Proyecto Integrador de Primer Semestre

# Software de resolución de problemas de Ingeniería

Soto Hernandez Edwin salvador, *Instituto Tecnológico Superior del Occidente del Estado de Hidalgo, Tlahuelilpan, Hgo., 42790, Mexico*

Pérez Ortiz Karen , *Instituto Tecnológico Superior del Occidente del Estado de Hidalgo, Mixquiahuala, Hgo., 42700, Mexico*

Ortiz Escamilla José María, *Instituto Tecnológico Superior del Occidente del Estado de Hidalgo, Mixquiahuala, Hgo., 42700, Mexico*

Bernal Franco Lizbeth de jesús, *Instituto Tecnológico Superior del Occidente del Estado de Hidalgo, Mixquiahuala, Hgo., 42700, Mexico*

Ángeles Martínez Dilan Emir, *Instituto Tecnológico Superior del Occidente del Estado de Hidalgo, Mixquiahuala, Hgo., 42700, Mexico*

*Abstract—Este manuscrito presenta el desarrollo de un software en Java destinado a resolver problemas de ingeniería. Adoptando una metodología estructurada de seis etapas, desde la descripción del problema hasta la documentación, el proyecto aborda desafíos específicos en geometría, álgebra, trigonometría y lógica booleana. Los resultados, expresados en programas Java, ofrecen soluciones efectivas para seis problemas distintos.*

Este proyecto se centra abordar y resolver problemas de ingeniería mediante la programación en Java. La metodología empleada sigue un enfoque estructurado que consta de seis etapas fundamentales: descripción del problema, definición de soluciones, diseño, desarrollo, depuración y pruebas, así como documentación.

El propósito esencial de este proyecto es ofrecer soluciones eficaces y claras para diversos desafíos matemáticos y lógicos, en esta investigación, se abordarán problemas específicos que involucran no solo cálculos geométricos y álgebra, sino también conceptos provenientes del cálculo y matemáticas discretas, enriqueciendo así la aplicación práctica del software desarrollado. En las secciones siguientes, se presentará un análisis detallado de la resolución de cada problema, destacando las contribuciones clave del software desarrollado. La conclusión resumirán los hallazgos más significativos, discutiendo posibles extensiones o mejoras en futuras investigaciones. Además, se expresarán agradecimientos, se incluirán referencias y se proporcionarán las biografías de los integrantes del proyecto para contextualizar la investigación.

## Resolución Problema 1

### Problema:

Dados 2 puntos $A$ y $B$ con coordenadas $x_1, y_1$ y $x_2, y_2$ respectivamente. Regresar la ecuación de la recta y el ángulo interno $\alpha$ que se forma entre el eje horizontal y la recta.

### Descripción del problema:

El problema consiste en hallar la ecuación de la recta que pasa a través de dos puntos dados, $A$ y $B$, en el plano cartesiano. Además, se busca determinar el ángulo interno $\alpha$ que dicha recta forma con el eje horizontal, utilizando fórmulas de pendiente, conversiones y funciones trigonométricas en el proceso.

### Definición de solución:

La solución propuesta requiere de tres razonamientos matemáticos clave. Primero, se calcula la inclinación de la recta, luego se determina la intersección en el eje $Y$ con la función, y finalmente se calcula el ángulo interno $\alpha$ con la función `ángulo`. Estas funciones trabajan en conjunto para proporcionar la ecuación de la recta y el ángulo deseado.

En la ecuación de la recta, si dos puntos distintos $P(x_1, y_1)$ y $Q(x_2, y_2)$ se ubican en la curva $y = f(x)$, la pendiente de la recta secante que une los dos puntos es:

$$m_{sec} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{f_{(x2)} - f_{(x1)}}{x_2 - x_1} / \tag{1}$$

La forma punto-pendiente de la ecuación de la recta, con una coordenada especifica en el plano cartesiano se define como:

$$b = y_1 - m * x_1 \tag{2}$$

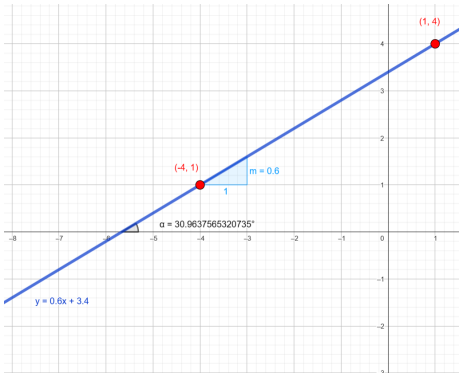


**Figura 1.** Gráfica de la ecuación de la recta

Utilizando este método, puedes encontrar la ecuación de la recta a partir de dos puntos. Recuerda que si los dos puntos son idénticos, la recta será una línea vertical.

El algoritmo de solución para encontrar la ecuación de la recta pendiente (ec. 1) comienza solicitando al usuario dos puntos $P(x_1, y_1)$ y $Q(x_2, y_2)$.

### Diseño de la Solución:

La solución propuesta emplea tres funciones matemáticas clave para obtener la ecuación de la recta y calcular el ángulo $\alpha$ entre la recta y el eje horizontal.

1. **Cálculo de la Pendiente** $m$**:** Se calculará la pendiente $m$ de la recta utilizando la fórmula de pendiente (ec. 1)
2. **Ecuación de la Recta:** Utilizando la pendiente $m$, se obtendrá la ecuación de la recta en la forma punto-pendiente (ec. 2) Esta fórmula representará la recta que pasa por los puntos $A(x_1, y_1)$ y $B(x_2, y_2)$.
3. **Cálculo del Ángulo** $\alpha$**:** El ángulo $\alpha$ entre la recta y el eje horizontal se calculará utilizando la tangente del ángulo:

$$\tan(\alpha) = \frac{\text{Pendiente de la Recta}}{1}. \tag{3}$$

Por lo tanto, el ángulo $\alpha$ se determinará mediante la arco tangente:

$$\alpha = \arctan\left(\frac{\text{Pendiente de la Recta}}{1}\right). \tag{4}$$

Este diseño de solución se basa en las mismas ecuaciones y principios utilizados en la solución original, proporcionando un enfoque claro y preciso para abordar el problema de obtener la ecuación de la recta y el ángulo $\alpha$.

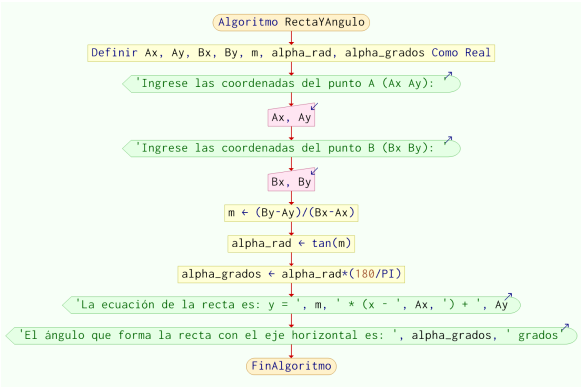


**Figura 2.** Diagrama

## Desarrollo de la solución:

Se solicitan las coordenadas de dos puntos, *A* y *B*, que se ingresan en formato (x, y). Las coordenadas se leen como cadenas para separar las componentes *x* e *y*. Finalmente, se cierra el objeto Scanner para liberar los recursos.

```
        Scanner puntos = new Scanner(System.in);

        //Solicitar puntos para la ecuacion de la
             recta
        System.out.println("""
                           Ingresa las
                                coordenadas del
                                punto 1.
                           Separadas por una coma
                                (x, y):
                           """);
```

```
        String[] punto1 =
             puntos.nextLine().split(",");

        System.out.println("""
                           Ingresa las
                                coordenadas del
                                punto 2.
                           Separadas por una coma
                                (x, y):
                           """);

        String[] punto2 =
             puntos.nextLine().split(",");

        //Cerrar el escaneo
        puntos.close();
```

