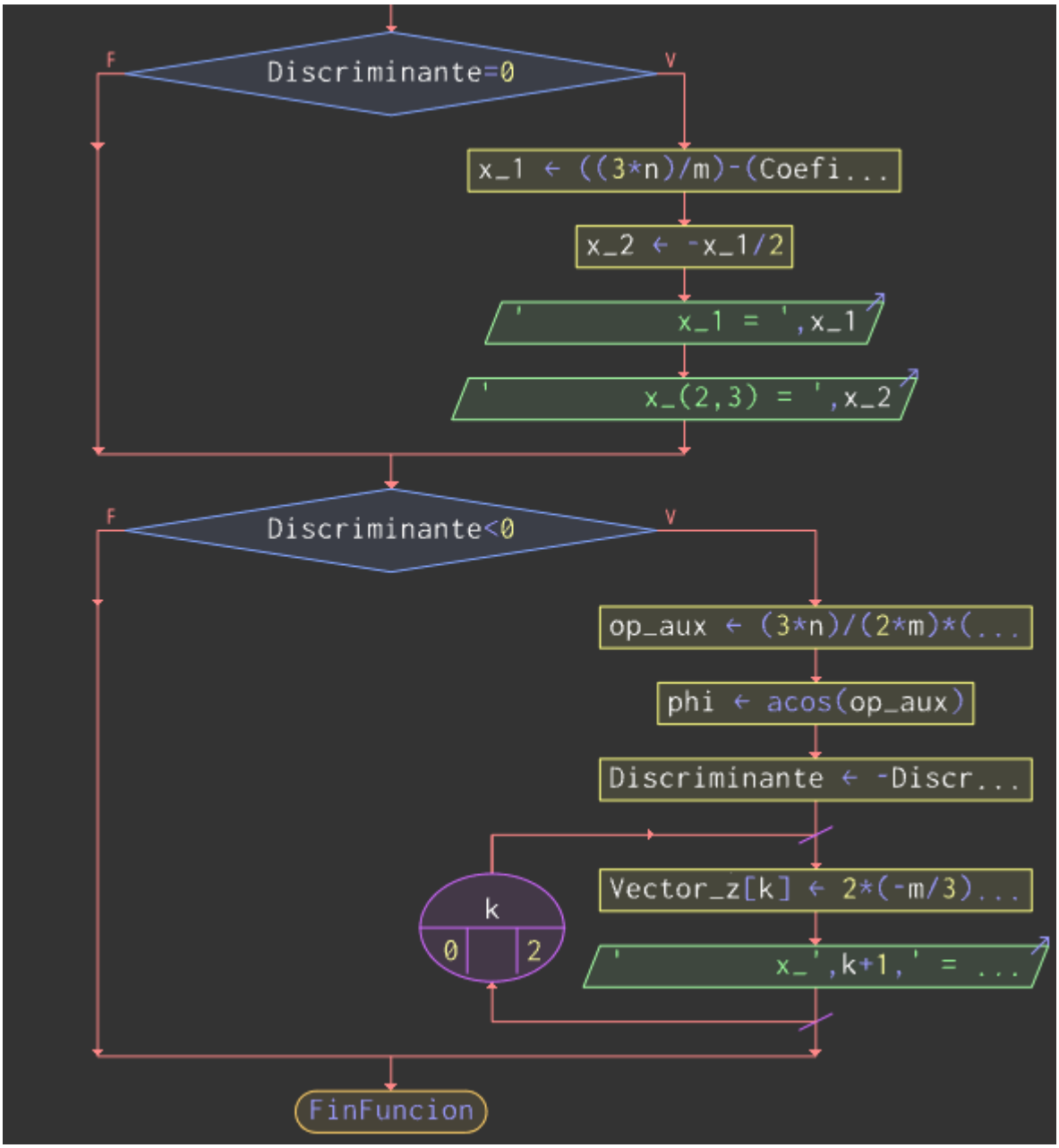
- Las funciones que se acaban de ver realizan los cambios de variable y demás operaciones que son necesarios para hallar las raíces de la ec de 3er grado.

- **Funcion_mostar_resultados(,)**

Zoom a la parte superior del diagrama de flujo.

