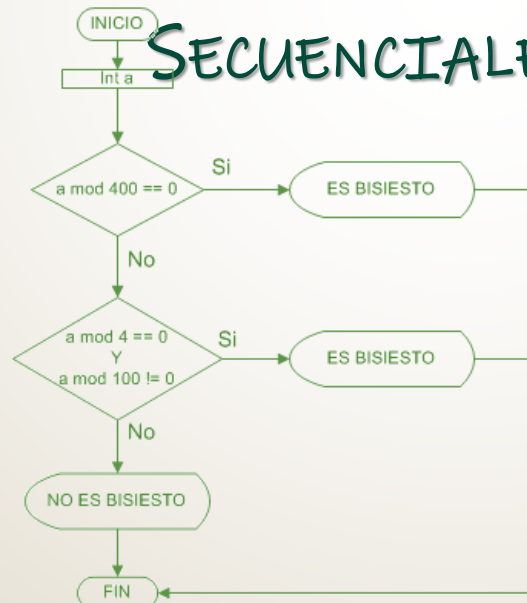


Programación Estructurada

ESTRUCTURAS DE CONTROL

SECUENCIALES Y SELECTIVAS



Índice

- Estructura y elementos de programa
- Programación Estructurada
- Teorema de la PE
- Estructuras de Control
 - Secuenciales
 - Selectivas
 - Repetitivas
- Anidamiento de Control
- Prueba de escritorio

Estructura de Programa

- Al escribir un programa, éste debe incluir al menos:
 - Nombre de programa
 - Declaración de variables y constantes
 - Inicio de programa
 - Cuerpo del programa
 - Fin de Programa

Elementos de Programa (1)

- Palabras Reservadas: palabras que tienen un significado especial para los lenguajes de programación.
- Identificadores: “nombres” que se dan a las variables, módulos, y otros elementos en un programa.
- Caracteres especiales: símbolos que tienen un significado especial. Por ejemplo: +, -, *, / se utilizan para indicar operaciones aritméticas.

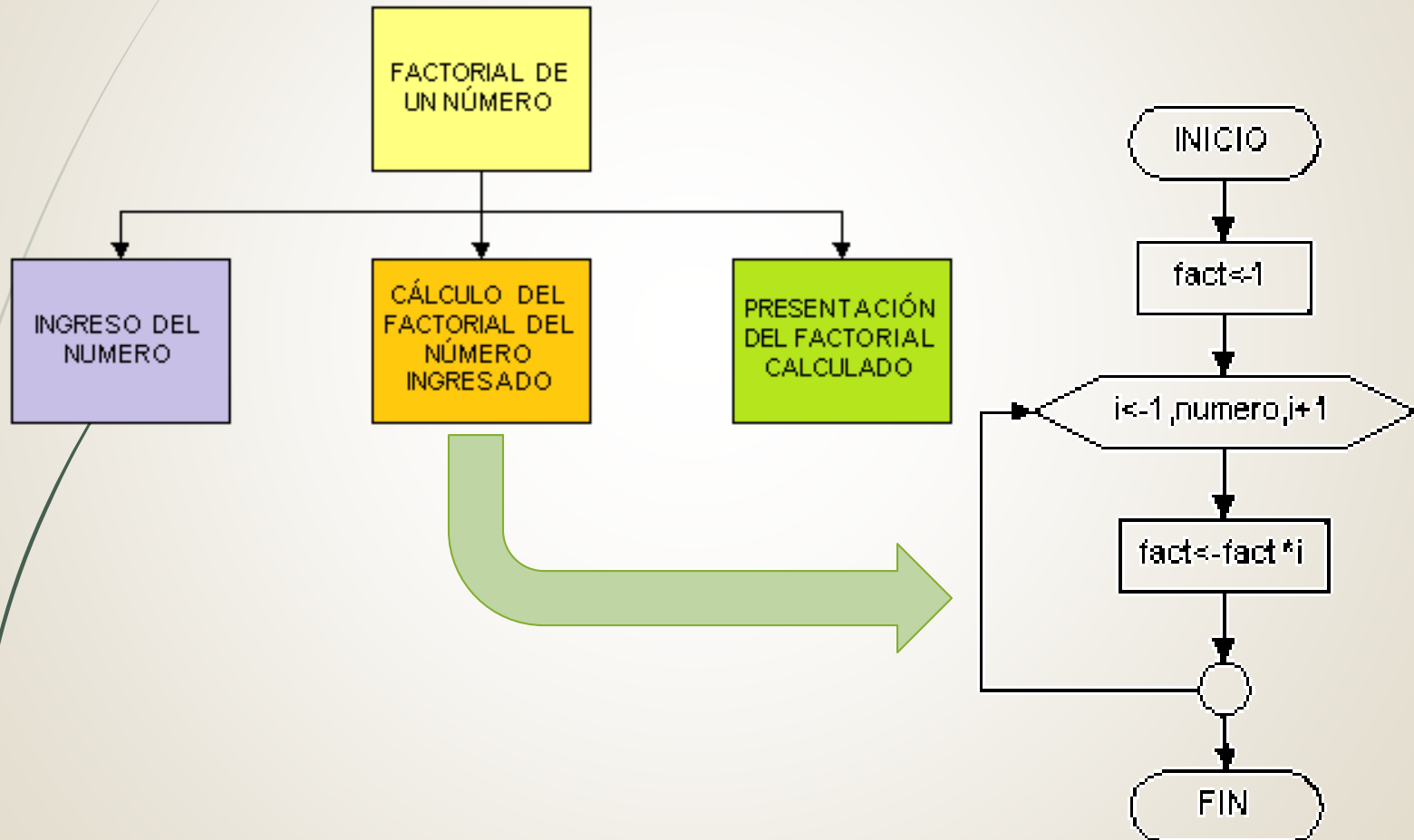
Elementos de Programa (2)