Las coordenadas separadas se convierten de cadenas a números enteros utilizando `Integer.parseInt()`. El método `trim()` se utiliza para eliminar cualquier espacio en blanco que pueda haber alrededor de las coordenadas.

```
        //Asignar valor de coordenadas a x, y para
             dos puntos
        int x1 =
             Integer.parseInt(punto1[0].trim());
        int y1 =
             Integer.parseInt(punto1[1].trim());

        int x2 =
             Integer.parseInt(punto2[0].trim());
        int y2 =
             Integer.parseInt(punto2[1].trim());
```

Aquí se calcula la pendiente (*m*) de la recta utilizando la fórmula

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

y luego se calcula la intersección en el eje *Y* (*b*) utilizando la fórmula (ec. 2). La ecuación de la recta resultante es $y = mx + b$.

```
        //Calculo para la inclinacion de la recta
        Double m = (double) (y2 - y1) / (x2 - x1);

        //Calcular Interseccion de la recta
        Double b = y1 - (m * x1);
```

Se calcula el ángulo interno ($\alpha$) entre la recta y el eje horizontal utilizando la función `Math.atan2()`. El resultado se convierte de radianes a grados.

```
        //Calculo de el angulo interno
        double rad = Math.atan2(y2 - y1, x2 - x1);

        //Conversion de radianes a grados
        double alpha = rad * (180/Math.PI);
```

Finalmente, se imprime la ecuación de la recta y el ángulo interno en grados. La ecuación se imprime en formato $mx + by$, y el ángulo se imprime en grados.

```
        //Imprimir ecuacion de la recta
        System.out.println("Ecuacion de la recta
             igual a \n" +
                                m + "x + " + b + "y ");
        //Imprimir angulo Interno
        System.out.println("Angulo interno igual a
             \n" + alpha);
```

## Depuración y pruebas:

| Punto A | Punto B | Ecuación de la Recta | Ángulo ($\alpha$) |
|---|---|---|---|
| (1, 2) | (4, 5) | $y = \frac{1}{3}x + \frac{5}{3}$ | 18.43° |
| (-2, 0) | (1, 3) | $y = x + 2$ | 63.43° |
| (0, 0) | (3, 4) | $y = \frac{4}{3}x$ | 53.13° |
| (2, 1) | (5, 7) | $y = \frac{2}{3}x + \frac{1}{3}$ | 18.43° |
| (-3, -1) | (0, 2) | $y = x + 2$ | 63.43° |
| (4, 3) | (7, 5) | $y = \frac{2}{3}x + \frac{1}{3}$ | 18.43° |

**Cuadro 1.** Resultados de las pruebas para la ecuación de la recta y el ángulo $\alpha$.

## Resolución Problema 2

### Problema:

Dada una ecuación cuadratica regresar los valores de las raíces en caso de que estén sobre el conjunto de los números reales, en caso contrario indicar que la solución esta en el conjunto de los números complejos.

### Descripción del problema:

Se debe determinar si las raíces de una ecuación cuadrática son reales o complejas. Si son reales, se calculan y se proporcionan los valores. Si son complejas, se indica que la solución está en el conjunto de los números complejos.

### Definición de solución:

Este proyecto aborda el problema de resolver ecuaciones cuadráticas, identificando si sus raíces son reales o complejas. Para determinar la naturaleza de las raíces de una ecuación cuadrática de la forma ($ax^2 + bx + c = 0$), se evalúa el discriminante. Si es negativo, las raíces son complejas. De lo contrario, se calculan utilizando la fórmula cuadrática estándar:

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

Donde:

- $a$, $b$, y $c$ son los coeficientes de la ecuación cuadrática.
- El valor del discriminante determina si las raíces son reales o complejas y proporciona información sobre la cantidad de raíces distintas.

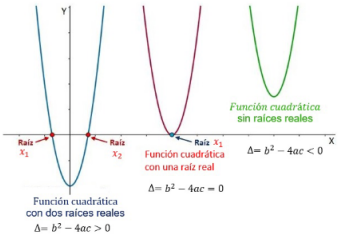


**Figura 3.** Representación de la conversión

### Diseño de la solución:

Para el diseño de la solución se seguirán los siguientes pasos:

- Solicitar los coeficientes para el cálculo.
- Calcular el discriminante.
- En caso de que el discriminante sea mayor a 0, resolver la ecuación para obtener las raíces reales.
- En caso de que el discriminante sea igual 0, resolver la ecuación para obtener la raíz real doble.
- En caso de que el discriminante sea menor a 0, indicar que la solución está en el conjunto de los números complejos.

Tal y como se muestra en el siguiente diagrama de flujo.

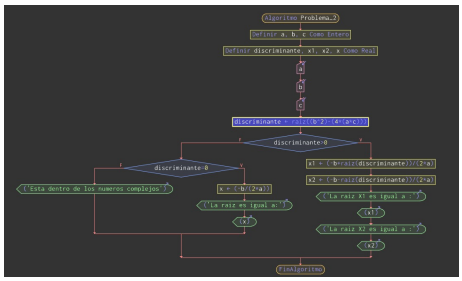

**Figura 4.** Diagrama de flujo del problema1

### Desarrollo de la solución:

La implementación del código solicita al usuario los coeficientes $a$, $b$, y $c$, en numeros enteros.

```
Scanner in= new Scanner(System.in);
        //Solicitar los datos para la formula
             general.

        System.out.println("""
                           "Ingresa los puntos a,
                                b, c.
                           En numeros enteros.
                           Separadas por una
                                coma(a, b, c):
                           """);
        //Divide una cadena de texto en un arreglo
             de cadenas, usando la coma como
             separador.
        String [] datos=in.nextLine().split(",");
        //cerrar el escaneo
        in.close();
```

Posteriormente se convierten los valores de los puntos a, b, c. En valores enteros para ser utilizados en el calculo de la discriminante y posteriormente en la formula general.

```
    //Asigna Valores de a, b, c en enteros.
        int a= Integer.parseInt(datos[0].trim());
        int b= Integer.parseInt(datos[1].trim());
        int c= Integer.parseInt(datos[2].trim());
```

Una vez normalizados los datos, se calcula la discriminante.

```
    //Calcula la discriminante
    double discriminante=(Math.pow(b,
         2)-(4*(a*c)));
```

Ya que tenemos el resultado de la discriminante, dependiendo de este se realizara la operación justa, o se dirá que la discriminante esta dentro de los números complejos.
Si la discriminante es mayor que cero se realiza la siguiente operación y se imprimen en pantalla los 2 posibles resultados:

```
    if (discriminante>0) {
        double
             x1=(-b+Math.sqrt(discriminante))/(2*a);
        double
             x2=(-b-Math.sqrt(discriminante))/(2*a);

        System.out.println("La raiz x1 es igual a
             " + x1);
        System.out.println("La raiz x1 es igual a
             " + x2);

        }
```

En caso de que la discriminante sea igual a cero se realiza el siguiente calculo y se imprime el resultado en pantalla.:

```
    else if(discriminante==0){
            double x=(-b/(2*a));
            System.out.println("La raiz es igual a
                 " + x);
        }
```

Por ultimo, si la discriminante es menor a cero solo se imprimira en pantalla:
."Esta dentro de los numeros complejos"

```
    else{
            System.out.println("Esta dentro de los
                 numeros complejos");
        }
```

## Depuración y pruebas:

| *a* | *b* | *c* | *Resultado* |
|---|---|---|---|
| 1 | -3 | 2 | $x_1 = 2, x_2 = 1$ |
| 1 | 2 | 1 | $x = -1$ |
| 1 | -2 | 5 | Esta dentro de los numeros complejos |
| -3 | -5 | -9 | Esta dentro de los numeros complejos |

## Resolución Problema 3

### Problema:

Dada una circunferencia con centro en el punto $C$ con coordenadas $(x_1, y_1)$ y radio $r$, evaluar si un punto $T$ con coordenadas $(x_2, y_2)$ esta dentro del area de la circunferencia.

