

**Proyecto de Programación: Teoremas del**

**Seno y del Coseno & Ecuación Cuadrática**

Actividad curricular: Matemática para computación

Carrera: Ingeniería ejecución en Computación e Informática.

Docente: José Miguel Zúñiga Núñez

Estudiantes: Fabián Tejo Castillo.

Matías Flores Leiva.

Claudio Sepúlveda Bustos.

Talca, 30 junio 2025

Contenido

[Tabla de Ilustraciones 3](#_Toc202207143)

[Introducción 4](#_Toc202207144)

[Lenguaje utilizado: Lenguaje de programación C 5](#_Toc202207145)

[Preparaciones Generales 6](#_Toc202207146)

[Resolución Triángulo 8](#_Toc202207147)

[Casos de prueba Triángulos 14](#_Toc202207148)

[Resolución de ecuaciones Cuadráticas: 15](#_Toc202207149)

[Casos de prueba Ecuación cuadrática: 17](#_Toc202207150)

[Diagrama de flujo 18](#_Toc202207151)

[Conclusión 21](#_Toc202207152)

[Bibliografía 22](#_Toc202207153)

# Tabla de Ilustraciones

[Ilustración 1 5](#_Toc202129294)

[Ilustración 2 6](#_Toc202129295)

[Ilustración 3 7](#_Toc202129296)

[Ilustración 4 8](#_Toc202129297)

[Ilustración 5 9](#_Toc202129298)

[Ilustración 6 9](#_Toc202129299)

[Ilustración 7 10](#_Toc202129300)

[Ilustración 8 10](#_Toc202129301)

[Ilustración 9 11](#_Toc202129302)

[Ilustración 10 11](#_Toc202129303)

[Ilustración 11 12](#_Toc202129304)

[Ilustración 12 12](#_Toc202129305)

[Ilustración 13 12](#_Toc202129306)

[Ilustración 14 13](#_Toc202129307)

[Ilustración 15 14](#_Toc202129308)

[Ilustración 16 14](#_Toc202129309)

[Ilustración 17 14](#_Toc202129310)

[Ilustración 18 14](#_Toc202129311)

[Ilustración 19 15](#_Toc202129312)

[Ilustración 20 15](#_Toc202129313)

[Ilustración 21 15](#_Toc202129314)

[Ilustración 22 16](#_Toc202129315)

# Introducción

Desarrollaremos en este proyecto dos programas que abordaran dos problemas matemáticos, el primero es la resolución de triángulos no rectángulos utilizando teoremas del Seno y del Coseno, el cual identificara el caso correspondiente según los datos ingresados. El segundo resolverá ecuaciones cuadráticas utilizando el discriminante para determinar el tipo de soluciones.

El lenguaje elegido es C en ambos programas, debido a que ya tenemos conocimientos previos con el mismo, será una gran oportunidad para reforzar nuestras habilidades de programación aplicando conceptos matemáticos y usando la oportunidad para expandir nuestros conocimientos.

# Lenguaje utilizado: Lenguaje de programación C

Para este proyecto utilizaremos el lenguaje de programación C, es de nivel medio, estructurado, de uso general, que es ampliamente utilizado en el desarrollo de sistemas operativos. Fue creado por Dennis Ritchie en las instalaciones de la empresa AT&T en el año 1972. El objetivo de la creación fue ser borrador del sistema UNIX, pero se dieron cuenta de que era un lenguaje de programación poderoso y manejable, por ese motivo se hizo muy conocido dentro del área de la programación, comenzó a ser muy utilizado e importante en los años 80 y 90.

Características importantes:

* Favorece a que la programación sea estructurada.
* Se puede usar para la programación de alto y bajo nivel.
* Lenguaje de programación portable.
* Flexibilidad y potencia.
* Sencillez, tiene pocas palabras claves, eso genera que sea un lenguaje fácil de aprender.

Es un lenguaje clave en la historia de la programación, mantiene un equilibrio entre simplicidad y potencia, lo cual genera que sea una base sólida para el desarrollo de software.

