FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

ACTIVIDAD

CÁLCULO DE LA COMPLEJIDAD CICLOMÁTICA Y CASOS DE USO

Jemmy Puzma	6627
Juan Carrera	6605
David Llumitaxi	6700
Víctor Montaluisa	6563

Ing. Raúl Rosero

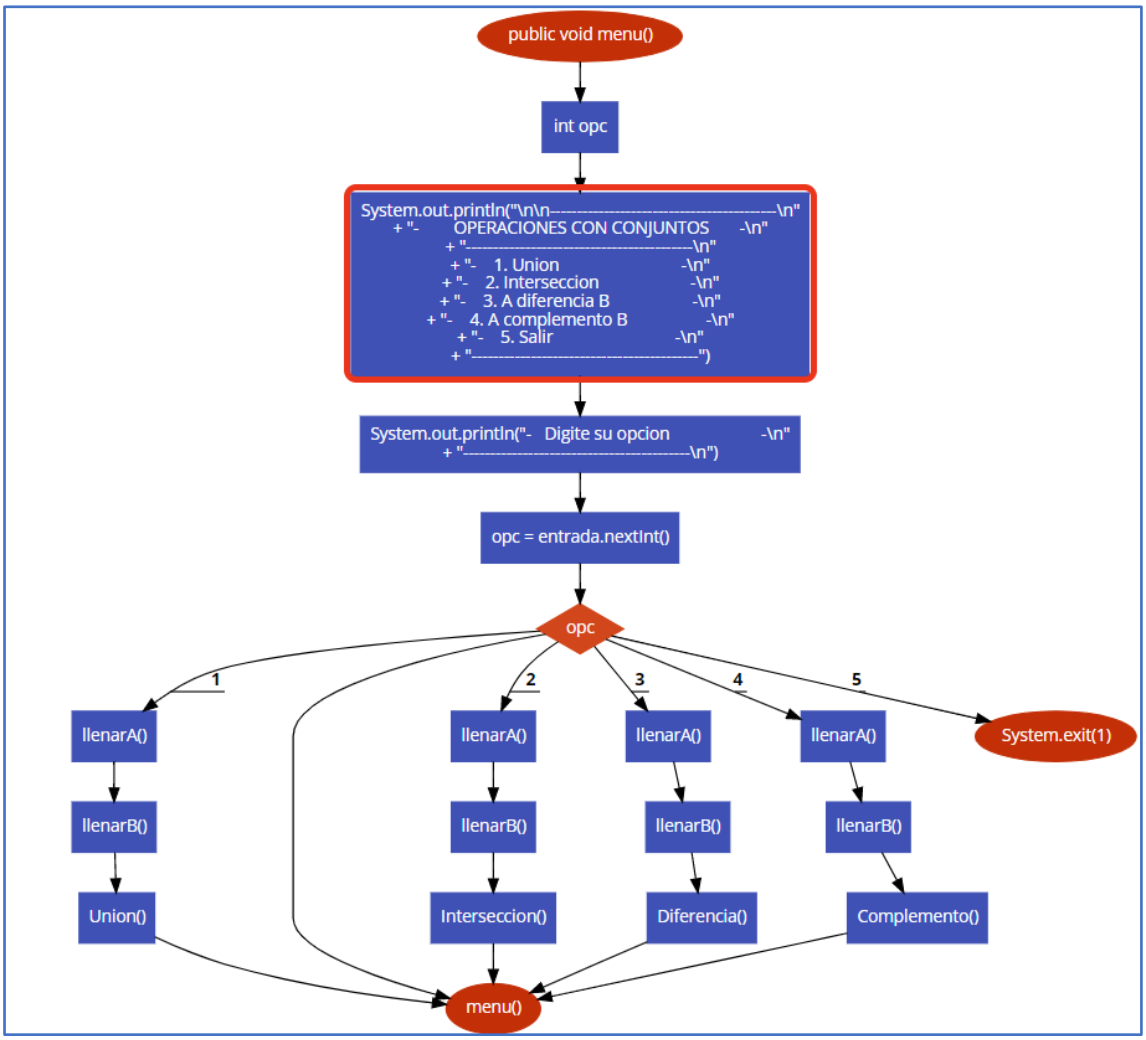
Octavo Semestre
Software

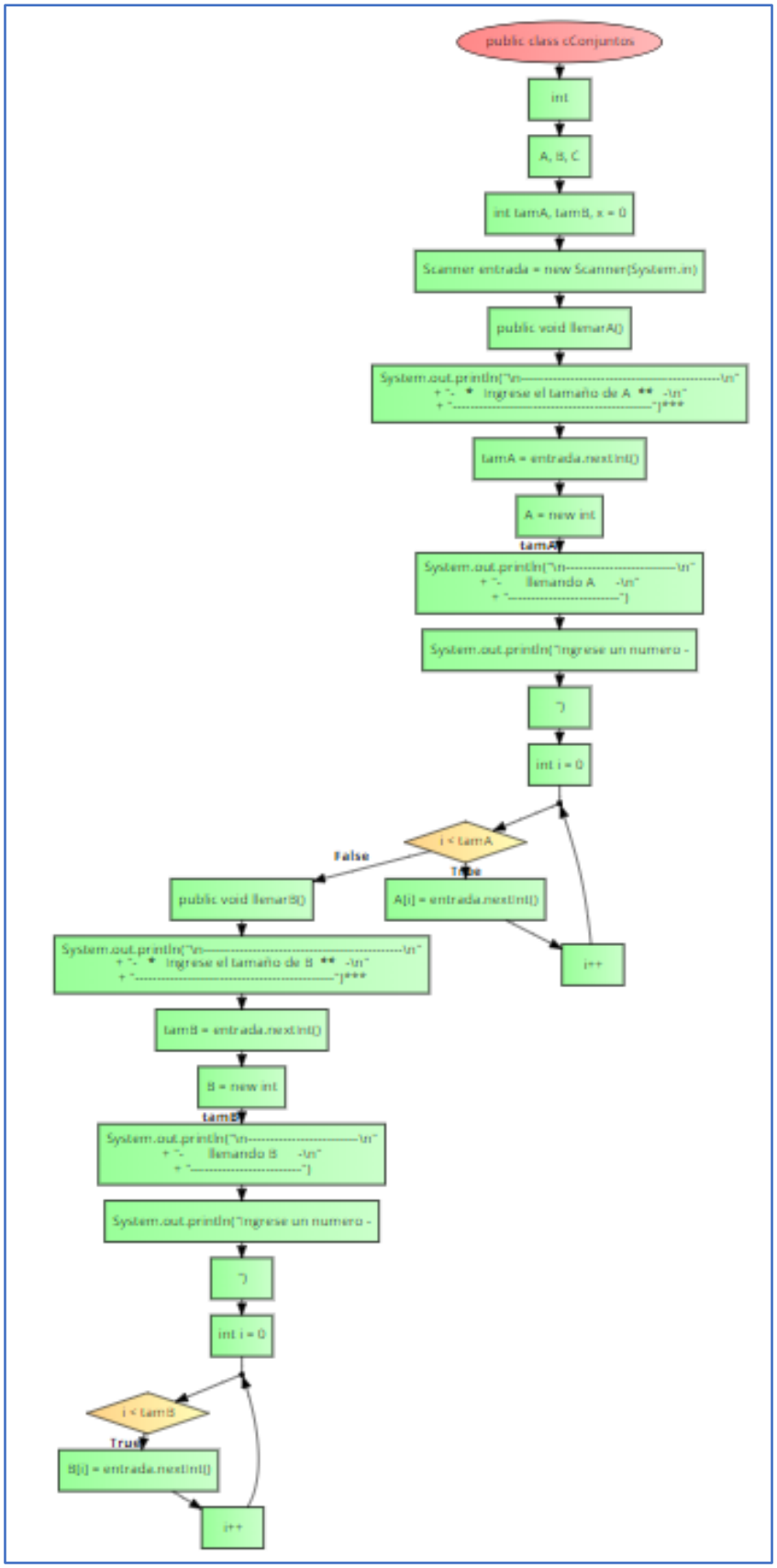
VALIDACIÓN Y VERIFICACIÓN DEL SOFTWARE

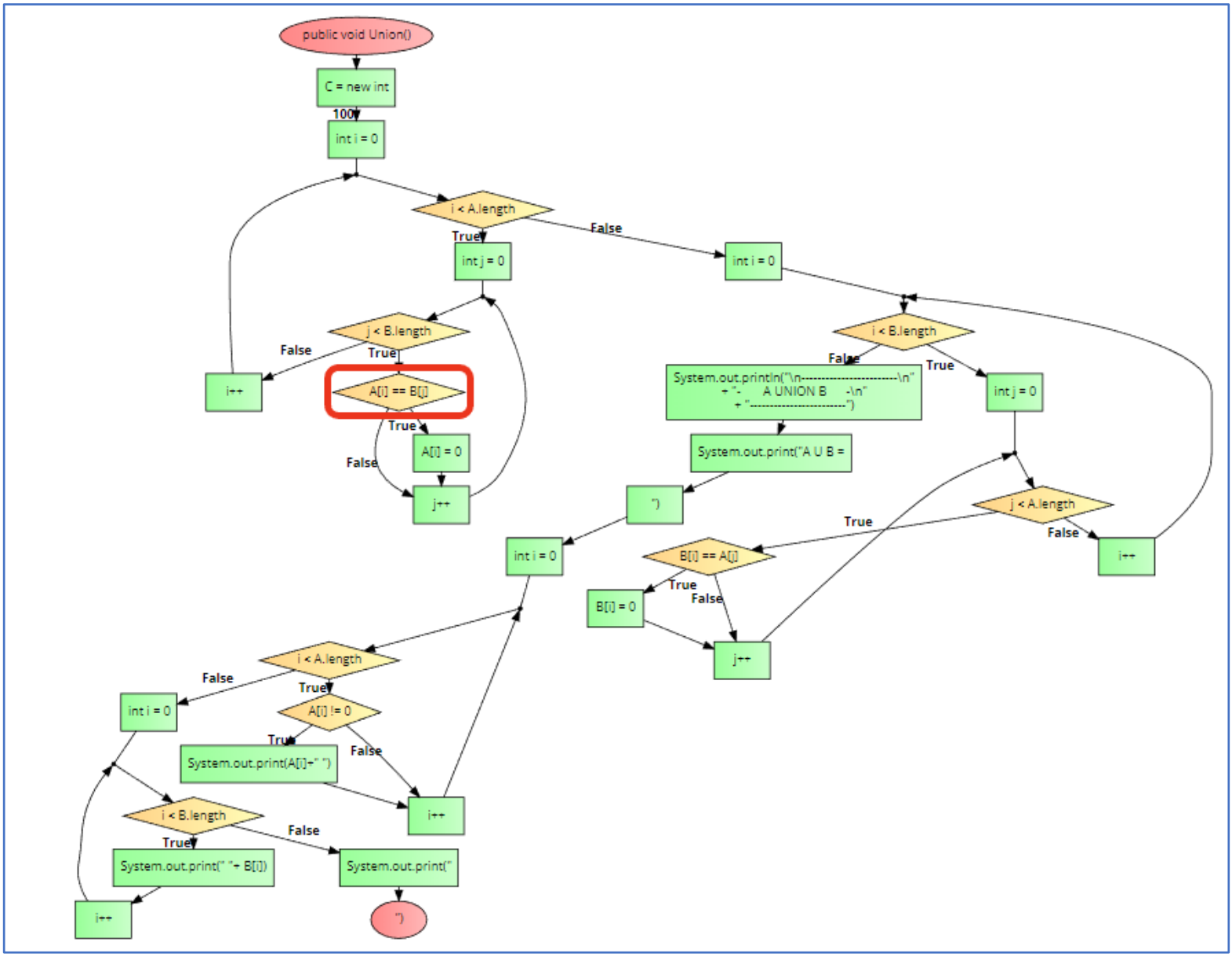
2022

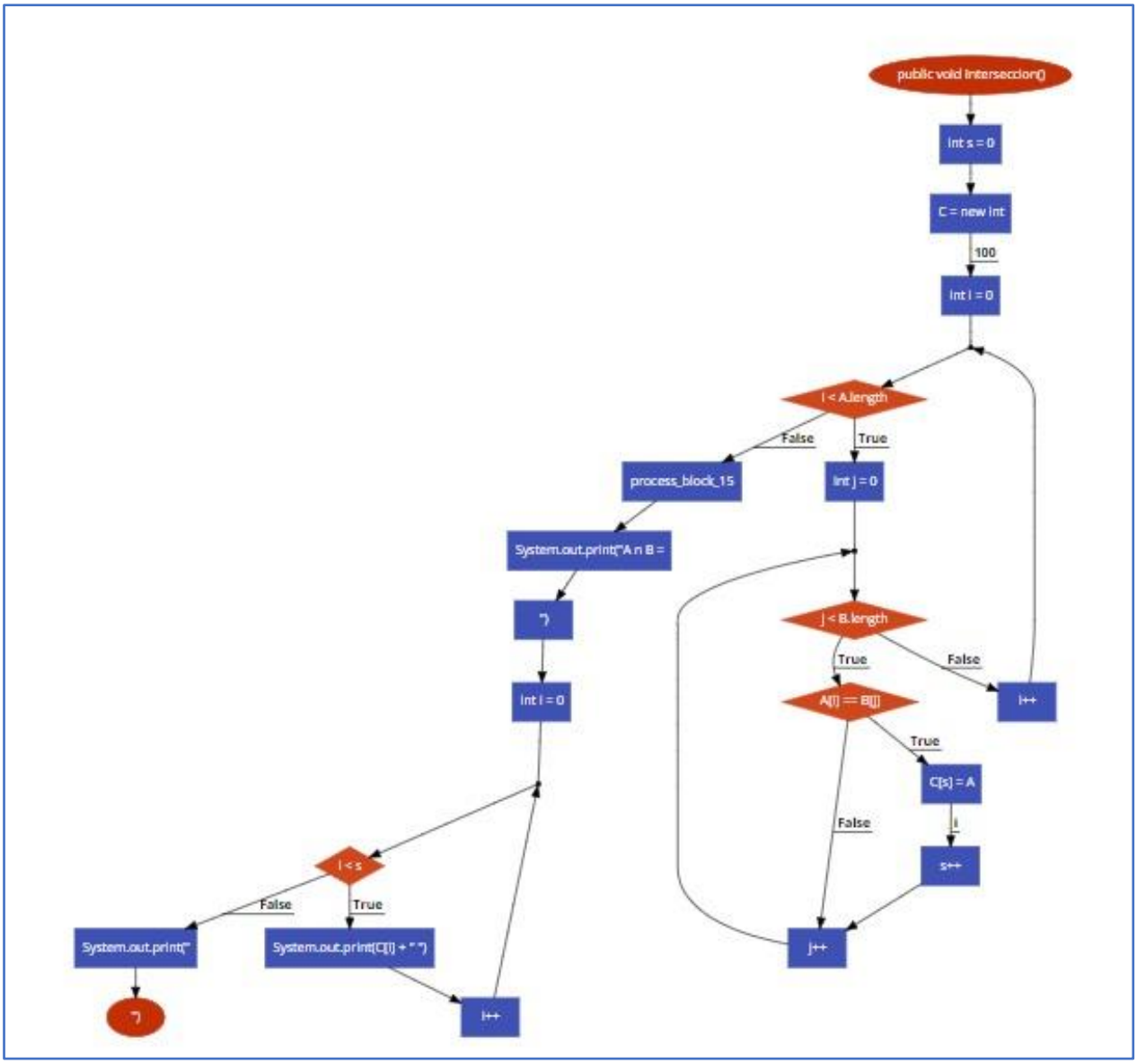
