

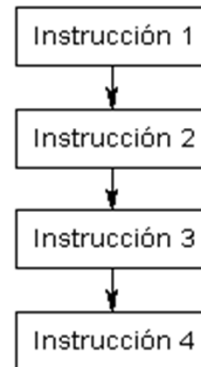
Tema 4: Estructuras de control

- Estructura secuencial
- Estructura alternativa
- Estructuras repetitivas
- Variables de trabajo

1. Estructura secuencial

- Indica el orden en que se ejecutan las instrucciones.
- Las instrucciones se ejecutan en orden de arriba-abajo y de izquierda-derecha.
- Representación:

FLUJOGRAMA



PSEUDOCÓDIGO

```
INSTRUCCIÓN 1  
INSTRUCCIÓN 2  
INSTRUCCIÓN 3  
INSTRUCCIÓN 4
```

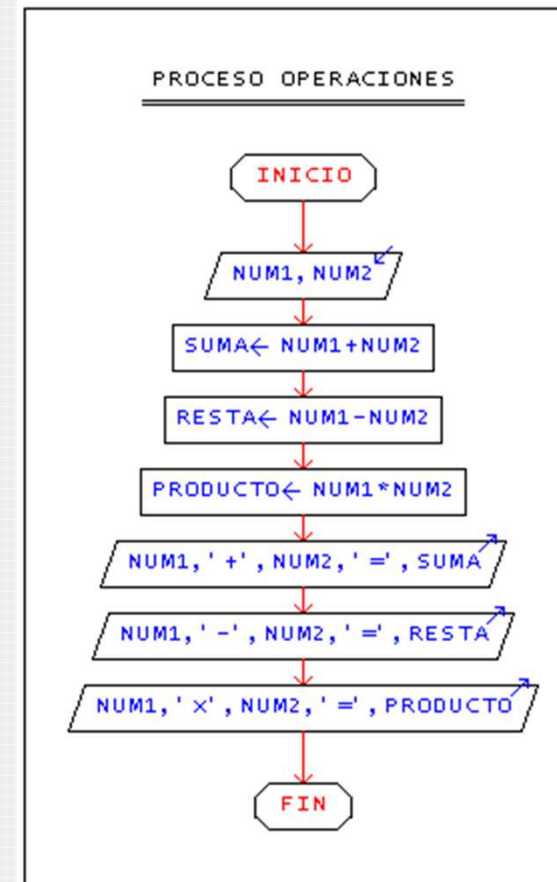
- En JAVA la secuencia se indica con el ;

Ejemplo:

- Diseñar un programa que calcule la suma, resta, y multiplicación de dos valores introducidos desde el teclado.

Algoritmo

```
1  Proceso operaciones
2      // Datos
3      //num1 y num2 son enteros
4      //suma, resta, producto son enteros
5      Leer num1,num2;
6      suma<-num1+num2;
7      resta<-num1-num2;
8      producto<-num1*num2;
9      Escribir num1,"+",num2,"=",suma;
10     Escribir num1,"-",num2,"=",resta;
11     Escribir num1,"x",num2,"=",producto;
12 FinProceso
```



Algoritmo en JAVA

Proceso operaciones

```
// Datos  
//num1 y num2 son enteros  
//suma, resta, producto son enteros
```

```
Leer num1,num2;
```

```
suma<-num1+num2;  
resta<-num1-num2;  
producto<-num1*num2;
```

```
Escribir num1,"+",num2,"=",suma;  
Escribir num1,"-",num2,"=",resta;  
Escribir num1,"x",num2,"=",producto;
```

FinProceso

```
void operaciones() {  
    int num1,num2,suma,resta,producto;  
    Scanner teclado=new Scanner(System.in);  
    System.out.println("Dame num1:");  
    num1=teclado.nextInt();  
    System.out.println("Dame num2:");  
    num2=teclado.nextInt();  
    suma=num1+num2;  
    resta=num1-num2;  
    producto=num1*num2;  
    System.out.println(num1+" "+num2+"="+suma);  
    System.out.println(num1+" "+num2+"="+resta);  
    System.out.println(num1+"x"+num2+"="+producto);  
}
```


Probando el algoritmo.