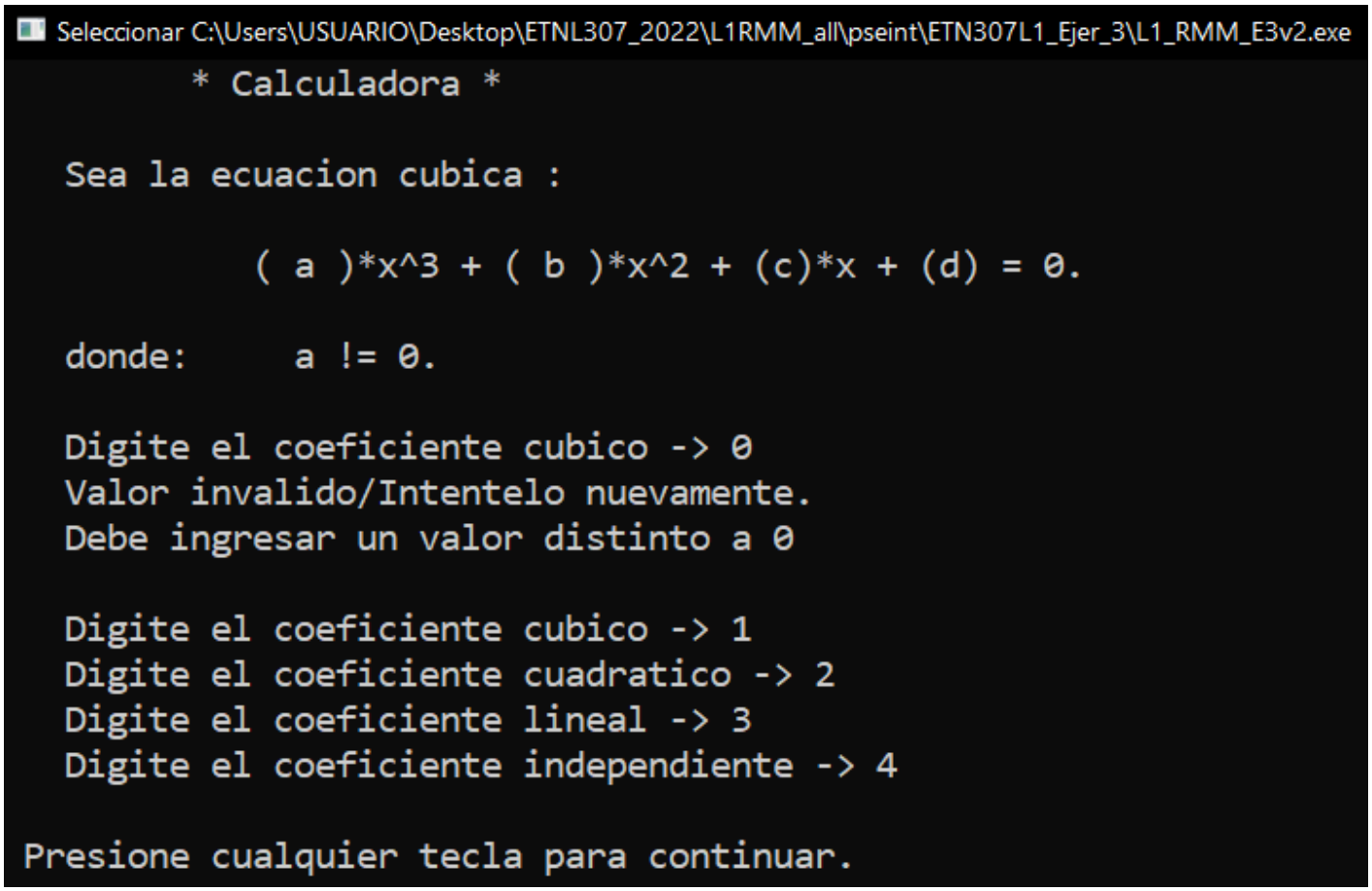


Zoom a la parte inferior del diagrama de flujo.



Como su nombre lo indica, mostrara lis resultados finales de las raíces.