CALCULO LA COMPLEJIDAD CICLOMÁTICA

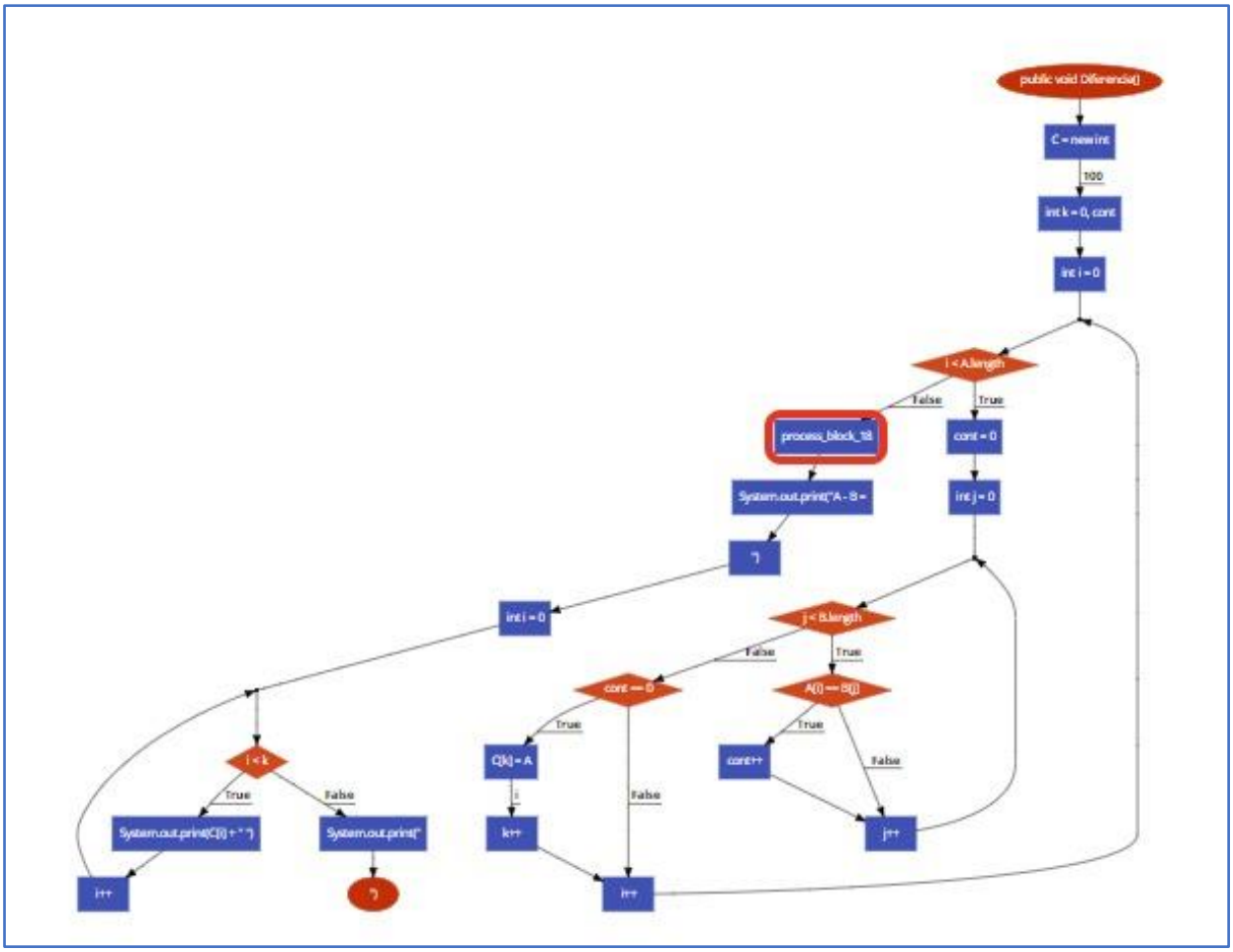
DIAGRAMAS DE FLUJO

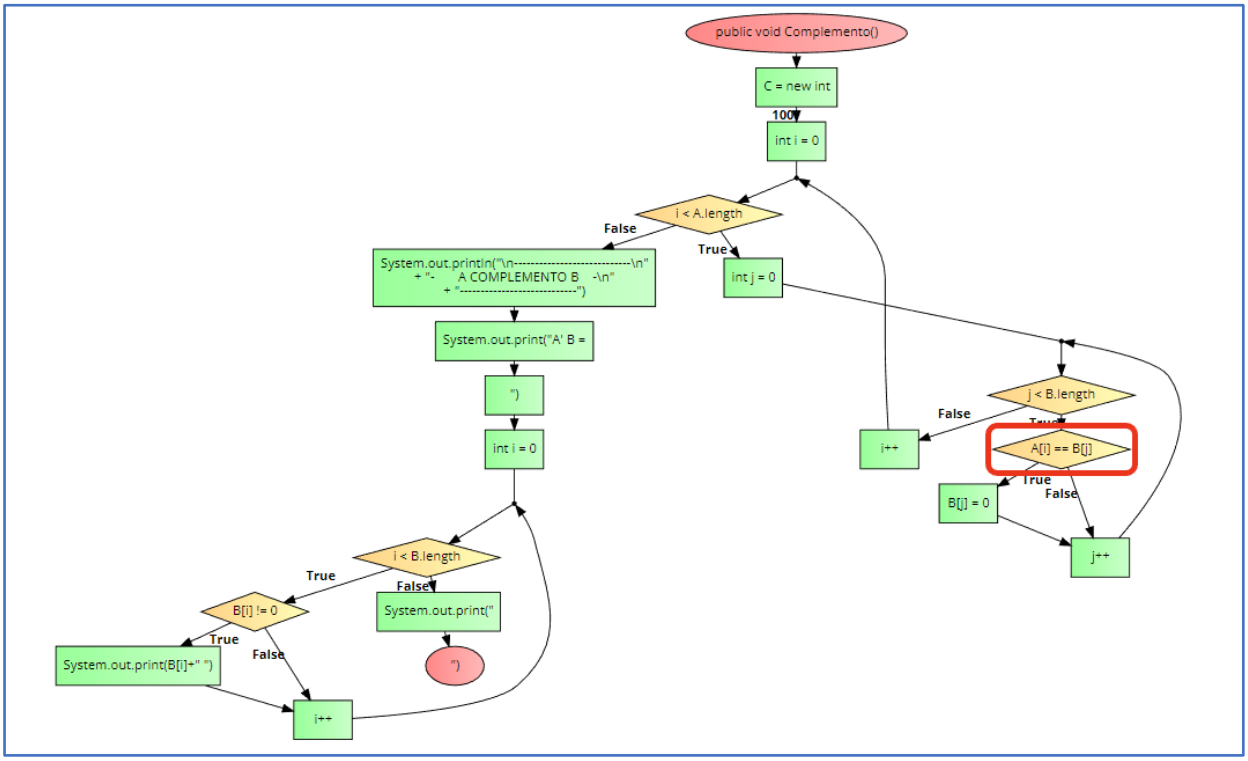



public class cConjuntos
int
A, B, C
int tamA, tamB, x = 0
Scanner entrada = new Scanner(System.in)
public void llenarA()
System.out.println("\n----------\n"
+ "- * Ingrese el tamaño de A ** -\n"
+ "----------"]***
tamA = entrada.nextInt()
A = new int
tamA
System.out.println("\n----------\n"