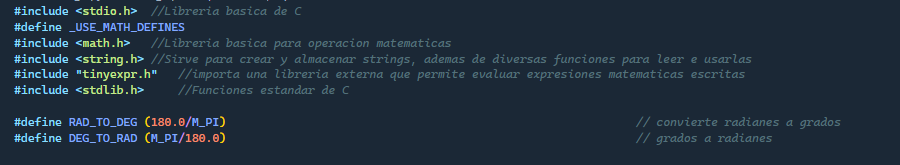


Ilustración 2

# Preparaciones Generales

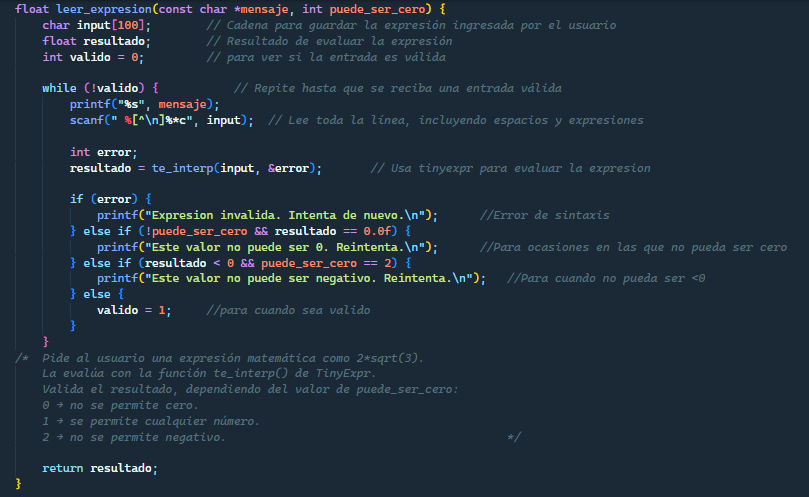
Comenzaremos agregando 5 bibliotecas, la biblioteca básica <stdio.h> que contiene las definiciones para funciones básicas de entrada y salida estándar. La usaremos principalmente para poder usar Printf y Scanf. La segunda biblioteca es <math.h>, con funciones para potencias, raíces, trigonometría y logaritmos con funciones como **sin**, **cos**, acos, **asin** y **sqrt**, simplifica el desarrollo de programas matemáticos, la tercera será <string.h> que nos servirá para poder almacenar y leer caracteres, también <stdlib.h> para cubrir algunas bases extra y por último, pero posiblemente la más importante, “tinyexpr.h”, una librería externa descargada desde GitHub que se encarga de leer las operaciones de los inputs y ejecutarlos directamente.

También tenemos 3 #define, el primero siendo “\_USE\_MATH\_DEFINES” que funciona para darle valores específicos a diversas funciones de la librería <math.h>, después tenemos “RAD\_TO\_DEG” que funciona para transformar los radianes a grados y su inversa “DEG\_TO\_RAD”, ambas necesitan el numero PI (M\_PI) para funcionar así que por eso es importante definirlo primero.



Ilustración

Justo después definimos la función que nos va a ayudar a leer, validar y calcular cada uno de los datos que ingresemos por nuestro input.



Ilustración

Tenemos la función “leer\_expresion” que se encarga de validar las entradas de datos y usa tinyexpr para poder leer las operaciones ingresadas en el input.

Primero creamos una cadena para guardar la expresión que se va a leer, después creamos un float que va a almacenar el resultado y un int llamado “valido” que dependiendo su valor (0 o 1) va a decidir si el input es correcto. Después creamos un bucle while que funciona hasta encontrar una entrada valida y escanea el input completo y lo guarda en la cadena “input”. Por ultimo tenemos 4 casos posibles, el “error” que es cuando nuestro input es invalido, otro para ocasiones en las que el valor no puede ser cero, otro para cuando no pueden ser negativos y el ultimo que permite el ingreso de cualquier número.

Tinyexpr funciona usando el comando te\_interp(), esta toma una expresión e inmediatamente devuelve el resultado de esta.

# Resolución Triángulo

Esta sección del código es el pie de la resolución de triángulos. Comienza confirmando que el usuario selecciono la correcta (opción 2).

Luego se declaran las variables valid con el objetivo de validar el formato del texto que sea ingresado como necesitamos, en este caso números enteros.

Dibujo de triángulo

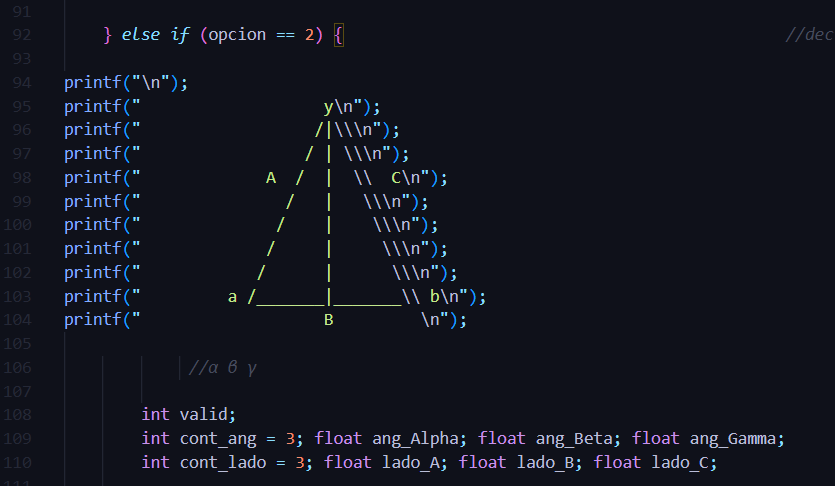
Antes de que el programa solicite datos, imprime un triángulo con las letras A, B y C con el objetivo de identificar vértices y los lados opuestos a cada uno, gran ayuda antes de ingresar los datos al programa.