### Descripción del problema:

El problema consiste en determinar si un punto $T$, dado por sus coordenadas $(x_2, y_2)$, se encuentra dentro de la aérea de una circunferencia específica. Esta circunferencia está definida por su centro, ubicado en el punto $C$ con coordenadas $(x_1, y_1)$, y su radio, que es una medida determinada.

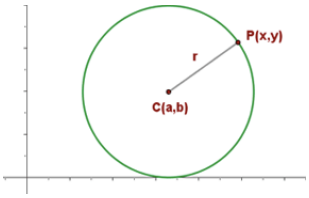


**Figura 5.** Circunferencia con dos puntos a distancia.

### Definición de solución:

La solución para evaluar este problema implica utilizar la fórmula de la distancia euclidiana, es la fórmula más común para calcular la distancia entre dos puntos. Esta fórmula se basa en el teorema de Pitágoras y es válida para puntos en un plano euclidiano.

La distancia se calcula como la raíz cuadrada de la suma de los cuadrados de las diferencias entre las coordenadas $x$ y las coordenadas $y$ de ambos puntos:

$$Distancia = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (5)$$

Por ejemplo, si tenemos dos puntos $A$ (2,3) y $B$ (5,7), podemos usar la fórmula de distancia euclidiana para calcular su distancia:

$$d = \sqrt{((5 - 2)^2 + (7 - 3)^2)} = \sqrt{(3^2 + 4^2)} = \sqrt{(9 + 16)} = \sqrt{25} = 5$$

**Figura 6.** Por lo tanto, la distancia entre Ay B es de 5 unidades

Al final solo comparamos la distancia calculada con el radio $r$ de la circunferencia:

- Si la distancia es mayor que el radio $r$, el punto $T$ está fuera de la circunferencia.
- Si la distancia es igual al radio $r$, el punto $T$ está en el borde de la circunferencia.
- Si la distancia es menor que el radio $r$, el punto $T$ está dentro del área de la circunferencia.

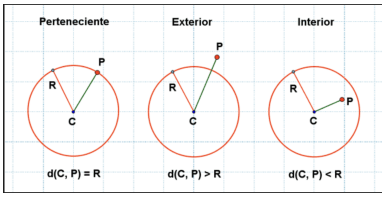


**Figura 7.** Ejemplo de comparación de la distancia y el radio.

### Diseño de la solución:

1. Solicitar al usuario que ingrese las coordenadas del punto $C$ y el punto $T$, es decir $(x_1, y_1)$ y $(x_2, y_2)$ y el radio $r$.
2. Calcular la distancia entre el centro y el punto.
3. Verificar si el punto está dentro del área de la circunferencia
4. Al final mostrar el resultado al usuario

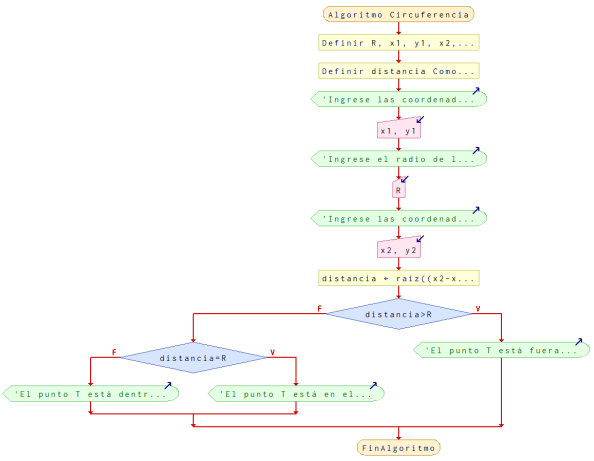


**Figura 8.** Diagrama de flujo del problema 3

### Desarrollo de la solución:

Se importa la clase Scanner para que se pueda leer las entradas del usuario

```
import java.util.Scanner;
```

Posteriormente se define la clase llamada ÇircufrenciaIntegrador"que contendrá el programa.

```
    public class CircuferenciaIntegrador {
    public static void main(String[] args) {}
```

Igualmente se crea una instancia de Scanner para leer la entrada del usuario

```
Scanner datC = new Scanner(System.in);
```

Se solicita al usuario que ingrese las coordenadas del punto C en el formato **"x1, y1"**, las cuales se almacenan en un arreglo de cadenas llamado **"puntoC"**

```
System.out.print("""
                Ingrese las coordenadas del punto C
                seperadas por una coma (x1, y1):
                        """);
    String[] puntoC = (datC.nextLine()).split(",");
```

Igualmente se solicita al usuario que ingrese el *radio* de la circunferencia, se almacena en una variable *r* de tipo float

```
System.out.print("Ingrese el radio de la
     circunferencia: ");
    float r = datC.nextFloat();
    datC.nextLine();
```

Seguidamente se solicita al usuario que ingrese las coordenadas del punto T que es el punto de , en el formato **"x2, y2"** que se almacenan en un arreglo de cadenas llamado **"puntoT"**

```
//Obtener las coordenadas del punto a verificar
System.out.print("""
                Ingrese las coordenadas del punto T
                seperadas por una coma (x2, y2):
                        """);
String[] puntoT = (datC.nextLine()).split(",");
```

Se Cierra el objeto Scanner para liberar los recursos.

```
                datC.close();
```

Se convierte las coordenadas del punto C y del punto T de cadenas a enteros (x1, y1, x2, y2). En Java, convertir datos en cadenas a enteros es útil cuando necesitas manipular o realizar operaciones matemáticas con números enteros que se encuentran representados como *cadenas de caracteres*.

```
int x1 = Integer.parseInt(puntoC[0].trim());
int y1 = Integer.parseInt(puntoC[1].trim());

int x2 = Integer.parseInt(puntoT[0].trim());
int y2 = Integer.parseInt(puntoT[1].trim());
```

En esta parte del desarrollo se calcula la distancia entre el centro (punto C) y el punto T utilizando la fórmula de la distancia euclidiana que significa; la distancia en línea recta entre dos puntos en un plano.

```
float distancia = (float)Math.sqrt(Math.pow(x2 -
     x1, 2) + Math.pow(y2 - y1, 2));
```