+ "- llenando A -\n"
+ "----------"]
System.out.println("Ingrese un numero -
")
int i = 0
i < tamA
False
True
public void llenarB()
A[i] = entrada.nextInt()
i++
System.out.println("\n----------\n"
+ "- * Ingrese el tamaño de B ** -\n"
+ "----------"]***
tamB = entrada.nextInt()
B = new int
tamB
System.out.println("\n----------\n"
+ "- llenando B -\n"
+ "----------"]
System.out.println("Ingrese un numero -
")
int i = 0
i < tamB
True
B[i] = entrada.nextInt()
i++


public void Union()
C = new int
100
int i = 0
i < A.length
True
False
int j = 0
j < B.length
False
True
i++
A[i] == B[j]
True
A[i] = 0
False
j++
int i = 0
i < B.length
False
True
System.out.println("\n----------\n"
+ "- A UNION B -\n"
+ "----------")
int j = 0
System.out.print("A U B =
")
j < A.length
True
False
i++
int i = 0
B[i] == A[j]
True
False
B[i] = 0
j++
i < A.length
False
True
int i = 0
A[i] != 0
True
False
System.out.print(A[i]+" ")
i++
i < B.length
True
False
System.out.print(" "+ B[i])
System.out.print("
")
i++


public void Interseccion()
int s = 0
C = new int
100
int i = 0
i < A.length
False
True
process_block_15
int j = 0
System.out.print("A n B =
"
int i = 0
j < B.length
True
False
A[i] == B[j]
True
i++
C[s] = A
i
s++
False
j++
i < s
False
True
System.out.print("
System.out.print(C[i] + " ")
i++
"