Int cont\_ang: es definido con 3 porque la cantidad de lados de un triángulo son 3.

* Float ang\_Alpha: Guarda lado del triángulo.
* Float ang\_Beta: Guarda lado del triángulo.
* Float ang\_Gamma: Guarda lado del triángulo.

Int cont\_lado: Contador de lados de un triángulo, por eso su valor de 3.

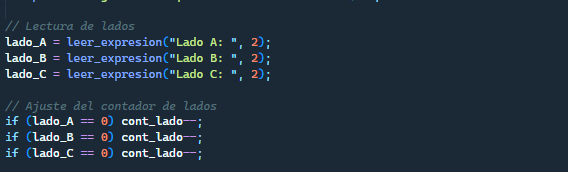
* Float lado\_A: variable float(valor con decimal) guarda el valor real del primer lado con el objetivo de más adelante determinar si es un caso LLL, LAL o AAL.
* Float lado\_B: variable float(valor con decimal) guarda el valor real del segundo lado con el objetivo de más adelante determinar si es un caso LLL, LAL o AAL.
* Float lado\_C: variable float(valor con decimal) guarda el valor real del tercer lado con el objetivo de más adelante determinar si es un caso LLL, LAL o AAL.



Ilustración

Se imprime en pantalla “Ingresa Lados, si es desconocido un 0: “, luego se definen los lados.

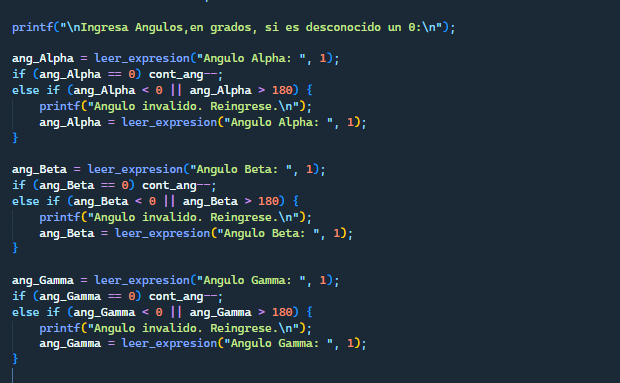
El lado A, B y C utiliza una función personalizada “leer\_expresion()” para evaluar expresiones matemáticas, luego se asigna 2 esto permite que el usuario escriba 0 si no conoce el valor, pero no permite negativos.



Ilustración

Luego solicita mediante Printf: “Ingresa Ángulos, en grados, si es desconocido un 0”. Luego como podemos ver es solicitado cada ángulo y realiza la siguiente acción

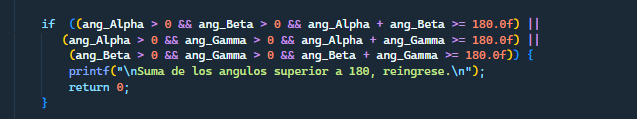
* Revisión de valor 0 mediante if, si los ángulos ingresados son igual a 0 se le resta 1 al contador cont\_ang.
* Revisión del rango mediante else if, si el valor es menor a cero o mayor que 180 grados, se pide reingresar los datos.
* Usamos la misma función de “leer\_expresion” junto con algunos if y else if para validar y leer los ángulos.



Ilustración

Validación de la suma de ángulos

Válida si dos ángulos suman 180 o más, no podría formar un ángulo plano, interrumpe el programa.



Ilustración

Caso 1: Lado, Lado, Lado (LLL)

if (cont\_lado==3 && con\_ang == 0) si se ingresan 3 lados, no los ángulos, se ocupa el teorema del coseno para calcular los ángulos.

Luego mediante un if se valida la desigualdad triangular: if ((lado\_A + lado\_B <= lado\_C) || (lado\_B + lado\_C <= lado\_A) || (lado\_C + lado\_A <= lado\_B)), asegurando que con los datos ya ingresados se pueda formar un triángulo, si no se cumple, aparece un mensaje en pantalla de “No se cumple la desigualdad triangular, reingrese”.

Si la desigualdad triangular se cumple se aplican las fórmulas del teorema del coseno con ang\_Alpha, ang\_Beta, ang\_Gamma. Las funciones ACOS devuelve el ángulo en radiantes, luego se multiplica por RAD\_TO\_DEG para convertir los grados a sexagesimales.

¿Por qué se transforman a sexagesimal?

Las funciones trigonométricas en C se trabajan en radianes, pero para mostrar los resultados al usuario, se necesitan grados sexagesimales(sistema de 0° a 180°). Simplifica el resultado, se vuelve más comprensible al usuario final.