Traza

	num1	num2	suma	resta	producto	SALIDA
Leer num1,num2;	12	23				
suma<-num1+num2;			35			
resta<-num1-num2;				-11		
producto<-num1*num2;					276	
Escribir num1,"+",num2,"=",suma;						12+23=35
Escribir num1,"-",num2,"=",resta;						12-23=-11
Escribir num1,"x",num2,"=",producto;						12x23=276

Estructura Alternativa

- Controlan la ejecución o no de una o más instrucciones en función de una condición.

Tipos:

- **Simple:** ejecuta un conjunto de acciones si se cumple la condición
- **Doble:** ejecuta un conjunto de acciones si se cumple una condición y otro conjunto diferente si no se cumple
- **Múltiple:** ejecuta un conjunto de acciones diferente en función del valor una expresión

Estructura Alternativa Simple

TIPO	FLUJOGRAMA	PSEUDOCÓDIGO
ALTERNATIVA SIMPLE	<pre>graph TD; Entry(()) --> Cond{CONDICIÓN}; Cond -- SI --> Bloque[Bloque de instrucciones]; Cond -- NO --> Conn1(()); Bloque --> Conn2(()); Conn1 --> Conn2; Conn2 --> Exit(())</pre>	SI (CONDICION) INSTRUCCIÓN 1 INSTRUCCIÓN 2 INSTRUCCIÓN N FINSI

Estructura simple en JAVA

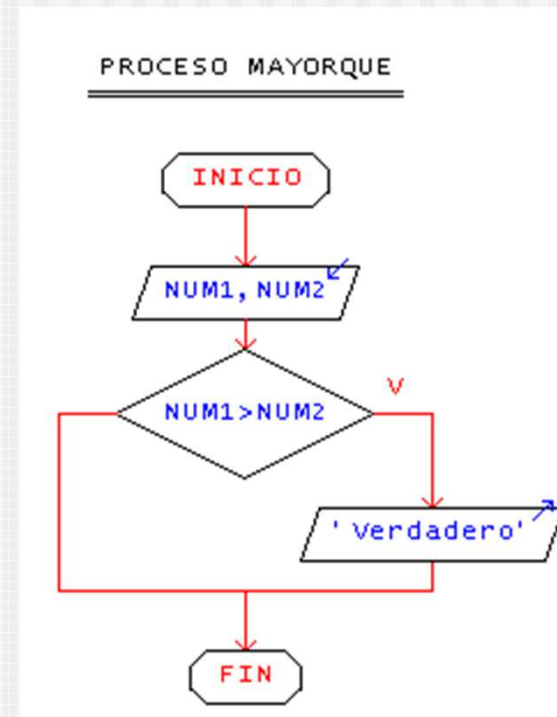
- Esta estructura permite ejecutar una o más instrucciones en función del cumplimiento o no de la condición

```
if (condición) {  
    // instrucción 1  
    // ...  
    // instrucción n  
}
```

Ejemplo:

- Diseñar un algoritmo que pida dos números y conteste "Verdadero" en caso en que el primero sea mayor que el segundo.

```
1  Proceso mayorque
2      Leer num1,num2 ;
3      Si num1>num2 Entonces
4          Escribir "Verdadero";
5      FinSi
6  FinProceso
```



En Java

```
1  Proceso mayorque
2      Leer num1,num2;
3      Si num1>num2 Entonces
4          Escribir "Verdadero";
5      FinSi
6  FinProceso
```

```
void mayorque() {
    int num1, num2;
    Scanner teclado = new Scanner(System.in);
    System.out.println("Dame num1:");
    num1 = teclado.nextInt();
    System.out.println("Dame num2:");
    num2 = teclado.nextInt();
    if (num1 > num2) {
        System.out.println("Verdadero");
    }
}
```