public void Diferencia()
C = new int
100
int k = 0, cont
int i = 0
i < A.length
False
True
process_block_till
cont = 0
System.out.print("A - B =
int j = 0
")
int i = 0
j < B.length
False
True
cont == 0
A[i] == B[j]
True
True
i < k
C[k] = A
cont++
True
False
False
False
System.out.print(C[i] + " ")
System.out.print("
k++
j++
i++
i++
")


public void Complemento()
C = new int
100
int i = 0
i < A.length
False
True
System.out.println("\n----------\n"
+ "- A COMPLEMENTO B -\n"
+ "----------")
int j = 0
System.out.print("A' B =
")
int i = 0
j < B.length
False
i++
A[i] == B[j]
True
False
B[j] = 0
j++
i < B.length
True
False
B[i] != 0
System.out.print("
True
False
System.out.print(B[i]+" ")
")
i++

CÁLCULO DE LA COMPLEJIDAD

Para calcularla se utilizó la siguiente fórmula:

Restar las **aristas** menos los **nodos** y sumar 2:

V(G) = Aristas – Nodos + 2

- Intersección

  Aristas= 25

  Nodos= 21
- Llenar

  Aristas= 27

  Nodos= 26
- Unión

  Aristas= 42

  Nodos= 33
- Diferencia

  Aristas= 29

  Nodos= 23
- Complemento

  Aristas= 24

  Nodos= 20

Aristas Totales = 147

Nodos Totales= 123

V(G)= 147-123+2= 26 (complejidad ciclomatica)

Complejidad ciclomática	**Tipo de código**
1-10	Simple
11-20	Algo complejo
21-50	Complejo
50	No testeable

Fórmula para casos de prueba

E= número de aristas

N= número de nodos

P=nodos de salida

Formula M=E-N+2P

M=147-123+2(5) = **34 (casos de prueba)**

CASOS DE USO

Descripción del problema

Desarrollo de un software con la capacidad de realizar operaciones de conjuntos (unión, intersección, diferencia y complemento), donde se deberá ingresar valores por parte del usuario.

Requisitos

Identificación del requerimiento	RQ1
Nombre del requerimiento	Ingreso de conjuntos
Funcionalidades	• Ingresar dos conjuntos • Verificar que los datos ingresados no excedan de la dimensión del vector • Verificar que los datos ingresados sean enteros
Descripción del requerimiento	El sistema permitirá el ingreso de los datos de dos conjuntos

Identificación del requerimiento	RQ2
Nombre del requerimiento	Ingreso de elementos
Funcionalidades	• Ingresar datos
Descripción del requerimiento	El sistema permitirá el ingreso de números enteros en cada conjunto

Identificación del requerimiento	RQ3
Nombre del requerimiento	Ingreso de tamaño de conjuntos o vectores
Funcionalidades	• Ingresar dimensión
Descripción del requerimiento	El sistema permitirá el ingreso de una dimensión en cada conjunto.

Identificación del requerimiento	RQ4
Nombre del requerimiento	Comparar los vectores
Funcionalidades	• Realiza una comparación y guarda en otro vector.
Descripción del requerimiento	El sistema permitirá comparar los vectores y guardar momentáneamente en otro.

Identificación del requerimiento	RQ5
Nombre del requerimiento	Generar operación de unión
Funcionalidades	• Realiza la operación de la unión por medio de la comparación del vector y obtiene el resultado. • Verificar que en la función de unión no se repita los datos.
Descripción del requerimiento	El sistema permitirá obtener la operación de la unión de conjuntos.

Identificación del requerimiento	RQ6
Nombre del requerimiento	Generar operación de intersección
Funcionalidades	• Realiza la operación de la intersección y obtiene un resultado.
Descripción del requerimiento	El sistema permitirá obtener la intersección de conjuntos.

Identificación del requerimiento	RQ7
Nombre del requerimiento	Generar operación de diferencia
Funcionalidades	• Realiza la operación de la diferencia y obtiene un resultado.
Descripción del requerimiento	El sistema permitirá obtener la diferencia de conjuntos.

Identificación del requerimiento	RQ8
Nombre del requerimiento	Generar operación de complemento
Funcionalidades	• Realiza la operación del complemento y obtiene un resultado.
Descripción del requerimiento	El sistema permitirá obtener el complemento de conjuntos.

Funcionalidades:

F1. Ingresar dos conjuntos

F2. Verificar que los datos ingresados no excedan de la dimensión del vector

F3. Verificar que los datos ingresados sean enteros.

F4. Verificar que no se ingresen letras o caracteres especiales.

F5. Verificar el ingreso de las dimensiones de los vectores

F5. Verificar límite de salida del nuevo vector

F6. Verificar la comparación y guarda en otro vector.

F7. Realizar la operación de la unión por medio de la comparación del vector y obtiene el resultado.

F8. Realiza la operación de la unión por medio de la comparación del vector y obtiene el resultado.