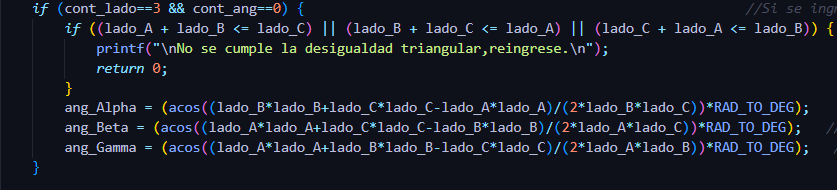


Ilustración 9

Caso 2: Lado, Ángulo, Lado (LAL)

Se evalúa mediante else if (cont\_lado==2 && cont\_ang==1), que se ha ingresado dos lados y un ángulo que no está entre ellos.

Aplica el teorema del coseno para calcular el tercer grado.

Se aplica el teorema del SENO “ang\_Alpha = asin((sin(Gamma) \* A) / C)”.

Para finalizar, se obtiene el tercer ángulo restando a 180° los otros dos:

“ang\_Beta = 180.0f - ang\_Alpha – ang\_Gamma"

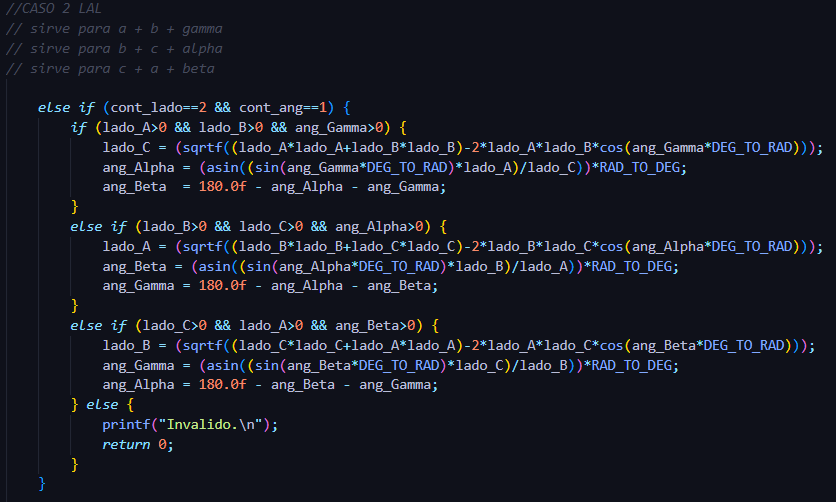


Ilustración 10

Realiza la misma acción y comprobación con los dos siguientes ángulos:

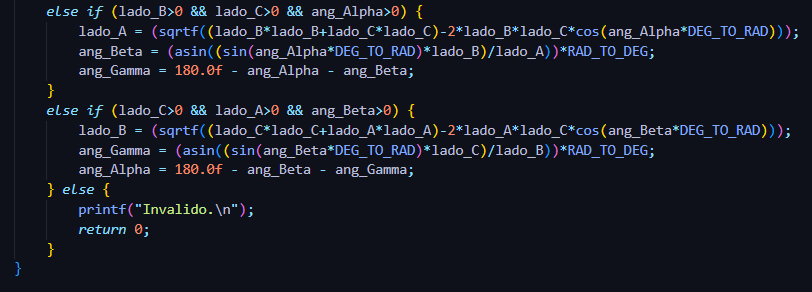
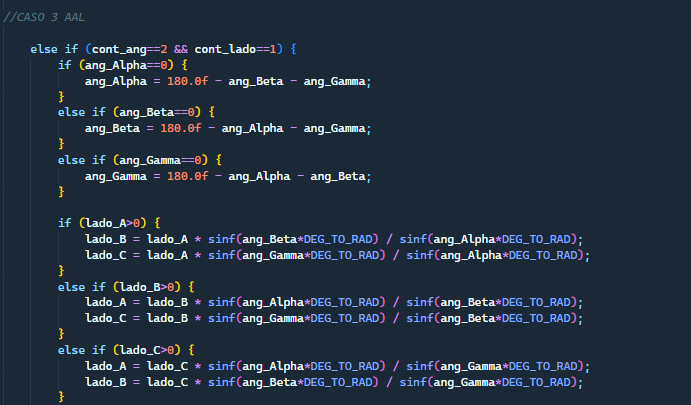


Ilustración 11

Al no cumplirse las condiciones necesarias para resolver el triángulo, el programa imprimirá en pantalla “inválido” y se finalizará la ejecución.

