

Funciones Segundo Grado testing

MP 0487



29 de noviembre de 2024

Daniel Garcia Brun

1DAM - MP 0487 - Entorns de Desenvolupament

Contenido

[Explicación código: 2](#_Toc183775133)

[Declaración Variables 2](#_Toc183775134)

[Función pedir datos 2](#_Toc183775135)

[Función discriminante 2](#_Toc183775136)

[Función calcular raizes 3](#_Toc183775137)

[Función mostrar resultados 3](#_Toc183775138)

[Main 3](#_Toc183775139)

[Código Funciones 4](#_Toc183775140)

[Testing 5](#_Toc183775141)

[Llamar Funciones 5](#_Toc183775142)

[Discriminante negativo 5](#_Toc183775143)

[Discriminante Cero 6](#_Toc183775144)

[Discriminante positiva 6](#_Toc183775145)

[Ejecución Pruebas 7](#_Toc183775146)

[Código Test 8](#_Toc183775147)

# Explicación código:

## Declaración Variables

Primero llamamos a las funciones necesarias y declaramos las variables que usaremos, en este caso dobles ya que los números pueden contener decimales.

Interfaz de usuario gráfica, Texto, Chat o mensaje de texto

Descripción generada automáticamente

## Función pedir datos

Creamos una función para pedir datos y le añadimos un bucle do while ya que a no puede ser 0

Texto

Descripción generada automáticamente

## Función discriminante

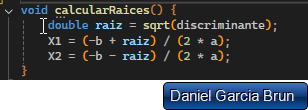
Ahora con otra función calculamos el discriminante

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

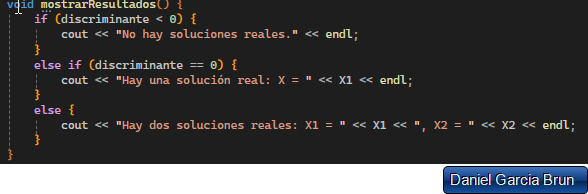
## Función calcular raíces

En una tercera función calculamos las raíces y hacemos las dos posibilidades con b+ raíz o b - raíz



## Función mostrar resultados

Ahora creamos la que sería la última función donde pondremos todas las posibilidades caso de que discriminante es menor que 0, caso de que discriminante se igual a 0 y el caso que es mayor que 0



## Main

Ahora en el main debemos invocar a las funciones lo ordenamos para que primero ejecute pedir daros, luego calcule el discriminante con un if para que si es menor que 0 no calcule la raíz y para finalizar muestre los resultados

Texto, Chat o mensaje de texto

Descripción generada automáticamente

# Código Funciones

#include <iostream>

#include <cmath>

using namespace std;

double a, b, c, discriminante, X1, X2;

void pedirDatos() {

do {

cout << "Ingresa un valor mayor que 0 para a: ";

cin >> a;

} while (a == 0);

cout << "Ingresa un valor para b: ";

cin >> b;

cout << "Ingresa un valor para c: ";

cin >> c;

}

void calcularDiscriminante() {

discriminante = (b \* b) - 4 \* a \* c;

}

void calcularRaices() {

double raiz = sqrt(discriminante);

X1 = (-b + raiz) / (2 \* a);

X2 = (-b - raiz) / (2 \* a);

}

void mostrarResultados() {

if (discriminante < 0) {

cout << "No hay soluciones reales." << endl;

}

else if (discriminante == 0) {

cout << "Hay una solución real: X = " << X1 << endl;

}

else {

cout << "Hay dos soluciones reales: X1 = " << X1 << ", X2 = " << X2 << endl;

}

}

int main() {

pedirDatos();

calcularDiscriminante();

if (discriminante >= 0) {

calcularRaices();

}

mostrarResultados();

return 0;

}

# Testing

## Llamar Funciones

Ahora veremos el código del testing:

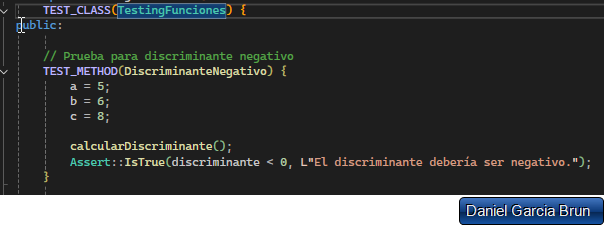
Primero de todo llamamos a las funciones y buscamos donde tenemos el código de C++ de las funciones para que pueda probar el testing

Texto

Descripción generada automáticamente

## Discriminante negativo

Ahora crearemos el Test method para el caso que el discriminante es negativo, como podemos observar se les asignan valores a las variables y llamamos a la variable, después en el assert si es verdadero que discriminante es menor que 0 lo de bien, en caso de que no sea menor que 0 imprimir por pantalla que el discriminante debería ser menor que 0



## Discriminante Cero

Ahora crearemos otra para el caso que el discriminante sea 0 esto es igual si es verdad la prueba será correcta si no es cero dará que el discriminante sea 0 aparte de esto también comprobara si X1 y X2 son iguales

Texto

Descripción generada automáticamente

## Discriminante positiva

Para finalizar haremos una prueba para el tercer caso que es el que el discriminante será positivo primero comprobamos que el discriminante sea mayor a 0 y después que X1 y X2 sean diferentes entre si

Texto

Descripción generada automáticamente

## Ejecución Pruebas

Ahora para comprobar que funcione le damos a prueba explorador de pruebas le damos a ejecutar todas las pruebas y debería verse así:

Captura de pantalla con la imagen de una pantalla

Descripción generada automáticamente

En caso de haber algo mal se vería así con el mensaje de error de esta forma sabríamos que función en específico está fallando

Captura de pantalla de computadora

Descripción generada automáticamente

# Código Test

#include "pch.h"

#include "CppUnitTest.h"

#include "../FuncionesSegundogradoV2/FuncionesSegundogradoV2.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace TestingFunciones {

TEST\_CLASS(TestingFunciones) {

public:

// Prueba para discriminante negativo

TEST\_METHOD(DiscriminanteNegativo) {

a = 5;

b = 6;

c = 8;

calcularDiscriminante();

Assert::IsTrue(discriminante < 0, L"El discriminante debería ser negativo.");

}

// Prueba para discriminante igual a cero

TEST\_METHOD(DiscriminanteCero) {

a = 1;

b = 2;

c = 1;

calcularDiscriminante();

Assert::IsTrue(discriminante == 0, L"El discriminante debería ser cero.");

calcularRaices();

Assert::AreEqual(X1, X2, L"Las raíces deberían ser iguales.");

}

// Prueba para discriminante positivo

TEST\_METHOD(DiscriminantePositivo) {

a = 1;

b = -3;

c = 2;

calcularDiscriminante();

Assert::IsTrue(discriminante > 0, L"El discriminante debería ser positivo.");

calcularRaices();

Assert::AreNotEqual(X1, X2, L"Las raíces deberían ser diferentes.");

}

};

}