Al final se utiliza las condiciones **if, else if y else** para evaluar la relación entre la distancia y el radio de la circunferencia, con esto se verifica si el punto T se encuentra dentro de la circunferencia, en el borde o fuera de ella y se muestra el resultado al usuario.

```
//Verificar si el punto est  dentro del  rea  de
     la circunferencia
     if (distancia > r) {
    System.out.println("El punto T("+x2+","+y2+")
         esta fuera de la circunferencia");
    }else if (distancia == r) {
     System.out.println("El punto T("+x2+","+y2+")
          esta en el borde de la circunferencia");
    }else if (distancia < r){
     System.out.println("El punto T("+x2+","+y2+")
          esta dentro de la circunferencia"); }
```

## Depuración y pruebas:

**Cuadro 2.** Tabla de Corridas del problema 3

| Corrida | C($x_1, y_1$) | T($x_2, y_2$) | R $r$ | Resultado |
|---|---|---|---|---|
| 1 | $(3, 4)$ | $(9, 2)$ | 10 | $T$ esta adentro |
| 2 | $(5, 2)$ | $(8, 1)$ | 12 | $T$ esta adentro |
| 3 | $(34, 23)$ | $(-90, 35)$ | 67 | $T$ esta afuera |
| 4 | $(-27, 3)$ | $(34, -5)$ | 30 | $T$ esta afuera |
| 5 | $(2, 8)$ | $(4, 9)$ | 17 | $T$ esta adentro |

## Resolución Problema 4

Problema:

Dado un numero decimal entero positivo o negativo regresar su equivalente en binario.

### Descripción del problema:

Dado un numero decimal entero positivo o negativo regresar su equivalente en binario.

### Definición de solución:

El informe analiza el proceso de conversión de un número decimal entero positivo o negativo a su representación binaria. Se resalta la relevancia de comprender, en el ámbito de las matemáticas discretas, cómo un número experimenta este cambio de representación.

*A. DECIMALES POSITIVOS*

La conversión de números decimales enteros positivos a binarios, esta basada en divisiones sucesivas por 2.

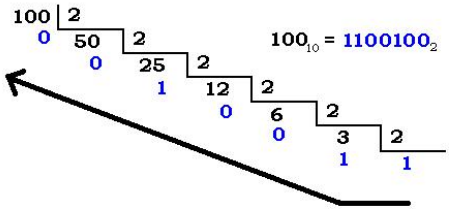


**Figura 9.** Proceso de conversión de 100 utilizando la técnica de sucesivas divisiones por dos

*B. DECIMALES NEGATIVOS*

Por n el caso de números negativos, el proceso se complica y se introduce el concepto de complemento 1 y 2.

- Complemento a 1:

En el contexto del çomplemento a 1"de un número binario, nos referimos a la secuencia de bits que se obtiene al invertir (cambiar de 0 a 1 y de 1 a 0) todos los bits del número original.

Por ejemplo, si tenemos el número binario 01010101, al aplicar el complemento a 1 obtendríamos 10101010, ya que hemos invertido cada bit. El complemento a 1 se utiliza principalmente para representar la magnitud negativa de un número binario y es una parte fundamental en el cálculo del

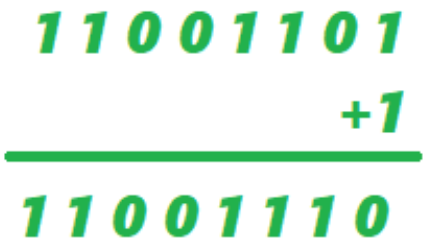


**Figura 10.** Proceso de conversión de 100 utilizando la técnica de sucesivas divisiones por dos

complemento a 2."

- Complemento a 2:

El proceso de obtener el complemento a 2 de un número binario es un paso esencial en la representación de números negativos en sistemas binarios. Este método se basa en la utilización del complemento a 1 y la adición de 1 al resultado

### Diseño de la solución:

- Solicitar al usuario que ingrese un número decimal entero positivo o negativo en el rango de 64 bits, específicamente entre -1024 y 1024.
- Lee la entrada del usuario y almacena el valor en la variable numeroD.
- Verifica si numeroDecimal está en el rango permitido. Si es positivo, realiza la conversión a binario. Si es negativo, calcula los complementos a uno y a dos.
- Si el número está fuera del rango, lanza un mensaje de error.

NÚMEROS ENTEROS POSITIVOS:

- Utiliza un bucle para realizar sucesivas divisiones por 2, registrando los residuos como bits en la representación binaria.

NÚMEROS ENTEROS NEGATIVOS:

- Para complemento a Uno, utiliza la representación binaria obtenida anteriormente.
- Determina la longitud de bits necesarios para representar el número según su rango.
- Añade ceros a la izquierda para alcanzar la longitud necesaria e invierte cada bit para obtener el complemento a uno.

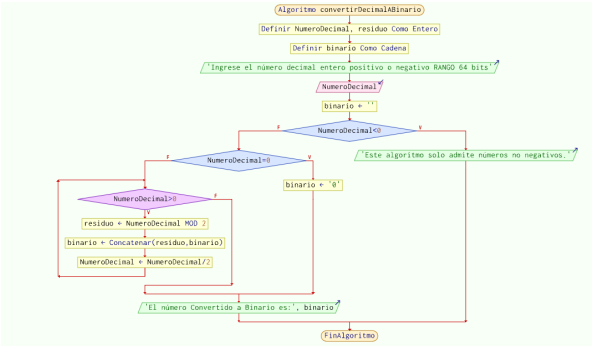


**Figura 11.** Diagrama de flujo en caso de ser números Positivos

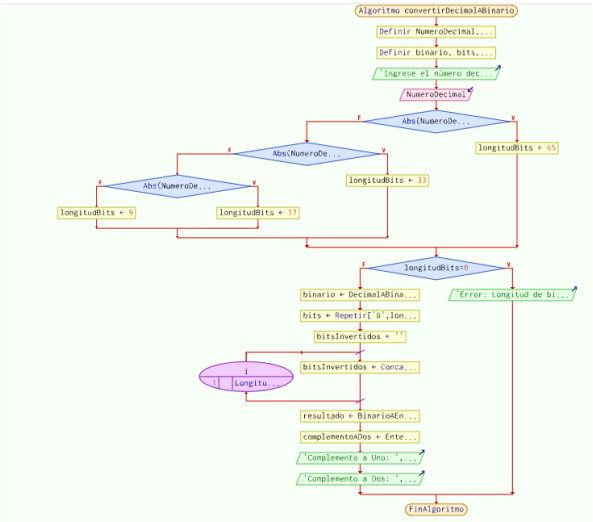


**Figura 12.** Diagrama de Flujo en caso de ser números negtivos

- Para el complemento a dos utiliza la representación del complemento a uno.
- Agrega 1 al dato de número valor resultado para obtener la representación del complemento a dos.
- Utiliza la clase BigInteger para manejar números grandes, ya que la representación de 65 bits podría exceder la capacidad de un tipo de datos primitivo.
- Imprime el resultado de la conversión a binario o los complementos a uno y a dos, según sea el caso.