- **Constantes:** elementos de datos que no se modifican, se nombran mediante identificadores. Por ejemplo: $\pi=3,14159265$
- **Variables:** elementos de datos modificables
 - contadores, acumuladores, banderas
- **Expresiones:** combinación de variables, constantes y operadores.
- **Instrucciones**
 - secuenciales, selectivas y repetitivas (bucles)

Programación Estructurada

- Los programas tienen una estructura.
- La PE permite desarrollar programas que son más fáciles de escribir, verificar, leer y mantener.
- Técnicas de la PE:
 - Diseño Top-Down (descomposición del problema)
 - Recursos Abstractos (acciones simples ejecutables por la computadora)
 - Estructuras de Control (instrucciones secuenciales selectivas y repetitivas)

¿Qué hace? ¿Cómo lo hace?

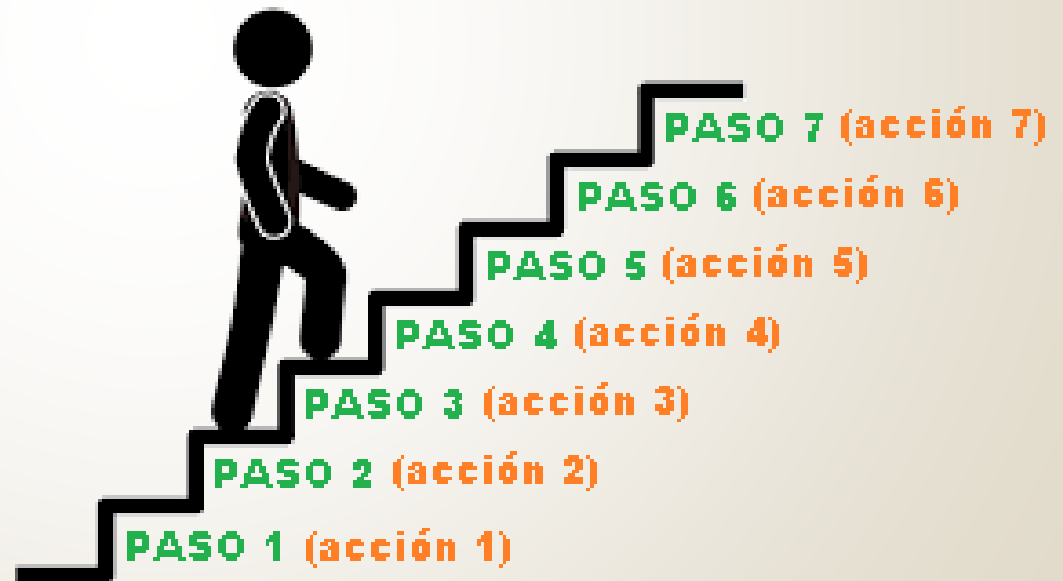


Teorema de la PE

- El *Teorema de la PE* dice que un programa **propio** puede ser escrito usando sólo estructuras de control:
 - Secuenciales
 - Selectivas
 - Repetitivas
- Un programa es **propio** si:
 - tiene un único punto de entrada y salida para el control del programa, y
 - todas las instrucciones son ejecutables.