Probando el algoritmo

Caso 1: $\text{num1} > \text{num2}$

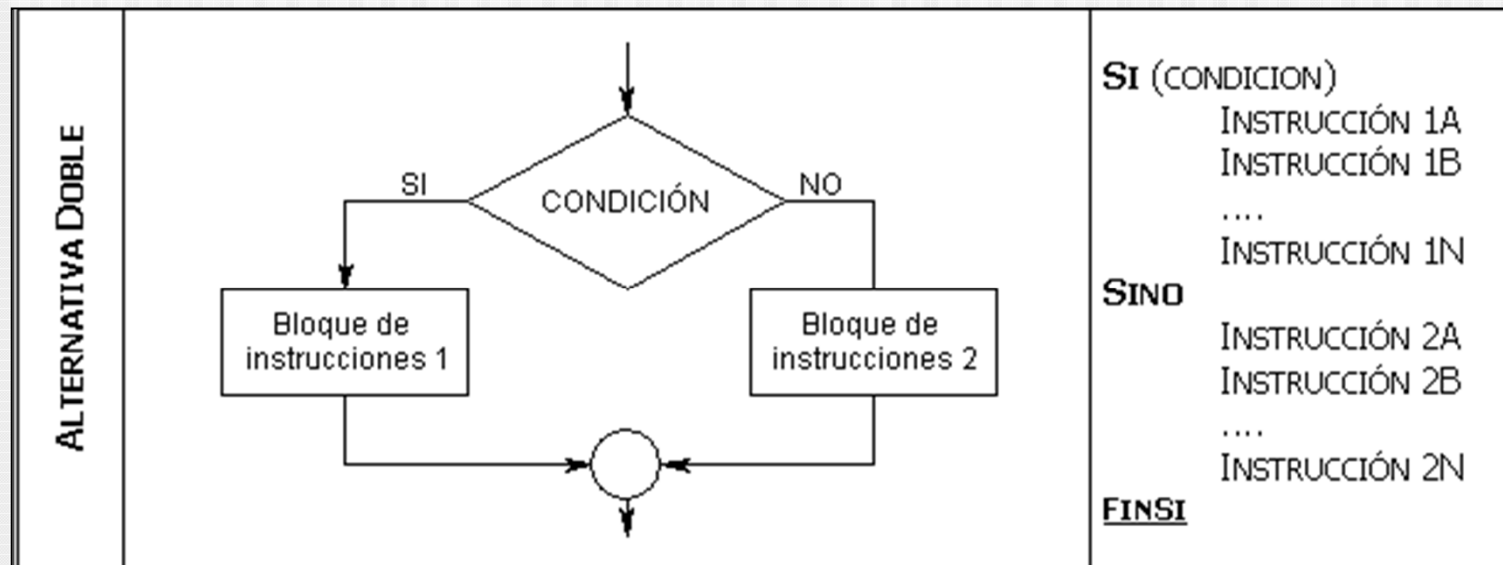
	num1	num2	SALIDA
Leer num1,num2;	13	12	
Si $\text{num1} > \text{num2}$ Entonces	13 > 12	V	
Escribir "Verdadero";			Verdadero
FinSi			

Probando el algoritmo

Caso 2: $\text{num1} \leq \text{num2}$

	num1	num2	SALIDA
Leer num1,num2;	13	22	
Si $\text{num1} > \text{num2}$ Entonces			$13 < 22$ F
Escribir "Verdadero";			
FinSi			

