**ANÁLISIS DEL DISEÑO DE UN CÓDIGO**

Código fuente escogido:

#include <iostream>

using namespace std;

class Recta{

    private:

        double a, b, c;

        bool validez(double a, double b){

            bool boolean = false;

            if(a != 0 || b != 0)

                boolean = true;

            return boolean;

        }

    public:

        Recta(double a, double b, double c){

            if(validez(a, b)){

                this -> a = a;

                this -> b = b;

            }

            else cout << "No valido \n";

            this -> c = c;

        }

        void setCoeficientes(double a, double b, double c){

            if(validez(a, b)){

                this -> a = a;

                this -> b = b;

            }

            else cout << "No valido \n";

            this -> c = c;

        }

        double pendiente(){

            return -(a/b);

        }

        double ordenadaY(double x){

            return (-c - x\*a)/b;

        }

        double ordenadaX(double y){

            return (-c - y\*b)/a;

        }

        void setA(double n){

            this -> a = n;

        }

        void setB(double n){

            this -> b = n;

        }

        void setC(double n){

            this -> c = n;

        }

};

int main(){

    Recta a(0, 0, 5);

}

Modularidad: En el código empleado se ha implementado una clase, que hace una tarea especifica y por lo tanto tiene una modularidad alta

Abstracción de datos: La abstracción de datos empleada es una abstracción procedimental ya que cada método de la clase hace una función especifica

Ocultamiento de la información: Los atributos y métodos propios de la clase esta correctamente protegidos y además se emplea en el método validez variable local en vez de una global

Independencia Modular:

Cohesión: Mi módulo es una clase y presenta un alto nivel de cohesión ya que modela una recta y todos los procedimientos sirven para ese propósito es decir darle valores a la recta, obtener pendiente etc.…

Acoplamiento: tiene un nivel bajo ya que es una sola clase y no tiene herencia ni interacciones con otras clases