Estructuras Secuenciales (1)

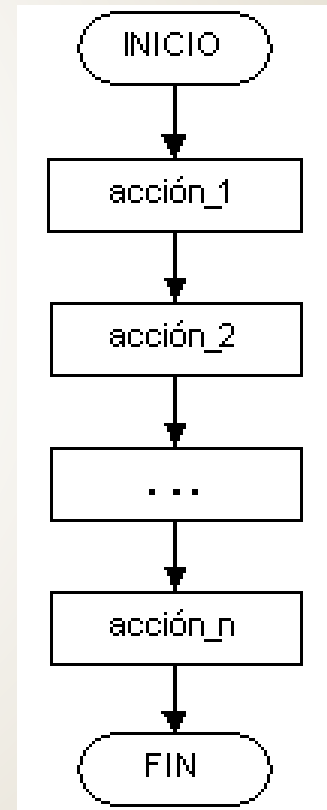
- Sucesión de operaciones, en la que el orden de ejecución coincide con el orden físico de aparición de las instrucciones.



Estructuras Secuenciales (2)

➡ LEER, ESCRIBIR y ASIGNACIÓN (←)

```
INICIO
    acciones_1
    acciones_2
    ...
    acciones_N
FIN
```



Representación

➤ LEER

LEER variables



➤ ESCRIBIR

ESCRIBIR "Mensaje", variables



➤ ASIGNACIÓN (←)

variable ← expresión



Ejemplo Secuenciales (1)

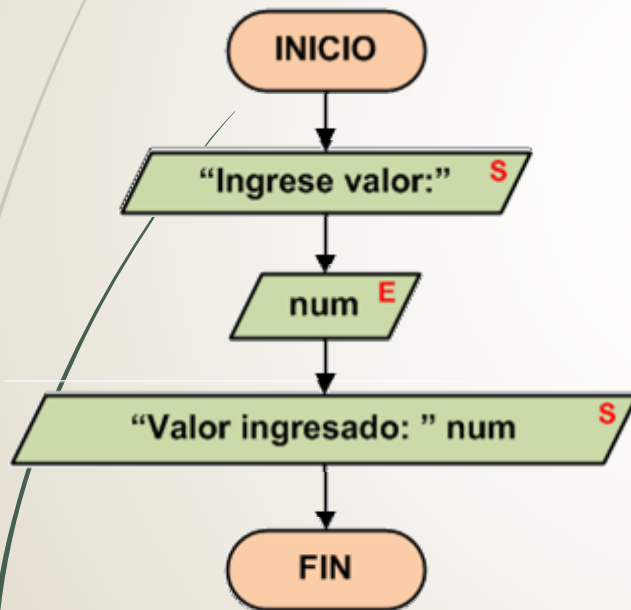
- Diseñe un algoritmo que muestre el mensaje “Programación Estructurada”



```
PROGRAMA ejemplo1
INICIO
    ESCRIBIR "Programación Estructurada"
FIN
```

Ejemplo Secuenciales (2)