F9. Realiza la operación de la diferencia y obtiene un resultado.

F10. Realiza la operación del complemento y obtiene un resultado.

F11. Verificar que en la función de unión no se repita los datos.

Casos de prueba

- **Ingreso de dos conjuntos**

Regla: $(\{A\},\{B\}) \in R$

CP1(({1}, {1}), LlenarA)

CP2(({0}, {1}), LlenarA)

CP3(({1}, {0}), LlenarA)

CP4(({-1}, {1}), LlenarA)

CP5(({1}, {-1}), LlenarA)

CP6(({1}, {1}), LlenarB)

CP7(({0}, {1}), LlenarB)

CP8(({1}, {0}), LlenarB)

CP9(({-1}, {1}), LlenarB)

CP10(({1}, {-1}), LlenarB)

- **Ingreso de dimensión en cada conjunto**

Regla: (([n],[m]) ∈ R & (n,m) > 0)

Regla: n, m representa dimensión del vector

CP11(([1], [1]), LlenarA)

CP12(([1], [1]), LlenarB)

- **Generar diferencia de conjuntos**

Regla1: ({∃ A,B ∈ R: A-B =A })

CP13(([0],[0]), llenar A)

CP14(([1],[2]), llenar A)

CP15(([3],[3]), llenar A)

CP16(([4],[2]), llenar A)

CP17(([3],[5]), llenar A)

CP18(([5],[1]), llenar A)

CP19(([1],[0]), llenar A)

CP20(([4],[10]), llenar A)

CP21(([4],[2]), llenar A)

CP22(([3],[5]), llenar A)

CP23(([2],[1]), llenar A)

CP24(([0],[3]), llenar A)

- **Generar complemento de conjuntos**

Regla1: ({∃ A, B ∈ R: Ac ∈ A v Ac ∈B= A-B})

CP25(([1],[2]), llenar A-B)

CP26(([2],[1]), llenar A-B)

CP27(([1],[4]), llenar A-B)

CP28(([6],[7]), llenar A-B)

CP29(([5],[6]), llenar A-B)

CP30(([3],[0]), llenar A-B)

CP31([01,[3]), llenar A-B)

CP32(([0],[7]), llenar A-B)

CP33(([1],[8]), llenar A-B)

CP34(([9],[3]), llenar A-B)

Matriz de trazabilidad												
Casos de prueba	RQ1			RQ2	RQ3	RQ4	RQ5	RQ6	RQ7	RQ8		
	F1	F2	F3	F4	F5	F6	F7	F8	F9		F10	F11
CP1(({1}, {1}), LlenarA)	X	x	x									
CP2(({0}, {1}), LlenarA)	X	X										
CP3(({1}, {0}), LlenarA)	X	X										
CP4(({-1}, {1}), LlenarA)	X	X		X								

CP5(({1}, {-1}), LlenarA)	X	X		X								
CP6(({1.9}, {1.4}), LlenarB)	X	X		X								
CP7(({0}, {1}), LlenarB)	X	X	X				X					
CP8(({1}, {0}), LlenarB)	X	X				X						
CP9(({-1}, {1}), LlenarB)	X	X		X				X	X			
CP10(({1}, {-1}), LlenarB)	X	X										
CP11(([1], [1]), LlenarA)	X	X										
CP12(([1], [1]), LlenarB)	X	X										
CP13(([0],[0]), llenar A)	X	X					X	X	X			
CP14(([1],[2]), llenar A)	X	X					X	X	X			
CP15(([3],[3]), llenar A)	X	X	X				X	X	X			
CP16(([4],[2]), llenar A)	X			X					X			
CP17(([3],[5]), llenar A)	X		X						X			
CP18(([5],[1]), llenar A)	X	X							X			
CP19(([1],[0]), llenar A)	X								X			
CP20(([4],[10]), llenar A)	X	X		X					X			
CP21(([4],[2]), llenar A)	X	X		X					X			
CP22(([3],[5]), llenar A)	X	X		X					X			

CP23(([2],[1]), llenar A)	X		X	X					X			
CP24(([0],[3]), llenar A)	X	X		X					X			
CP25(([1],[2]), llenar A-B)	X		X	X							X	
CP26(([2],[1]), llenar A-B)	X			X							X	
CP27(([1],[4]), llenar A-B)	X			X							X	
CP28(([6],[7]), llenar A-B)	X	X		X							X	
CP29(([5],[6]), llenar A-B)	X		X	X							X	
CP30(([3],[0]), llenar A-B)	X	X		X							X	
CP31([01,[3]), llenar A-B)	X	X		X							X	
CP32(([0],[7]), llenar A-B)	X	X		X							X	
CP33(([1],[8]), llenar A-B)	X		X	X							X	
CP34(([9],[3]), llenar A-B)	X		X	X							X	