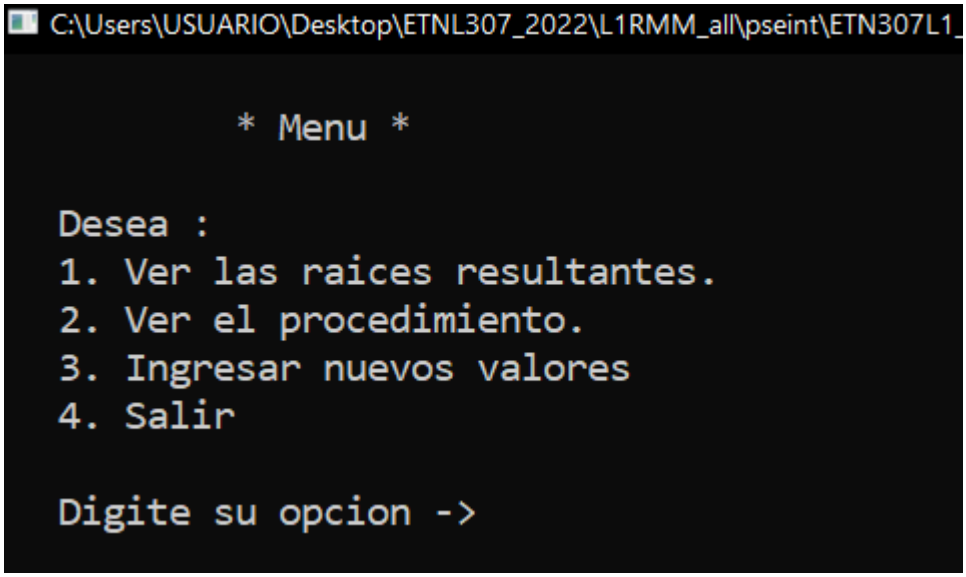
- Ingresando al programa.



El programa inicia mostrando la forma de una ecuación de 3er grado, para luego aclarar que el valor de 'a' debe ser distinto de 0 y finalmente pide al usuario ingresar un valor para a; Si el usuario ingresara el valor de '0' para el coef. Cubico, se mostrará el mensaje "valor invalido/Inténtelo nuevamente." Y aclarara que el valor ingresado para el coeficiente 'a' debe ser distinto de '0'

Luego de que se ingrese algún valor valido para 'a', se procede a pedir los coeficientes restantes.

Presionando cualquier tecla para continuar



Se muestra un menú, es cual será recurrente hasta que el usuario elija la opción 3 o la 4.

Eligiendo la 1 opción de ver procedimiento

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe
  Sea la ecuacion de 3er grado :

        ( 1 )*x^3 + ( 2 )*x^2 + ( 3 )*x + ( 4 ) = 0.

  Sea la ecuacion ordenada del modo :
    [x+2/(1*3)]^3 + {[3*(1)*(3) -(2)^2] / [3*(1)^2]}*x +...
                ...+{[27*(1)^2*(4) -2^3]/(1*3)^3} = 0

  Realizando el siguiente cambio de variable :
    x + 2/(1*3) = z

  Entonces :
    x = z - [2/(1*3)]

Presione cualquier tecla para continuar.
```

Presionando cualquier tecla para continuar

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe
  Reemplazando y operando saldra :
    z^3 + {[3*(1)*(3) -(2)^2] / [3*(1)^2]}*z + {[ (2)^3*3 -9*(1)(2)(3) +27*(4)*(1)^2 ]/(1*3)^3} = 0

  Realizando el 2do cambio de variable :
    m = [3*(1)*(3) -(2)^2] / [3*(1)^2]
    * m = 1.66667

    n = [ (2)^3*3 -9*(1)(2)(3) +27*(4)*(1)^2 ]/(1*3)^3
    * n = 2.59259

  Realizdo la operacion auxiliar:
     z = r_1 + r_2   // ()^3
    z^3 = (r_1)^3 +(r_2)^3 +3*[r_1*r_2]*[ r_1+r_2 ]
    z^3 -[(r_1)^3 +(r_2)^3] -[3*r_1*r_2]*z = 0

Presione cualquier tecla para continuar.
```

Presionando cualquier tecla para continuar

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe

    z^3 -[(r_1)^3 +(r_2)^3] -[3*r_1*r_2]*z = 0

  Obteniendo el siguiente sistema de ecuaciones :
    ( ) ... -(3*r_1*r_2) = m   // ()^3
    (1) ... (r_1*r_2)^3 = -(m/3)^3