Caso 3: Ángulo, Ángulo, Lado (AAL)

Se evalúa mediante else, if (cont\_ang==2 && cont\_lado==1), se conocen dos ángulos y un lado, el programa completa el ángulo con:



Ilustración

Luego se aplica el teorema del SENO para encontrarse los otros lados, según el lado conocido:

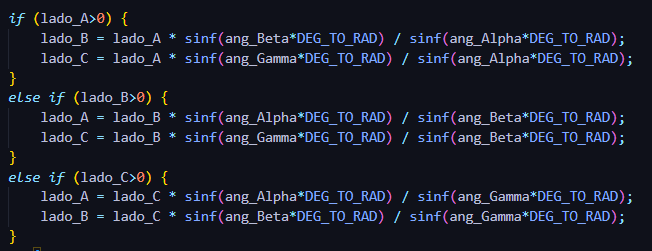


Ilustración 13

Caso no válido: si no se reconoce correctamente los datos ingresados, se imprime en pantalla una ayuda textual con las combinaciones válidas para el usuario. Si no se cumple ningún caso, se termina el programa:

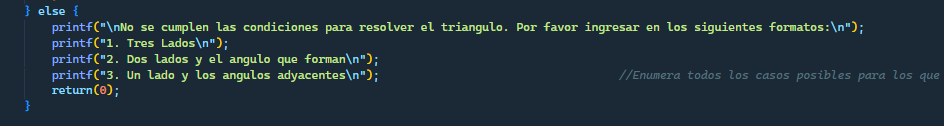


Ilustración 14

Muestra de resultados

Luego de que el usuario ingrese los datos de forma correcta y siguiendo los pasos del programa, llegamos a la sección de imprimir los resultados en pantalla, se realiza luego de cumplir con los pasos anteriormente mencionados en el caso de que sea ingresado.

Se imprimen los tres lados y los tres es a ángulos con decimales:

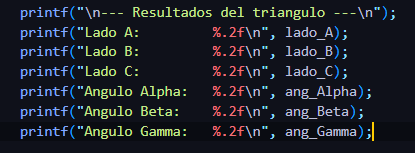
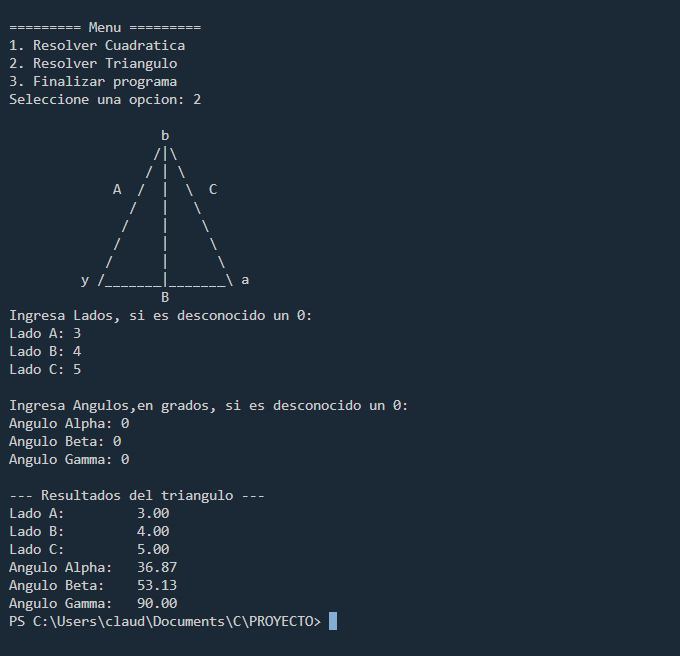
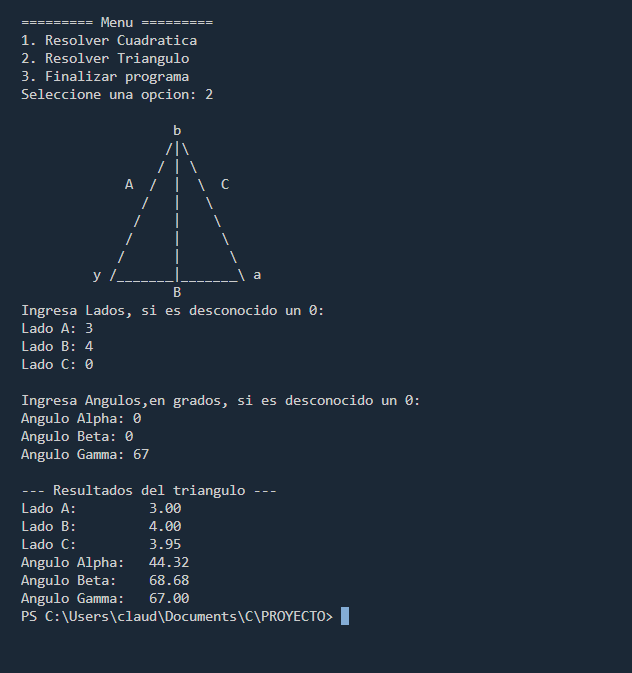