Código

```
package Circulos;

import java.util.Scanner;

/*
 * @Autores: Jemmy Puzma 6627
 *                   Juan Carrera 6605
 *                   David Llumitaxi 6700
 *                   Victor Montaluisa 6563
 */
public class cConjuntos {
	int[] A, B, C;
```

```
int tamA, tamB, x = 0;
Scanner entrada = new Scanner(System.in);

public void llenarA() {
   System.out.println("\n-----------------------------------------\n"
         + "-   *****   Ingrese el tamaño de A  *****   -\n"
         + "-----------------------------------------");
       tamA = entrada.nextInt();
       A = new int[tamA];




   System.out.println("\n-------------------------\n"
         + "-       llenando A      -\n"
         + "-------------------------");
   System.out.println("Ingrese un numero -> ");
      for (int i = 0; i < tamA; i++) {

         A[i] = entrada.nextInt();
      }
}

public void llenarB() {
   System.out.println("\n-----------------------------------------\n"
         + "-   *****   Ingrese el tamaño de B  *****   -\n"
         + "-----------------------------------------");
   tamB = entrada.nextInt();
   B = new int[tamB];

   System.out.println("\n-------------------------\n"
         + "-       llenando B      -\n"
         + "-------------------------");
   System.out.println("Ingrese un numero -> ");
   for (int i = 0; i < tamB; i++) {

      B[i] = entrada.nextInt();
   }
}

public void Union() {
   C = new int[100];

   for (int i = 0; i < A.length; i++) {
      for (int j = 0; j < B.length; j++) {
         if (A[i] == B[j]) {
            A[i] = 0;
         }
      }
   }
```

```
    for (int i = 0; i < B.length; i++) {
        for (int j = 0; j < A.length; j++) {
            if (B[i] == A[j]) {
                B[i] = 0;
            }
        }
    }

    System.out.println("\n------------------------\n"
            + "-       A UNION B      -\n"
            + "------------------------");
    System.out.print("A U B = { ");
    for (int i = 0; i < A.length; i++) {

        if (A[i] != 0) {
            System.out.print(A[i]+" ");
        }
    }

    for (int i = 0; i < B.length; i++) {
        System.out.print(" "+ B[i]);

    }
    System.out.print(" }");
}

public void Interseccion() {
    int s = 0;
    C = new int[100];

    for (int i = 0; i < A.length; i++) {
        for (int j = 0; j < B.length; j++) {
            if (A[i] == B[j]) {
                C[s] = A[i];
                s++;
            }
        }
    }

    System.out.println("\n--------------------------------\n"
            + "-       A INTERSECCION B      -\n"
            + "--------------------------------");
    System.out.print("A n B = { ");
    for (int i = 0; i < s; i++) {
        System.out.print(C[i] + " ");
    }
    System.out.print(" }");
}

public void Diferencia() {
    C = new int[100];
```

```
    int k = 0, cont;

    for (int i = 0; i < A.length; i++) {
        cont = 0;
        for (int j = 0; j < B.length; j++) {
            if (A[i] == B[j]) {
                cont++;
            }
        }

        if (cont == 0) {
            C[k] = A[i];
            k++;
        }

    }

    System.out.println("\n----------------------------\n"
            + "-          A - B         -\n"
            + "----------------------------");
    System.out.print("A - B = { ");
    for (int i = 0; i < k; i++) {
        System.out.print(C[i] + " ");
    }
    System.out.print(" }");
}

public void Complemento() {
    C = new int[100];
    for (int i = 0; i < A.length; i++) {
        for (int j = 0; j < B.length; j++) {
            if (A[i] == B[j]) {
                B[j] = 0;
            }
        }
    }

    System.out.println("\n----------------------------\n"
            + "-       A COMPLEMENTO B    -\n"
            + "----------------------------");
    System.out.print("A' B = { ");
    for (int i = 0; i < B.length; i++) {
        if (B[i] != 0) {
            System.out.print(B[i]+" ");
        }
    }
    System.out.print(" }");
}

public void menu() {
    int opc;
```

```
        System.out.println("\n\n----------------------------------------\n"
                + "-       OPERACIONES CON CONJUNTOS       -\n"
                + "----------------------------------------\n"
                + "-    1. Union                        -\n"
                + "-    2. Interseccion                 -\n"
                + "-    3. A diferencia B               -\n"
                + "-    4. A complemento B              -\n"
                + "-    5. Salir                        -\n"
                + "----------------------------------------");
        System.out.println("-   Digite su opcion                    -\n"
                + "----------------------------------------\n");

        opc = entrada.nextInt();
        switch (opc) {
           case 1:
              llenarA();
              llenarB();
              Union();
              menu();
              break;
           case 2:
              llenarA();
              llenarB();
              Interseccion();
              menu();
              break;
           case 3:
              llenarA();
              llenarB();
              Diferencia();
              menu();
              break;
           case 4:
              llenarA();
              llenarB();
              Complemento();
              menu();
              break;
           case 5:
              System.exit(1);
              break;
           default:
              menu();
              break;
package Circulos;

import java.util.Scanner;

public class App {

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

```
                // TODO Auto-generated method stub
                cConjuntos oConjuntos = new cConjuntos();
        oConjuntos.menu();
            }
}
```