## Desarrollo de la solución:

El algoritmo de solución para la conversión: comienza importando librerías específicamente "Scanner"para capturar todo lo que ingrese el usuario y "BigInteger".

```
import java.math.BigInteger;
import java.util.Scanner;
```

Se validan los datos: el número debe ser solo entero ya sea positivo o negativo entre el rango de 64 bits, dependiendo su valor va a imprimirlo de acuerdo al número ingresado.

```
    public static void main(String[] args) {
        Scanner decimal = new Scanner(System.in);


        // Solicitud del n mero decimal
        System.out.println("Ingrese el n mero
            decimal entero positivo o negativo
            RANGO 64 bits [-1024, 1024]: ");
        String numeroD = decimal.nextLine();

        // Convierte la variable a un n mero de
            tipo long
        numeroDecimal = Long.parseLong(numeroD);

        // Valida el rango solicitado
        if (numeroDecimal >= -1024 &&
            numeroDecimal <= 1024) {
            if (numeroDecimal >= 0) {
                // Impresi n del n mero
                    convertido en caso de ser
                    positivo
                System.out.println("El n mero
                    convertido a binario es: " +
                    decimalABinario());
            } else {
                //Impresi n del n mero en caso
                    de ser negativo
                System.out.println("Complemento a
                    1: " + complementoAUno());
                System.out.println("Complemento a
                    2: " + complementoADos());
            }

}
```

En caso contrario que el usuario haya tecleado un número diferente al solicitado , donde si se encuentra fuera del rango o sea otro tipo de valor va a arrojar un mensaje de .Error".

```
    //Excepci n en caso de no encontrase en el
        rango solicitado
    } else {
        System.out.println("Error: Formato no
            v lido" );
    }
// Cerrar el scaneo
decimal.close();
```

En el método decimalABinario se crea una variable llamada "binario" tipo cadena, toma el valor de decimal y lo convierte en el valor absoluto, automáticamente si el número ingresado es 0 lo retornara.

```
    public static String decimalABinario() {
        String binario = "";
        // Convetir decimal a valor absoluto
        long numero = Math.abs(numeroDecimal);
```

```
        if (numero == 0) {
            return "0";
        }
```

Se crea un bucle para números distintos de 0, considerando su valor absoluto y permitiendo manejar números negativos y positivos. En el bucle, se divide el número por 2, concatenando su residuo de derecha a izquierda y actualizando su valor.

```
    // Bucle para convertir a n mero decimal
        while (numero != 0) {
            long residuo = numero % 2;
            binario = residuo + binario;
            numero = numero / 2;
        }
        return binario;
    }
```

De acuerdo al número ingresado, se realiza una comparación de su rango con una estructura condicional. Sin embargo, debido a la necesidad de realizar una suma para calcular el complemento a dos, se sumara 1 bit.

```
    public static String complementoAUno() {
        // Obtener la representaci n binaria del
            n mero absoluto
        String binario = decimalABinario();

        // Determinar la longitud de bits
            necesarios
        int longitudBits;
        if (numeroDecimal >= -128 && numeroDecimal
            <= 128) {
            longitudBits = 9;
        } else if ((numeroDecimal >= 129 &&
            numeroDecimal <= 256) ||
            (numeroDecimal >= -256 &&
            numeroDecimal <= -129)) {
            longitudBits = 17;
        } else if ((numeroDecimal >= 257 &&
            numeroDecimal <= 512) ||
            (numeroDecimal >= -512 &&
            numeroDecimal <= -257)) {
            longitudBits = 33;
        } else if ((numeroDecimal >= 513 &&
            numeroDecimal <= 1024) ||
            (numeroDecimal >= -1024 &&
            numeroDecimal <= -513)) {
            longitudBits = 65;
        } else {
            return "Error";
        }
```

Se ajusta la longitud de la cadena con ceros, se invierte para construir el complemento utilizando StringBuilder y se devuelve como una cadena.

```
    // A adir ceros a la izquierda seg n la
        longitud necesaria
        String bits = "0".repeat(Math.max(0,
            longitudBits - binario.length())) +
            binario;

        // A adir ceros a la izquierda seg n la
            longitud necesaria
        String bits = "0".repeat(Math.max(0,
            longitudBits - binario.length())) +
            binario;

        // Invertir cada bit guardandolo en
            complemento
        StringBuilder complemento = new
            StringBuilder();
        for (int i = 0; i < bits.length(); i++) {
            char bit = bits.charAt(i);
            complemento.append((bit == '0') ? '1'
                : '0');
        }
        return complemento.toString();
    }
```

Por último calcula el complemento a dos de un número binario representado como cadena. Obtiene la representación binaria del complemento a uno, convierte a BigInteger (para 65 bits), suma 1 y devuelve la cadena del resultado.

```
    public static String complementoADos() {
        // Obtener la representaci n del
            complemento a uno
        String binario = complementoAUno();

        // Agregar 1 al resultado para obtener la
            representaci n de complemento a 2
        //Declarar el resultado en BigInterger
            para numeros de 65 bits
        BigInteger resultado = new
            BigInteger(binario,
            2).add(BigInteger.ONE);
        return resultado.toString(2);
    }
```

## Depuración y pruebas:

- DECIMAL POSITIVO

| Número Decimal | Número Binario |
|---|---|
| 7 | 111 |
| 10 | 1010 |
| 0 | 0 |

- DECIMAL NEGATIVO

| Decimal | comp. a 1 | comp. a 2 |
|---|---|---|
| -4 | 111111011 | 111111100 |
| -124 | 110000011 | 110000100 |

## Resolución Problema 5

### Problema:

Dado un numero binario de *n* bits regresar su equivalente en decimal.

### Descripción del problema:

El reporte analiza el concepto de ingresar un número de tipo binario con *n* bits, para posteriormente regresar su equivalente en decimal.

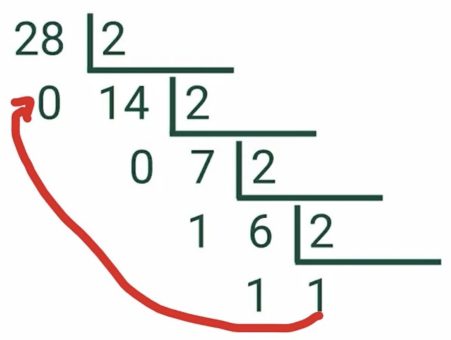


**Figura 13.** Conversión de binario a su equivalente decimal

### Definición de solución:

Los números binarios se conforman con los símbolos 0 y 1, que se combinan para representar cualquier número. Para representar números superiores a 1 dígito, se aplica un método semejante al de formación de números decimales, combinando ordenadamente los símbolos. Se repiten las combinaciones de dígitos anteriores y se le añade un dígito para continuar la numeración

**Cuadro 3.** Valores binarios

| Corrida | Binario |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |
| 11 | 1011 |
| 12 | 1100 |

En el sistema binario se utiliza la ponderación por posición de dígito en la misma forma que en la numeración decimal, por ejemplo:

**Cuadro 4.** Ponderaciones

| Posición | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| ponderacion | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| Peso en decimal | 8 | 4 | 2 | 1 |
| Número en binario | 0 | 1 | 0 | 1 |

El resultado da la operación se obtiene de hacer la operación “res=0*8+1*4+0*2+1*1”

### Diseño de la solución:

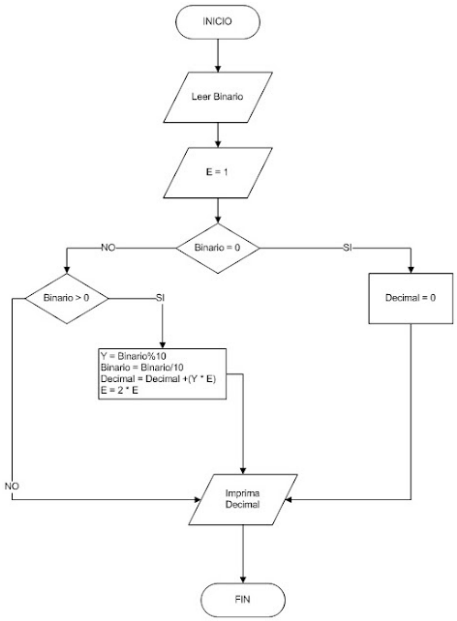


**Figura 14.** Diagrama de flujo de el algoritmo de solución

1. Solicitar al usuario que ingrese el número binario de n bits.
2. Validar que el número binario ingresado sea válido, es decir, que esté compuesto únicamente de 0’s y 1’s y tenga una longitud de n bits.
3. Calcular el número decimal equivalente utilizando el método binAdecimal.
4. Mostrar el resultado al usuario.