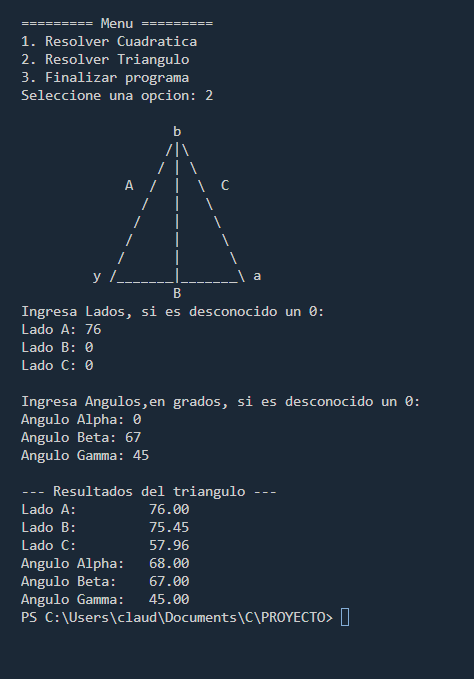
Ilustración 15

# Casos de prueba Triángulos



Ilustración

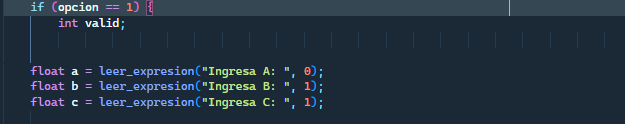
Ilustración



Ilustración

# Resolución de ecuaciones Cuadráticas:

Entrada de lados



Ilustración

Después de ingresar la opción 1 en el menú, se crean los 3 floats que nos van a servir para cada coeficiente de la ecuación cuadrática, usamos también la función de “leer expresión” en el modo 0 (en este modo no se permiten 0 en el input) y en el modo 1 (este acepta todo tipo de números)

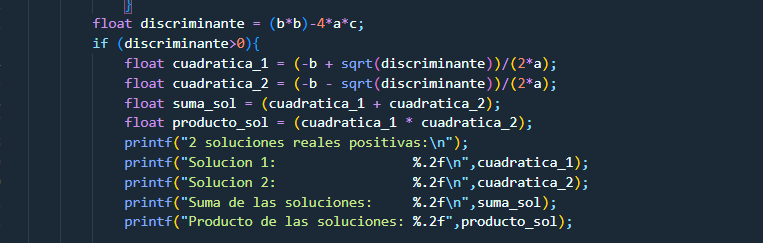


Ilustración 20

Después creamos un float con la fórmula del discriminante, está la usaremos en todos los casos así que está bien incluso si la declaramos afuera de los if/ else if, con las ecuaciones cuadráticas sabemos que existen 3 casos contados y todos ellos se clasifican por su discriminante, así que es buena opción dejarlo como la condición principal de los if, después aplicamos fórmulas simples y se printean los resultados, este es el primer caso con el discriminante mayor 0. En todos los casos para obtener el discriminante usaremos la función sqrt() que sirve para obtener la raíz cuadrada de un número.

La fórmula cuadrática se utiliza para resolver ecuaciones de segundo grado en donde a,b y c son conocidos y a no puede ser cero, las soluciones se obtienen de la siguiente manera: X1 y X2 = (-B +/- sqrt(b^2-4AC)/2A), es lo mismo que esta escrito en el codigo pero con lenguaje C, después se filtra por su discriminante sqrt(b^2-4AC) y calcula lo necesario.

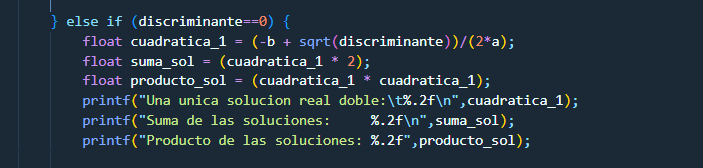


Ilustración 21

En el segundo caso seguimos usando las mismas variables, solamente que en este caso el discriminante es 0, así que lo que nos encontremos será una única solución real doble

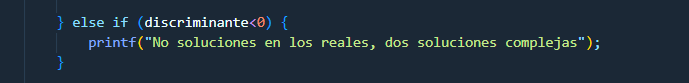


Ilustración 22

Y el último caso cuando el discriminante es negativo así que simplemente no tenemos solución en los reales, imprimimos el mensaje y termina el programa.