    ( ) ... -(r_1)^3-(r_2)^3 = n
    (2) ... +(r_1)^3+(r_2)^3 = -n

Presione cualquier tecla para continuar.
```

Presionando cualquier tecla para continuar

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe
  Sea una ecuacion de 2do grado :
            y^2 +(-1)*(P_1+P_2)*y +P_1*P_2 = 0
  Cuyas raices cumplen :
            P_1 +P_2 = -(Q_1/Q_2)     Y     P_1*P_2 = -(Q_0/Q_2)

  La ec. de 2do grado quedaria :
            y^2 +(Q_1/Q_2)*y +(Q_0/Q_2) = 0
  Para nuestro caso :
                (r_1)^3 = P_1     Y     (r_2)^3 = P_2

  Reemplazando en el sistema de ec. anterior :
    (1) ... ( Q_0/Q_2 ) = -(m/3)^3
    (2) ... -( Q_1/Q_2 ) = -n
  En la ec. de 2do grado :
            y^2 +y*n -(m/3)^3 = 0

Presione cualquier tecla para continuar.
```

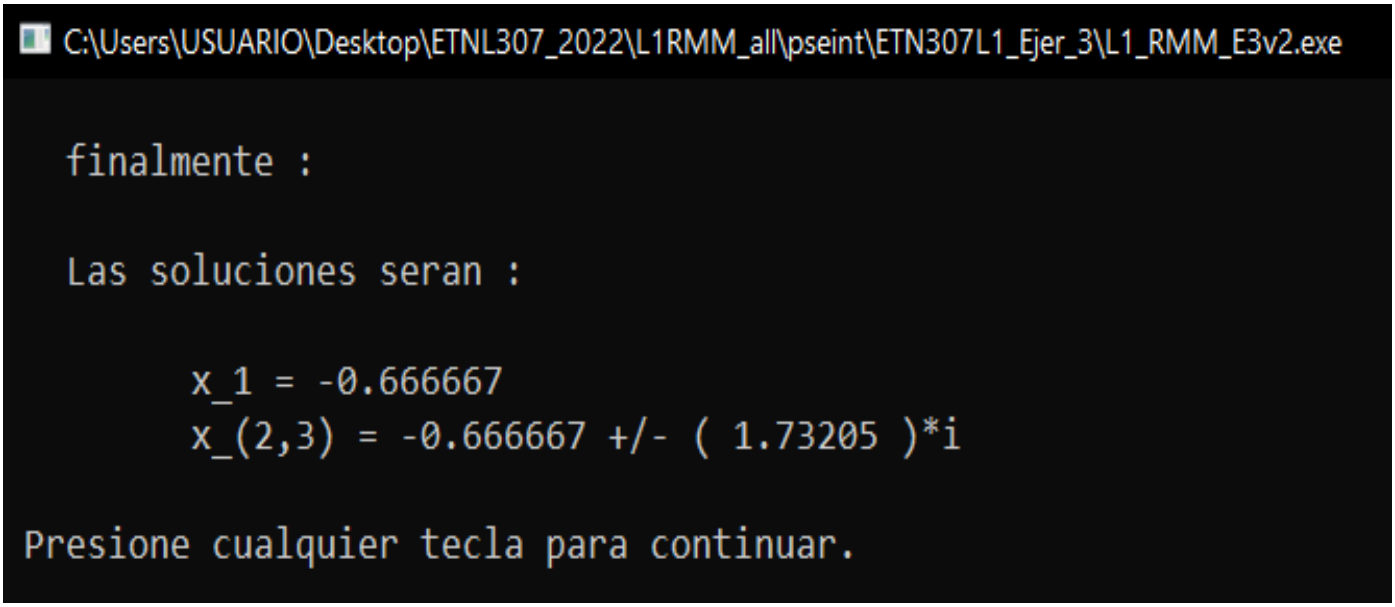
Presionando cualquier tecla para continuar

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe

            y^2 +y*n -(m/3)^3 = 0

Hallando su discriminante :
         discriminante = n^2 -(-m/3)^3
         discriminante = 2.59259^2 +(1.66667/3)^3
        * discriminante = 7.40741