### Desarrollo de la solución:

El algoritmo de solución del problema comienza solicitando al usuario teclee el número binario a convertir, para posteriormente almacenarlo en la variable *nbinario*.

```
Scanner in = new Scanner(System.in);

    System.out.println("ALGORITMO PARA CONVERTIR
        UN N MERO BINARIO A DECIMAL");
```

```
    System.out.println("------");
    System.out.println("Ingresa el n mero binario
         correspondiente: ");
    String nbinario = in.nextLine();
    in.close();
```

Se inicializa un *if*, esto con la finalidad de que si se llega a ingresar el signo ", la conversión lo elimine.

```
if (nbinario.contains("-")) {
    nbinario = nbinario.replace("-", "");
    }
```

Posteriormente, se vuelve a inicializar un *if*, este para realizar la operación de conversión únicamente si el dato de entrada contiene 0 y 1, de lo contrario, cancelar la conversión arrojando un mensaje de error.

```
if (nbinario.matches("[01]+")) {
    int num = binAdecimal(nbinario);
    System.out.println("N mero decimal: " + num);
    }else{
        System.out.println("Car cter de tipo
             inv lido");
    }
```

Si la condición anterior se cumple, se manda a llamar al método *binAdecimal*, que es el que contiene todo el proceso de conversión.

```
public static int binAdecimal(String binario) {
    int n = binario.length();
    int decimal = 0;

    for (int i = 0; i < n; i++) {
        int bit =
        Character.getNumericValue(binario.charAt(i));
    decimal += bit * Math.pow(2, n - 1 - i);
    }

    return decimal;
}
```

## Depuración y pruebas:

**Cuadro 5.** Tabla de Corridas

| Corrida | Binario | Decimal |
|---|---|---|
| 1 | 010111 | 23 |
| 2 | 011011 | 27 |
| 3 | 101010 | 42 |
| 4 | 010101 | 21 |
| 5 | 111011 | 59 |

## Resolución Problema 6

### Problema:

Dada una tabla de verdad de *n* bits generar la expresión booleana que genere de manera fidedigna las salidas de esta tabla.

### Descripción del problema:

Se dispone de una tabla que exhibe múltiples combinaciones de unos y ceros que se forman debido a la naturaleza de la representación de las entradas binarias, junto con los resultados deseados asociados a cada combinación.
La tarea consiste en hallar una expresión booleana que describa de manera lógica estas relaciones.

### Definición de solución:

Para resolver este problema, se adopta la técnica conocida como "Suma de Productos"(Sum-Of-Products, o SOP en inglés). Esta metodología implica identificar las combinaciones específicas para las cuales la salida es igual a 1. Posteriormente, estas condiciones se combinan utilizando operaciones lógicas AND, incluyendo la posibilidad de la operación NOT para algunas variables.
Estos conjuntos de condiciones AND, que pueden incluir negaciones (NOT), se consolidan mediante la operación lógica OR. El resultado final es una expresión booleana que representa de manera clara y sencilla la lógica detrás de la tabla de verdad.

**Solo se toman en cuenta las salidas que son 1 para formar la suma de productos**

| A | B | C | Salida | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | ABC |
| 1 | 1 | 0 | 0 | |
| 1 | 0 | 1 | 1 | A¬BC |
| 1 | 0 | 0 | 0 | |
| 0 | 1 | 1 | 0 | |
| 0 | 1 | 0 | 1 | ¬AB¬C |
| 0 | 0 | 1 | 0 | |
| 0 | 0 | 0 | 1 | ¬A¬B¬C |

**ABC + A¬BC + ¬AB¬C + ¬A¬B¬C**

**Figura 15.** Suma de productos para una tabla de verdad

La siguiente notación se relaciona directamente con la técnica de la Suma de Productos (SOP) mencionada previamente.

$$F(A, B, C, ...) = \Sigma(m_i) \quad (6)$$

Donde:

- $\Sigma(m_i)$*:* Indica que se está sumando sobre los términos $m_i$, donde *i* representa los índices que van desde el valor inicial hasta el valor final.
- $m_i$*:* Cada término $m_i$ es un producto que representa una combinación específica de las variables $A, B, C, ...$ y sus negaciones.

Se centra únicamente en las salidas con valor 1 porque estas representan los casos en los que la función booleana es verdadera o activa, si la salida es 0 para una combinación particular de entradas, ese caso no contribuirá al término de suma de productos

### Diseño de la Solución:

- *Entrada de Datos:* Se solicita la cantidad de bits (*n*) y las posiciones de salidas activas.
- *Generación de la Tabla de Verdad:* Se crea la tabla de verdad con todas las combinaciones posibles de*n* bits.
- *Identificación de Combinaciones Activas:* Se determinan las combinaciones de entradas que resultan en salidas activas (1).
- *Generación de Términos de Producto:* Para cada combinación activa, se crea un producto con la multiplicación binaria (AND).
- *Construcción de la Expresión Booleana:* Los producto se combinan con la suma de todos estos (OR).
- *Presentación de la Expresión:* Se muestra la expresión booleana final.

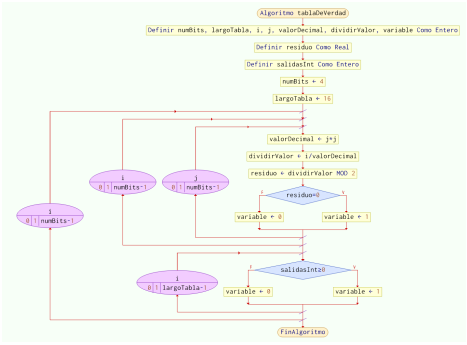

**Figura 16.** Llenar una tabla de verdad

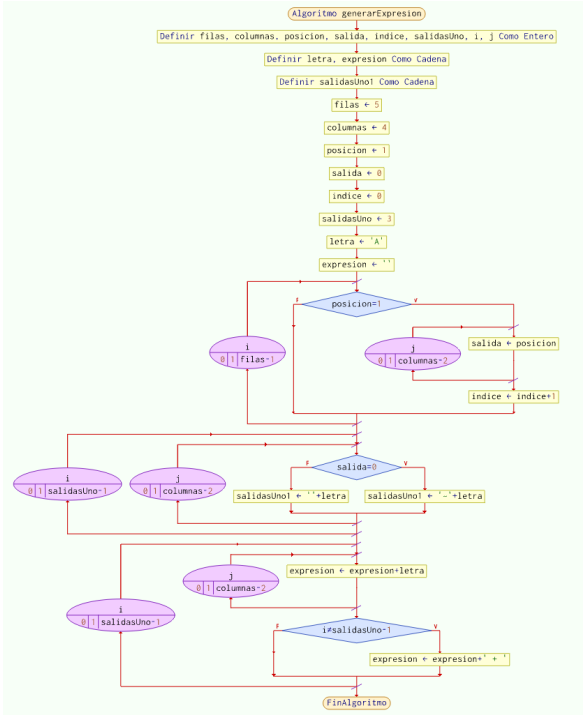


**Figura 17.** Generar suma de productos para una tabla de verdad

## Desarrollo de la solución:

Se solicita al usuario la cantidad de bits y las posiciones de las salidas activas, con esto de calcula la cantidad de columnas y filas para la tabla desde el método(`main`).

```
Scanner entradaDatos = new Scanner(System.in);
    //Entrada de datos cantidad de bits
    System.out.print("Ingrese la cantidad de bits:
        ");
    int numBits = entradaDatos.nextInt();
    entradaDatos.nextLine();

    //Entrada de salidas con valor 1
    int numSalidas = (int)Math.pow(2, numBits);
    int[][] tabla = new int[numSalidas][numBits+1];
    System.out.print("Ingrese las salidas que son
        1 sepadados por comas
        ("+1+"-"+numSalidas+"): ");
    String[] salidas =
        entradaDatos.nextLine().split(",");
    entradaDatos.close();//cerrar captura de datos
```