Estructura Alternativa Doble



Estructura doble en JAVA

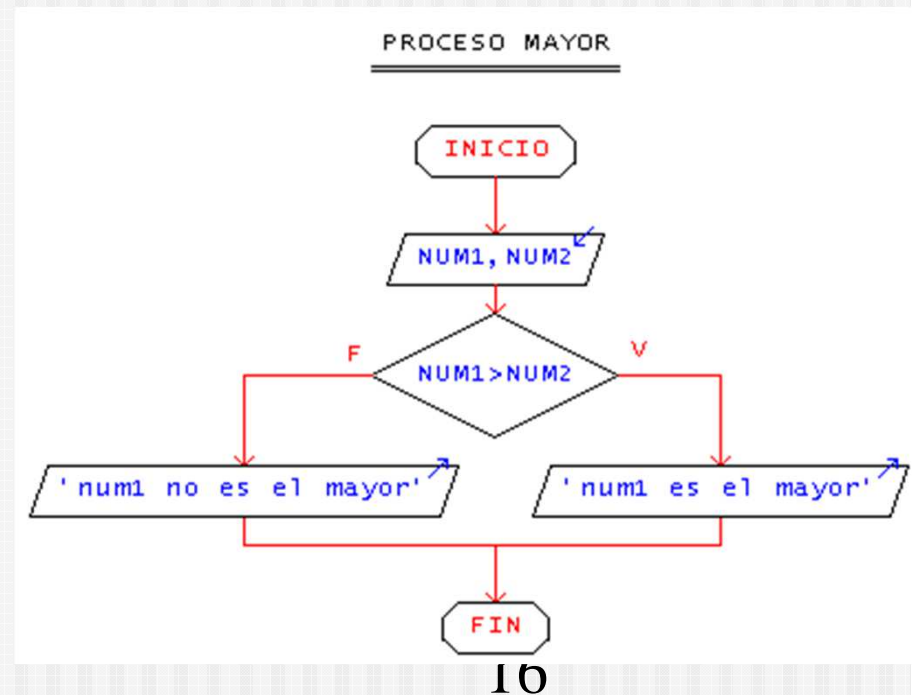
- Esta estructura permite ejecutar una o más instrucciones en función del cumplimiento de la condición y otras si no se cumple

```
if (condición){  
    // instrucción 1A  
    //...  
    // instrucción 1N  
} else {  
    // instrucción 2A  
    //...  
    // instrucción 2N  
}
```

Ejemplo:

- Diseñar un algoritmo que pida dos números y conteste "num1 es mayor" en caso en que el primero sea mayor que el segundo, o "num1 no es el mayor" en caso contrario.

```
1  Proceso mayor
2    Leer num1,num2 ;
3    Si num1>num2 Entonces
4      Escribir "num1 es el mayor";
5    Sino
6      Escribir "num1 no es el mayor";
7    FinSi
8  FinProceso
```



En Java

```
1  Proceso mayor
2      Leer num1,num2;
3      Si num1>num2 Entonces
4          Escribir "num1 es el mayor";
5      Sino
6          Escribir "num1 no es el mayor";
7      FinSi
8  FinProceso
```

```
void mayorquedoble() {
    int num1, num2;
    Scanner teclado = new Scanner(System.in);
    System.out.println("Dame num1:");
    num1 = teclado.nextInt();
    System.out.println("Dame num2:");
    num2 = teclado.nextInt();
    if (num1 > num2) {
        System.out.println("num1 es mayor");
    } else {
        System.out.println("num1 no es mayor");
    }
}
```