Presione cualquier tecla para continuar.
```

Presionando cualquier tecla para continuar

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe
 * Formula de cardano :
    {y -[-n +(discriminante)^0.5] / 2} * {y -[-n -(discriminante)^0.5] / 2} = 0
 Donde :     (r_1)^3 = -n +(discriminante)^0.5
             (r_2)^3 = -n -(discriminante)^0.5
Solucion real :
   z_1 = r_1 + r_2
Retornando al cambio de variable :
   x_1 = r_1 +r_2 -(b/3*b)
   x_1 = (-n+(discriminante)^0.5)^1/3 +(-n-(discriminante)^0.5)^1/3
   x_1 = (-2.59259+(7.40741)^0.5)^1/3 +(-2.59259-(7.40741)^0.5)^1/3

y para los terminos imaginarios :
   z_(2,3) = [ -0.5*( r_1 +r_2 ) ] +(+/-){ (3)^0.5/(2)[ r_1 -r_2 ] }*i
   z_(2,3) = [ -0.5*( (-2.59259+(7.40741)^0.5)^1/3 +(-2.59259-(7.40741)^0.5)^1/3 ) ] +...
           (+/-){(3)^0.5/(2)[ (-2.59259+(7.40741)^0.5)^1/3 -(-2.59259-(7.40741)^0.5)^1/3]}*i

Presione cualquier tecla para continuar.
```

Presionando cualquier tecla para continuar



Para las siguientes pruebas solo se tomarán capturas de pantalla a las partes importantes del procedimiento.

❖ Ejemplos prácticos. –

i. Caso de una raíz triple $(p = q = 0)$

Sea la ecuación:

$$1x^3 + 3x^2 + 3x + 1 = 0$$

Resuelto por factorización tiene como raíces

$$x_{1,2,3} = -1$$

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe
  Sea la ecuacion de 3er grado :

        ( 1 )*x^3 + ( 3 )*x^2 + ( 3 )*x + ( 1 ) = 0.

  Las soluciones seran :

        x_(1,2,3) = -1

Presione cualquier tecla para continuar.
```

Para la opción ver procedimiento:

Donde p = m y q = n.

```
Realizando el 2do cambio de variable :
  m = [3*(1)*(3) -(3)^2] / [3*(1)^2]
  * m = 0

  n = [ (3)^3*3 -9*(1)(3)(3) +27*(1)*(1)^2 ]/(1*3)^3
  * n = 0
```

Luego

```
Hallando su discriminante :
         discriminante = n^2 -(-m/3)^3
         discriminante = 0^2 +(0/3)^3
        * discriminante = 0
```

ii. Si $\Delta > 0$

Sea la ecuación:

$$1x^3 + 2x^2 + 3x + 4 = 0$$

- Valores calculados:

$$p = \frac{3 * 2 - 1^2}{3} \rightarrow \quad p = 1.6666667$$

$$q = \frac{2 * 2^3 - 9 * 2 * 3 + 27 * 4}{27} \rightarrow \quad q = 2{,}5925925926$$

$$\Delta = 7{,}4074074074$$

Resuelto por factorización tiene como raíces:

$$x_1 = -1.6506291914$$
$$x\_2 = -0.1746854043 \pm 1{,}5468688872i$$

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe
Sea la ecuacion de 3er grado :

      ( 1 )*x^3 + ( 2 )*x^2 + ( 3 )*x + ( 4 ) = 0.

Las soluciones seran :

      x_1 = -1.65063
      x_(2,3) = -0.174685 +/- ( 1.54687 )*i
```

Para m y n

```
Realizando el 2do cambio de variable :
  m = [3*(1)*(3) -(2)^2] / [3*(1)^2]
  * m = 1.66667

  n = [ (2)^3*3 -9*(1)(2)(3) +27*(4)*(1)^2 ]/(1*3)^3
  * n = 2.59259
```

La discriminante.

```
Hallando su discriminante :
       discriminante = n^2 -4*(-m/3)^3
       discriminante = 2.59259^2 +4*(1.66667/3)^3
      * discriminante = 7.40741
```

Otro ejemplo ecuación:

$$1x^3 - 3x^2 + 9x - 5 = 0$$

Valores calculados:

$$p = \frac{3 * 9 - 3^2}{3} \quad \rightarrow \quad p = 6$$

$$q = \frac{2 * 9^3 - 9 * (-3) * 9 + 27 * (-5)}{27} \rightarrow \quad q = 2$$

$$\Delta = 36$$

Resuelto por factorización tiene como raíces:

$$x_1 = 0{,}6725199979$$
$$x\_2 = 1{,}163740001 \pm 2{,}465853273i$$

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe
  Sea la ecuacion de 3er grado :

        ( 1 )*x^3 + ( -3 )*x^2 + ( 9 )*x + ( -5 ) = 0.