La función `llenarTabla` llena una matriz con todas las combinaciones de bits y asigna los valores de salida correspondientes y retorna la matriz que representa la tabla con los valores corrspondientes.

```
    public static int[][] llenarTabla(int[][]
        tabla, int numBits, String[] salidas) {
    //...codigo restante
    return tabla;
    }
```

› Convierte las posiciones de salida activas en un arreglo tipo entero para su fácil manejo.

```
    int[] salidasInt = new
         int[salidas.length];
    for (int i = 0; i < salidasInt.length;
         i++) {
        salidasInt[i] =
            Integer.parseInt(salidas[i])-1;
    }
    Arrays.sort(salidasInt);
```

› Utiliza bucles `for` para asignar todas las combinaciones de bits y asigna las salidas activas haciendo uso de condicionales.

```
    /Llenar las combinaciones
    for (int i = 0; i < tabla.length; i++) {
        for (int j = 0; j < numBits; j++) {
            int division = i / ((int)
                Math.pow(2, j));
            int valorPosicion = division % 2;
            tabla[i][(numBits)-1-j] =
                (valorPosicion == 0)   ?
                1 : 0 ;
        }
    }
    // LLenar las salidas de la tabla
    for (int i = 0; i < tabla.length; i++) {
        if (Arrays.binarySearch(salidasInt,
             i) >= 0) {
            tabla[i][numBits] = 1;
        } else {
            tabla[i][numBits] = 0;
        }
    }
```

Las combinaciones activas (salida igual a 1) se identifican y se almacenan en una matriz auxiliar.

```
//Obtener las filas con salidas de valor 1
String[][] salidasUno = new
     String[salidas.length][filas-1];
int indice = 0;
for (int i = 0; i < filas; i++) {
    if (tablaDeVerdad[i][columnas-1] == 1) {
        for (int j = 0; j < columnas-1; j++) {
            salidasUno[indice][j] =
            Integer.toString(tablaDeVerdad[i][j]);
        }
        indice++;
    }
}
```

Se asigna la letra correspondiente y se crea un producto para cada combinación activa utilizando la multiplicación binaria.

```
//Cambiar valores por la letra correspondiente
    for (String[] salidasUno1 : salidasUno) {
        for (int j = 0; j < columnas-1; j++) {
            if ("0".equals(salidasUno1[j])) {
                salidasUno1[j] =
                    "-"+preposiciones.charAt(j);
            } else {
                salidasUno1[j] =
                    ""+preposiciones.charAt(j);
            }
        }
    }
```

Se construye la expresión booleana combinando los términos de producto con la suma de todos ellos.

```
//Concatear arreglo en una cadena
for (int i = 0; i < salidasUno.length; i++) {
    for (int j = 0; j < columnas-1; j++) {
        expresion += salidasUno[i][j];
    }
    if (i != salidasUno.length-1) {
        expresion += " + ";
    }
}
```

La función `imprimirTabla` imprime la tabla de verdad en la consola con colores para una mejor visualización iterando por cada encabezado y elementos de la matriz.

```
//Datos
    int filas = tablaDeVerdad.length;
    int columnas = tablaDeVerdad[1].length;
    String preposiciones=
         "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    String[] colores = {
        "\u001B[31m", // rojo
        "\u001B[34m", // azul
        "\u001B[32m", // verde
        "\u001B[35m", // morado
        "\u001B[33m", // amarillo
        "\u001B[36m", // cian
    };
```

› Se itera por los encabezados correspondientes

```
    //Imprimir la tabla
    System.out.println("\nLa tabla de verdad
         es:");
    // Encabezados
    System.out.print("     ");
    for (int i = 0; i <= columnas-1; i++) {
        if (i == columnas-1) {
            System.out.print(colores[i] +
                 "Salida");
        }else{
            System.out.print(colores[i] +
                 preposiciones.charAt(i) + "
                  ");
        }
    }
    System.out.println();
```

› Se itera por los el contenido de la tabla

```
    //Combinaciones de la tabla
    for (int i = 0; i < filas; i++) {
        if (i<9) {
            System.out.print(" ");
        }
        System.out.print(i+1+"| ");
        for (int j = 0; j <= columnas-1;
             j++) {
            System.out.print(" " +colores[j]
                 + tablaDeVerdad[i][j] + "
                 ");
        }
        System.out.println();
    }
```

En el método principal (`main`) se genera la tabla de verdad, la imprime y muestra la expresión booleana resultante.

```
public static void main(String[] args) {
    //resto del codigo
    /*PROCESOS*/
    tabla = llenarTabla(tabla,
         numBits,salidas);//Llenar tabla

    imprimirTabla(tabla);//imprimimos la tabla
         generada

    System.out.println("\nExpresion booleana: ");
    String expresion =
         generarExpresionBooleana(salidas,
         tabla);//Generar expresion booleana
    System.out.println(expresion+"\n");//Mostrar
         la expresion
}
```

## Depuración y pruebas:

| Nº | A | B | C | Salida |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 1 | 0 | 1 | 0 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 1 | 0 |
| 6 | 0 | 1 | 0 | 1 |
| 7 | 0 | 0 | 1 | 0 |
| 8 | 0 | 0 | 0 | 0 |

**Cuadro 6.** Tabla de verdad 1

Expresión booleana: $ABC + A'BC'$

| Nº | A | B | Salida |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 2 | 1 | 0 | 1 |
| 3 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 |

**Cuadro 7.** Tabla de verdad 2

Expresión booleana: $AB + AB'$

## CONCLUSION

En este proyecto, hemos alcanzado con éxito la implementación de un software en Java para abordar problemas de ingeniería, destacando la eficacia de la metodología estructurada de seis etapas (6D). La eficiencia de este enfoque se evidencia en los resultados obtenidos, los cuales no solo proporcionan soluciones prácticas y claras, sino que también resaltan la versatilidad del software al integrar conceptos de álgebra, cálculo y lógica booleana. Más allá de la creación del software, la verdadera trascendencia de este proyecto radica en el proceso mismo de construir soluciones mediante la programación. La sinergia entre conceptos de geometría, álgebra, cálculo y matemáticas discretas no solo ha sido evidente en las soluciones propuestas, sino que también destaca la importancia continua en la resolución de problemas complejos en la ingeniería moderna. Este proyecto no solo abre nuevas posibilidades para futuras investigaciones en la intersección de programación y disciplinas matemáticas, sino que también refuerza la idea de que el método de resolución es tan valioso como el resultado mismo.