Probando el algoritmo

Caso 1: $\text{num1} > \text{num2}$

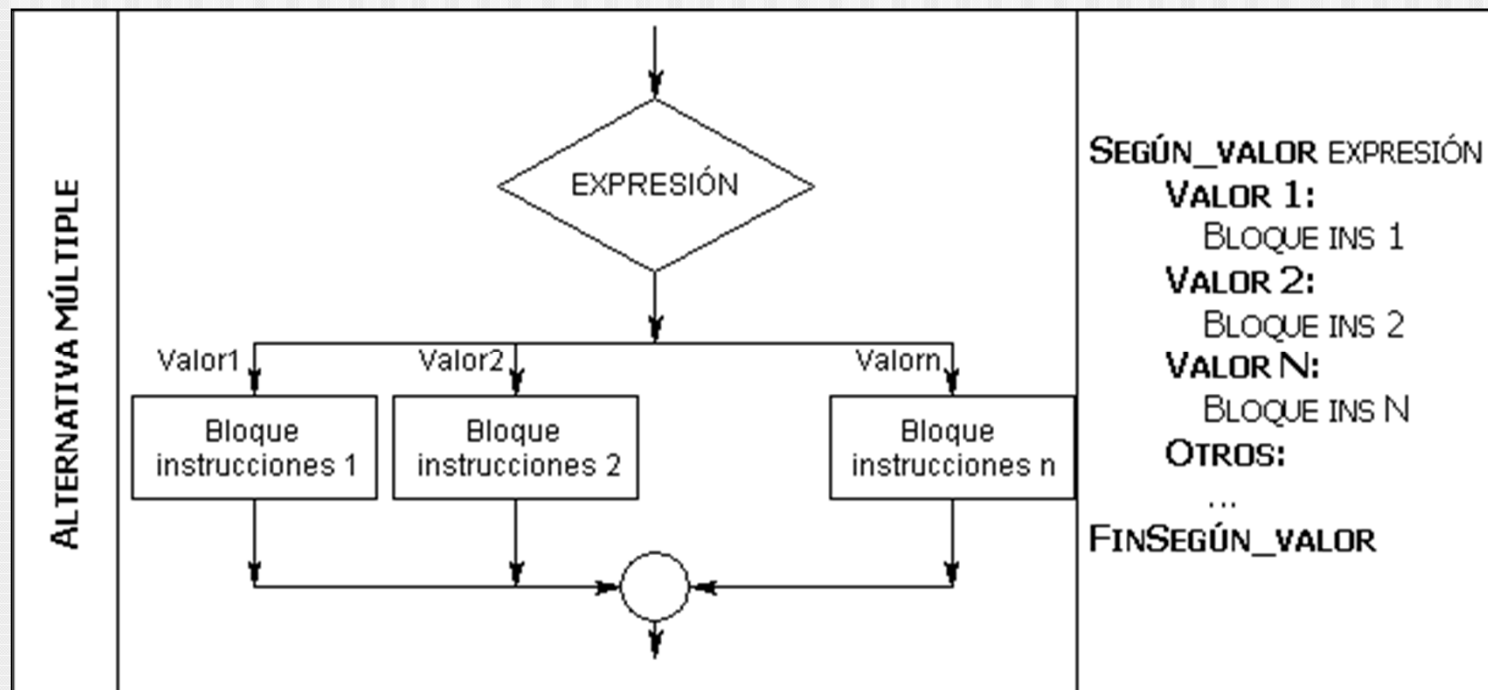
	num1	num2	SALIDA
Leer num1,num2;	13	12	
Si $\text{num1} > \text{num2}$ Entonces	13 > 12 V		
Escribir "num1 es mayor";			num1 es mayor
Sino			
Escribir "num1 no es mayor";			
FinSi			

Probando el algoritmo

Caso 2: $\text{num1} \leq \text{num2}$

	num1	num2	SALIDA
Leer num1,num2;	23	12	
Si $\text{num1} > \text{num2}$ Entonces	23 > 12	F	
Escribir "num1 es mayor";			
Sino			
Escribir "num1 no es mayor";			num1 no es mayor
FinSi			

Estructura Alternativa Múltiple



En JAVA

```
switch (expresion){  
    case valor2: //Instrucciones  
                break;  
    case valor2: //Instrucciones  
                break;  
  
    ....  
  
    default: //Instrucciones  
}
```