  Las soluciones seran :

        x_1 = 0.67252
        x_(2,3) = 1.16374 +/- ( 2.46585 )*i
```

Para m y n

```
Realizando el 2do cambio de variable :
  m = [3*(1)*(9) -(-3)^2] / [3*(1)^2]
  * m = 6

  n = [ (-3)^3*3 -9*(1)(-3)(9) +27*(-5)*(1)^2 ]/(1*3)^3
  * n = 2
```

La discrimínate.

```
Hallando su discriminante :
        discriminante = n^2 -4*(-m/3)^3
        discriminante = 2^2 +4*(6/3)^3
       * discriminante = 36
```

iii. Si $\Delta = 0$

Sea la ecuación:

$$27x^3 + 27x^2 - 72x + 28 = 0$$

- Valores calculados:

$$p = -3$$
$$q = 2$$
$$\Delta = 0$$

Resuelto por factorización tiene como raíces:

$$x_1 = -2.3333333333$$
$$x_{2,3} = 0.666666666$$

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe
  Sea la ecuacion de 3er grado :

        ( 27 )*x^3 + ( 27 )*x^2 + ( -72 )*x + ( 28 ) = 0.

  Las soluciones seran :
        x_1 = -2.33333
        x_(2,3) = 1.16667
```

Para m y n

```
Realizando el 2do cambio de variable :
  m = [3*(27)*(-72) -(27)^2] / [3*(27)^2]
  * m = -3

  n = [ (27)^3*3 -9*(27)(27)(-72) +27*(28)*(27)^2 ]/(27*3)^3
  * n = 2
```

La discrimínate.

```
Hallando su discriminante :
          discriminante = n^2 -4*(-m/3)^3
          discriminante = 2^2 +4*(-3/3)^3
        * discriminante = 0
```

iv. Si

Sea la ecuación:

$$x^3 + 5x^2 - 8x - 42 = 0$$

- Valores calculados:

$$p = -16{,}3\overline{33}$$
$$q = -19{,}\overline{407}$$
$$\Delta = -268{,},\overline{888}$$

Resuelto por factorización tiene como raíces:

$$x_1 = 2{,}8729833462$$
$$x_2 = -4{,}8729833462$$
$$x_3 = -3$$

```
C:\Users\USUARIO\Desktop\ETNL307_2022\L1RMM_all\pseint\ETN307L1_Ejer_3\L1_RMM_E3v2.exe
  Sea la ecuacion de 3er grado :

        ( 1 )*x^3 + ( 5 )*x^2 + ( -8 )*x + ( -42 ) = 0.

  Las soluciones seran :
        x_1 = 2.87298
        x_2 = -4.87298
        x_3 = -3
```

Para m y n

```
Realizando el 2do cambio de variable :
  m = [3*(1)*(-8) -(5)^2] / [3*(1)^2]
  * m = -16.3333

  n = [ (5)^3*3 -9*(1)(5)(-8) +27*(-42)*(1)^2 ]/(1*3)^3
  * n = -19.4074
```

La discrimínate.

```
Hallando su discriminante :
        discriminante = n^2 -4*(-m/3)^3
        discriminante = -19.4074^2 +4*(-16.3333/3)^3
       * discriminante = -268.889
```

Conclusiones. -

El algoritmo realizo bien todos los casos que se le presento.

Observaciones. -

Se presentó un problema para el siguiente caso.

```
a_raiz = -pow(var_aux,(0.33333333));//a_raiz = -pow(var_aux,(1/3));
```

Esta operación se utilizó para cuando la discriminante tiene un valor positivo mayor a 0, sin embargo, como se ve en la imagen hubo que cambiar la forma de expresar la raíz cubica puesto que si se lo escribía como una división el resultado final no correspondía al valor de la raíz.

Link en github. -

https://github.com/MijahelRuelas/Laboratorio_1.git