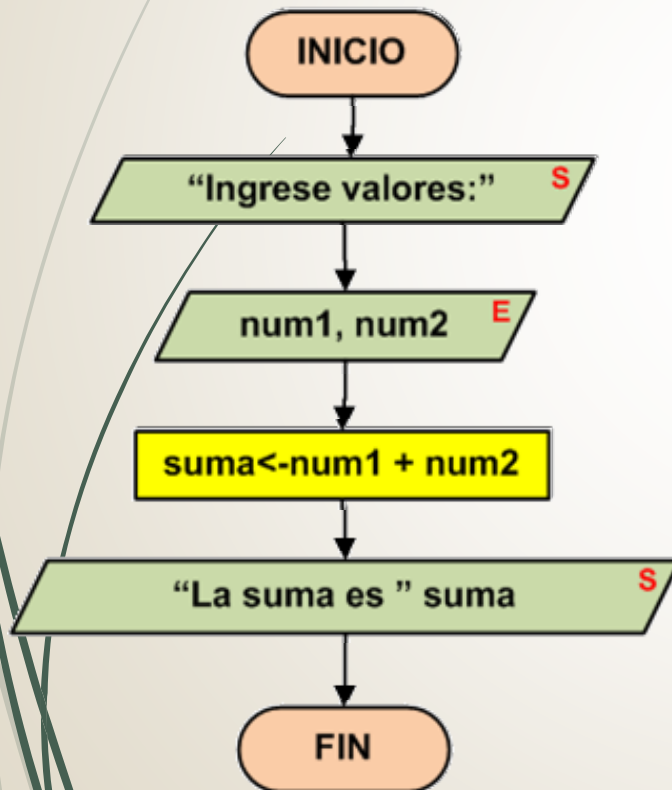
- Diseñe un algoritmo que muestre un valor ingresado por el usuario.



```
PROGRAMA ejemplo2
VARIABLES
    num: ENTERO
INICIO
    ESCRIBIR "Ingrese valor: "
    LEER num
    ESCRIBIR "Valor ingresado: " num
FIN
```

Ejemplo Secuenciales (3)

- Diseñe un algoritmo que sume 2 valores ingresados por el usuario.



```
PROGRAMA ejemplo3
VARIABLES
    num1, num2, suma: ENTERO
INICIO
    ESCRIBIR "Ingrese valores: "
    LEER num1, num2
    suma <- num1 + num2
    ESCRIBIR "La suma es: " suma
FIN
```


Ejemplo Secuenciales (4)

- Diseñe un algoritmo que calcule el perímetro y área de un círculo.
- Diseñe un algoritmo que determine el porcentaje de varones y el porcentaje de mujeres de un curso.
- Diseñe un algoritmo que convierta a segundos los datos de horas, minutos y segundos ingresados por el usuario.
- Diseñe un algoritmo que calcule la raíces de una ecuación cuadrática.



Estructuras Selectivas

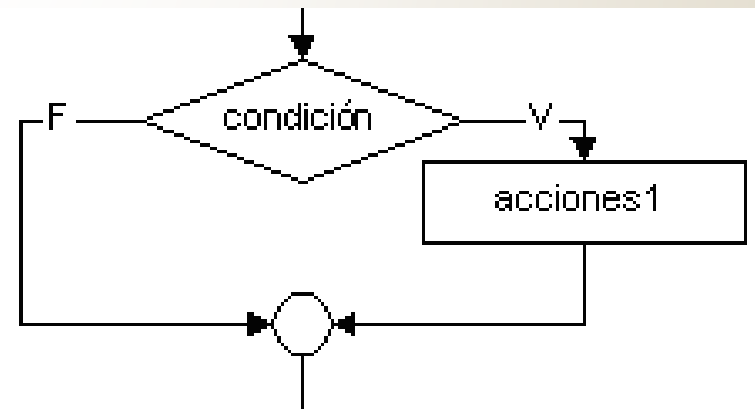
- ¿Qué ocurre con los problemas que no pueden resolverse sólo con estructuras secuenciales?
- Muchas veces es necesario elegir caminos alternativos de acción en base a condiciones del problema.
- Estructuras Selectivas
 - Simples
 - Dobles
 - Múltiples



Selectivas Simples

- La estructura **SI/ENTONCES/FIN_SI** permite realizar un conjunto de acciones si la condición que se evalúa es VERDADERA, caso contrario, dichas acciones se omiten.

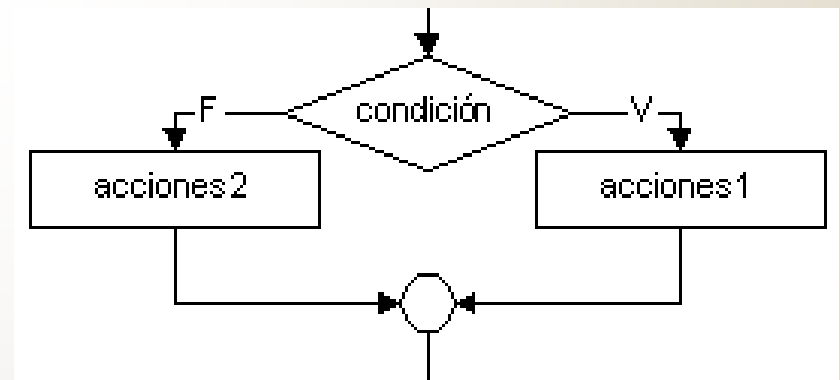
```
SI condición ENTONCES  
    acciones_1  
    acciones_2  
    ...  
    acciones_N  
FIN_SI
```