# Casos de prueba Ecuación cuadrática:

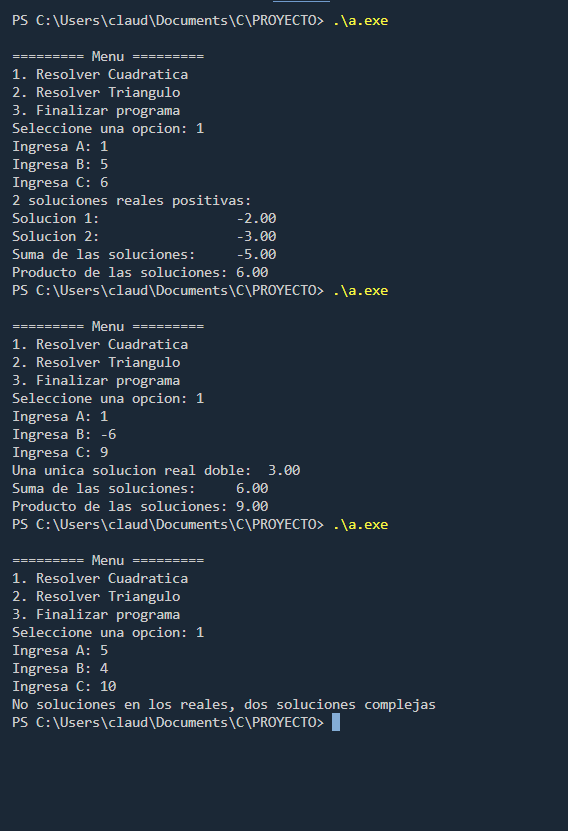


Ilustración 23

Diagrama de flujo  
   
Inicio del programa

El programa comienza presentando un menú con tres opciones:

1. Resolver una ecuación cuadrática.

2. Resolver un triángulo.

3. Salir.

Lectura y validación de la opción

• El usuario ingresa su elección.

• El programa checking si es una opción válida.

• Si no lo es, le muestra un mensaje de error y la vuelve a solicitar.

• Si es válida, pasa al siguiente paso dependiendo de la opción.

Opción 1: Resolver ecuación cuadrática

• Se piden los coeficientes A, B y C.

• A no puede ser 0.

• Se calcula el discriminante D = B² - 4AC.

Dependiendo del valor de D:

• D > 0: Hay dos raíces reales. Se calculan e imprimen.

• D == 0: Hay una raíz doble. Se imprime.

• D < 0: No hay raíces reales. El programa informa esto.

Finaliza esta opción.

Opción 2: Resolver un triángulo

• Se muestra un esquema ASCII del triángulo.

• Se piden los lados A, B y C (0 si se desconoce).

• Se cuentan cuántos lados se conocen.

• Se piden los ángulos α, β y γ (también con 0 si se desconocen).

• Se verifica que la suma parcial de los ángulos no supere 180°.

Según los datos ingresados, se sigue una estrategia:

1. Tres lados conocidos (LLL):

• Usar la Ley del Coseno para calcular los ángulos.

2. Dos lados y un ángulo (LAL o ALA):

• Usar el coseno para calcular el tercer lado.

• Luego aplicar la Ley del Seno para calcular los otros ángulos.

3. Un lado y dos ángulos (AAL):

• Calcular el tercer ángulo por resta.

• Luego usar la Ley del Seno para calcular los lados faltantes.