## AGRADECIMIENTOS

Queremos expresar nuestro sincero agradecimiento a aquellos que contribuyeron de manera significativa al desarrollo de nuestro proyecto integrador. En particular, deseamos reconocer y agradecer a los siguientes docentes por su invaluable apoyo y orientación: Agradecemos al profesor Neri Pérez Giovany Humberto por su dedicación y asesoramiento en el aprendizaje de LaTeX, así como por su labor en la corrección de la codificación y la explicación detallada de los problemas de cálculo que surgieron durante el desarrollo de nuestro proyecto. Extendemos nuestro agradecimiento al profesor Agustín Soto Arista, cuya colaboración fue fundamental en las pruebas de los problemas relacionados con Matemáticas discretas. Su expertise y disposición para ayudarnos fueron esenciales para mejorar la calidad de nuestro trabajo. Además, agradecemos a la profesora Eunice Santiago Manzano por su valiosa asistencia en los ensayos de exposición y su dedicación en la depuración de problemas. Su orientación y perspectiva fueron esenciales para mejorar nuestra presentación y abordar eficientemente los problemas que encontramos. Su guía y enseñanzas han dejado una huella significativa en nuestro aprendizaje y desarrollo académico, muchas gracias por su tiempo, paciencia y dedicación.

## REFERENCES

1. Cunoticias. (s. f.). Complemento a 1 (uno) con ejemplos. https://www.cunoticias.com/internet/complemento-a-1-uno-con-ejemplos.php
2. Clases particulares y Profesores particulares.(2023, May 10).(Fórmulas para calcular la distancia entre dos puntos). [Online]. Available: https://www.tusclasesparticulares.com/blog/como-calcular-distancia-entre-dos-puntos (URL)
3. Johnsonbaugh, R. (s. f.). MATEMÁTICAS DISCRETAS. Pearson Educación.
4. Fernández, M. Y. A., Hernández, Y. J. S., Montaña, M. M. (s. f.). Matemáticas Discretas:: Con un enfoque desde la ingeniería y ciencias sociales - Conceptos básicos. Editorial de la Universidad Pedagógica y Tecnológica de Colombia - UPTC.
5. Sección 5: Soto, J. A. - GEEKNETIC. (Cómo convertir binario en decimal paso a paso). [Online]. Available: https://www.geeknetic.es/Guia/2667/Como-convertir-binario-en-decimal-paso-a-paso.html (URL)
6. Sección 5: Escobar, B., Perfil, V. T. mi. - Blogspot.com.. (Algoritmo para pasar de Binario a Decimal.). [Online]. Available: http://mensajeseintereses.blogspot.com/2011/10/2-algoritmo-para-pasar-de-binario.html (URL)
7. All About Circuits. (2023). Conversión de tablas de verdad en expresiones booleanas. En Libro de texto de electrónica. https://www.allaboutcircuits.com/textbook/digital/chpt-7/converting-truth-tables-boolean-expressions/
8. Xiang, Y., Dalchau, N., Wang, B. (2018). Scaling up genetic circuit design for cellular computing: advances and prospects. Natural computing, 17(4), 833-853.
9. Cunoticias. (s. f.). Complemento a 1 (uno) con ejemplos. https://www.cunoticias.com/internet/complemento-a-1-uno-con-ejemplos.php
10. /u/school¡sa.(2017, *septiembre*29). *FORMLAGENERALPARAECUACIONESCUADRTICAS. GeoGebra.https* : *//www.geogebra.org/m/GYXrzYEF*

**Karen Perez Ortiz** Es una estudiante de la Ingeniería en Tecnologías de la Información y Comunicaciones, sus intereses son las series animadas, la astronomía y ver streamers. Nació en Mixquiahuala de Juarez, aunque desconocía las ramas de la tecnología no fue desde la secundaria que a partir de una convención de tecnología fue que se intereso en estudiar en las tecnologias. La forma en que a partir de una computadora se pueden crear diversas cosas se volvio fasivanate para ella, Sus metas no solo es concluir la universidad si no que además quiere dejar algo significativo que

beneficie a las personas que siempre la apoyaron, sus principales intereses son desarrollo web en Experiencia de Usuario, así Inteligencia Artificial y Aprendizaje Automático la forma en que le explicaron sobre este tema la animo a estudiar mas y mas de ello .Visita mi página personal en Github.

**Edwin Salvador Soto Hernández** Edwin Salvador Soto Hernández, de 18 años y originario del estado de Querétaro, se encuentra inmerso en la ingeniería en Tecnologías de la Información y Comunicaciones en el ITSOEH, ubicado en Tlahuelilpan, Hidalgo. Desde una edad temprana, demostró un ferviente interés por la tecnología e informática, y ahora, tras dos años en este estado, ha consolidado sus estudios de nivel medio superior. Su exploración en diversas ramas tecnológicas abarca el desarrollo web, desarrollo móvil y UX/UI, mientras que sus principales intereses personales incluyen el modelado 3D, disfrutar de series animadas y videojuegos. A un corto plazo, aspira a concluir la universidad y especializarse en áreas que le apasionan, como ciberseguridad, redes y desarrollo de software. A largo plazo, su ambición es colaborar en proyectos grandes que tengan relevancia en el futuro, contribuyendo así al avance tecnológico y tener relevancia en el campo. Visita mi página personal en GitHub.

**José María Ortiz Escamilla** Originario de Mixquiahuala de Juárez, Hgo. Es un estudiante de Ingeniería en Tecnologías de la Información y Comunicaciones. El cual desde una edad temprana mostró curiosidad por las tecnologías y generando un grande interés por ellas desde su educación de nivel medio superior al estudiar la especialidad de programación, muestra una fascinación por la música y el anime de todo tipo, le apasiona ir al gimnasio, ya que mejora el cómo se siente con el mismo y lo hace sentir capaz de lograr lo que se proponga, lo que desea conseguir a largo plazo desea poder trabajar en empresas que tengan una gran relevancia en el área de la ciberseguridad o el desarrollo de software teniendo la oportunidad de dejar huella en el área de la informática.Visita mi página personal en GitHub.

**Bernal Franco Lizbeth de Jesús** es una estudiante que cursa el primer semestre de la ingeniería en Tecnologías de la Información y Comunicaciones con muchas pasiones y metas personales que la impulsan a seguir. Originaria de Mixquiahula de Juarez, desde el segundo año de bachirerato ella desarrollo un interés hacia las redes, desarrollo web y ciberseguridad, esto fue un impulso para que ella viera la posibilidad de estudiar o convertirse en una especialista en la tecnologia, su gran meta es terminar su carrera y convertirse en toda una profesional y trabajar en proyectos a un mejor crear un proyecto. Visita mi página personal en GitHub.

**Ángeles Martínez Dilan Emir** Es un estudiante originario de el municipio de Mixquiahuala de Juárez, en el estado de Hidalgo. Es estudiante de la carrera de Ingeniería en Tecnologías de la Información y Comunicaciones, al cual le apasionan algunas cosas, como perder el tiempo en TikTok, comer, destapar sus computadoras para darles su mantenimiento y comer, pero sobre todo le llama la atención la programacion aunque no sea bueno, su sueño es poder titularse como Ingeniería en Tecnologías de la Información y Comunicaciones y trabajar en la empresa de Huawei, es una persona muy simpatica aunque cuando lo hacen enojar es mejor esperar a que se le pase .Visita mi página personal en GitHub.