Limitaciones en JAVA

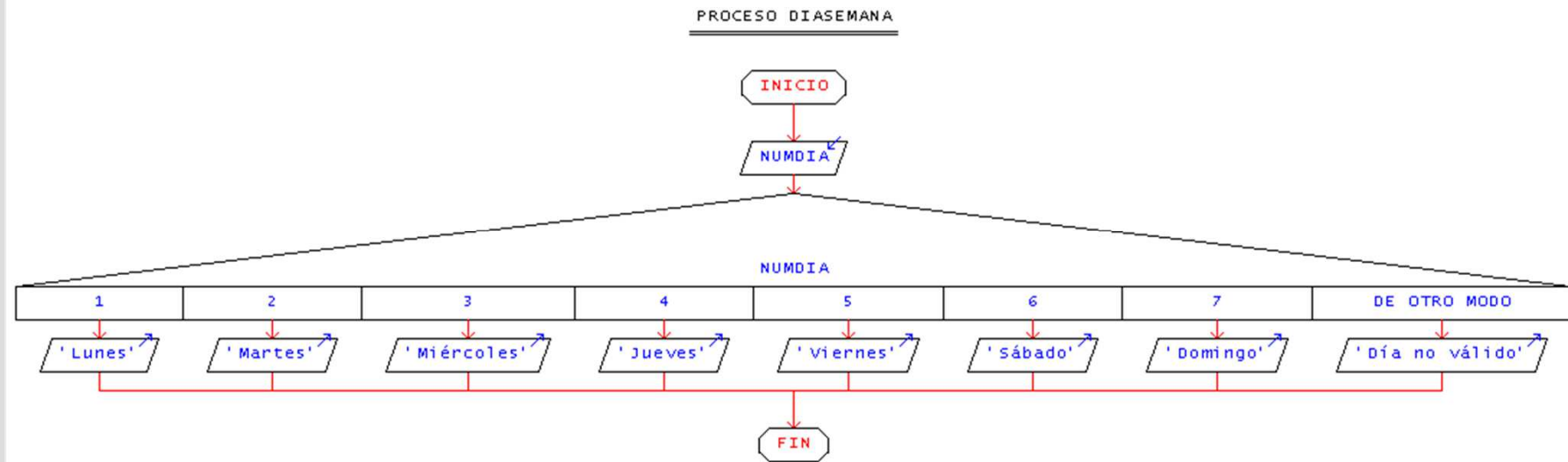
- En la expresión solo se permiten enteros y caracteres (*char*, *byte*, *short* o *int*)
- Funcionamiento: evalúa la expresión hasta encontrar el valor, realiza las acciones descritas hasta encontrar el *break* o finalizar el switch.
- La opción *default*, si está, sirve para realizar acciones en caso de no encontrar ningún valor que se ajuste.

Ejemplos:

- Crea un algoritmo llamado **Diasemana** que permita visualizar el nombre del día de la semana en función de una variable entera que se introduce desde el teclado.

Pseudocódigo y flujograma

```
Proceso diasemana
//numDia es una variable entera
Leer numDia;
Segun numDia Hacer
    1: Escribir "Lunes";
    2: Escribir "Martes";
    3: Escribir "Miércoles";
    4: Escribir "Jueves";
    5: Escribir "Viernes";
    6: Escribir "Sábado";
    7: Escribir "Domingo";
    De Otro Modo: Escribir "Día no válido";
FinSegun
FinProceso
```



En JAVA

```
void diaSemana() {  
    int numDia;  
    Scanner teclado = new Scanner(System.in);  
    //Leer segundos a transformar  
    System.out.println("Dame nº día:");  
    numDia = teclado.nextInt();  
    switch (numDia) {  
        case 1: System.out.println("Lunes"); break;  
        case 2: System.out.println("Mares"); break;  
        case 3: System.out.println("Miércoles"); break;  
        case 4: System.out.println("Jueves"); break;  
        case 5: System.out.println("Viernes"); break;  
        case 6: System.out.println("Sábado"); break;  
        case 7: System.out.println("Domingo"); break;  
        default: System.out.println("Día no válido");  
    }  
}
```

Anidar condicionales

```
Proceso relacion_n1n2
  Leer n1,n2
  Si n1>n2 Entonces
    Escribir n1," es mayor que ",n2
  Sino
    Si n1<n2 Entonces
      Escribir n1," es menor que ",n2
    Sino
      Escribir n1," es igual a ",n2
    Fin Si
  Fin Si
FinProceso
```