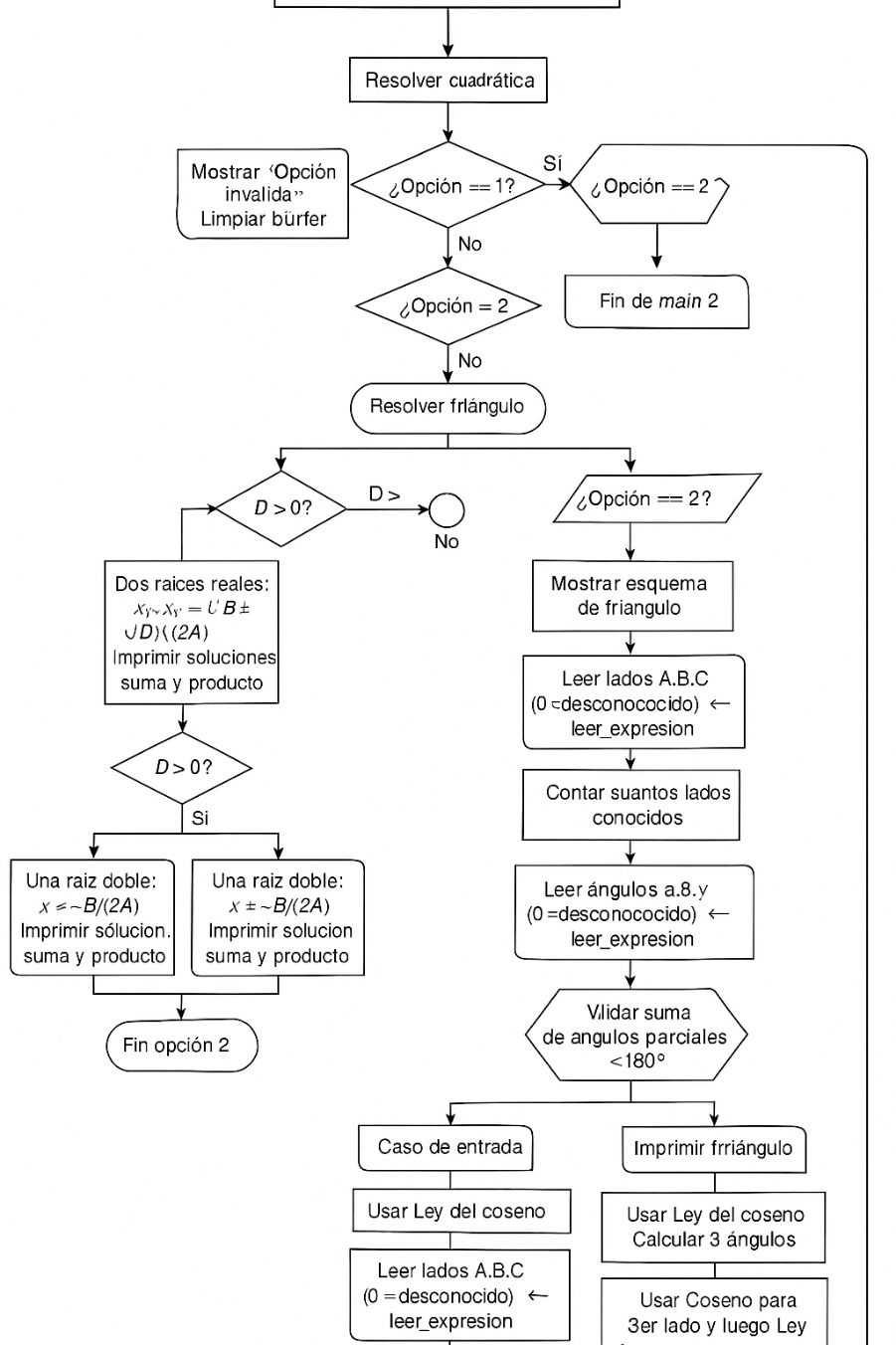
4. Caso inválido:

• Se muestra un mensaje indicando que la entrada no es válida.

Opción 3: Salir

• El programa muestra el mensaje “Programa finalizado”.

• Luego termina (return 0).



# Conclusión

Creo que como grupo nuestro obstáculo más grande durante el transcurso del proyecto fue la complejidad de los inputs, el trabajar con inputs simples de int o floats no es complicado, pero el hecho de querer que el programa pueda reconocer cuando operaciones son aplicadas y resolverlas es un paso superior a nuestros conocimientos, por eso empezamos a revisar en diversas fuentes para intentar solucionar este problema, primero intentamos hacer alguna especia de varios int que se separaran después de detectar cada operador de manera individual, pero pronto nos dimos cuenta que esto es prácticamente imposible, entonces uno de nosotros encontró en GitHub una publicación de una librería llamada “Tinyexpr”, aunque nos costó un poco entender su funcionamiento y forma de escribir una función con ella que funcionara pronto nos dimos cuenta que era exactamente lo que estábamos buscando, nos permitía escribir directamente las operaciones en los inputs y la función se encargaba de hacer todo automáticamente, lamentablemente después tuvimos problemas con el formato en que se guardaban nuestros datos así que terminamos por reescribir todos los inputs y aplicamos esta función. Además, se complementa de manera muy buena con la librería de <math.h>, ambos tienen el mismo formato base, por ejemplo, sqrt() hace los mismo en ambas.

Si tuviera que comentar otra dificultad que tuvimos serían las fórmulas de los triángulos, por si solas no son complejas, pero se van acumulando hasta ocupar un tamaño considerable en el código , además hacer que el programa reconozca de manera autónoma que caso usar para resolver un triángulo crea problemas innecesarios cuando el usuario perfectamente podría ingresar que caso quiere usar y que se adapte de mejor manera a sus necesidades, además de que de esa manera se podrían controlar de mejor manera las variaciones entre inputs.

En este proyecto desarrollamos programas que abordan problemas matemáticos. El primero resuelve triángulos no rectángulos aplicando los teoremas del Seno y del Coseno, identificando automáticamente el caso según los datos ingresados.

El segundo resuelve ecuaciones cuadráticas utilizando el discriminante para determinar el tipo de soluciones. Ambos fueron implementados en lenguaje C, ya que contamos con conocimientos previos y nos permite reforzar habilidades de programación.

# Bibliografía

* <https://informatica.uv.es/estguia/ATD/apuntes/laboratorio/Lenguaje-C.pdf>
* <https://repositorio.une.edu.pe/server/api/core/bitstreams/c43192c0-a8ae-4148-838e-9b4b1dc60064/content>
* <https://github.com/codeplea/tinyexpr>