Selectivas Dobles

- La estructura **SI/ENTONCES/SINO/FIN_SI** presenta 2 caminos alternativos de acción, que se eligen según el valor de una condición (VERDADERA o FALSA).

```
SI condición ENTONCES  
    acciones_1  
SINO  
    acciones_2  
FIN_SI
```



Selectivas Múltiples (1)

- La estructura **SEGÚN/HACER/FIN_SEGÚN** elige las acciones a ejecutar entre n caminos alternativos.
- La elección del camino se basa en una expresión de tipo ordinal que puede tomar n valores distintos.



Selectivas Múltiples (2)

SEGÚN **valor** HACER

opción_1: acciones_1

opción_2: acciones_2

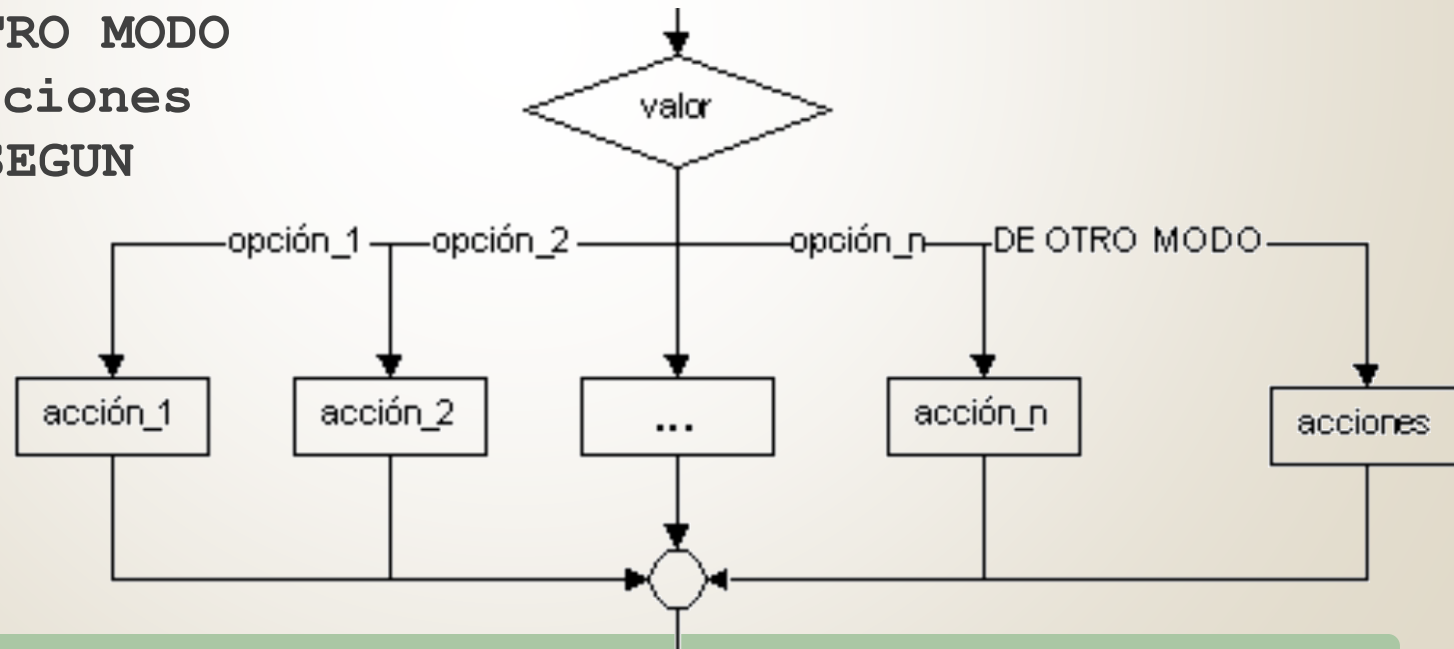
. . .

opción_n: acciones_n

DE OTRO MODO

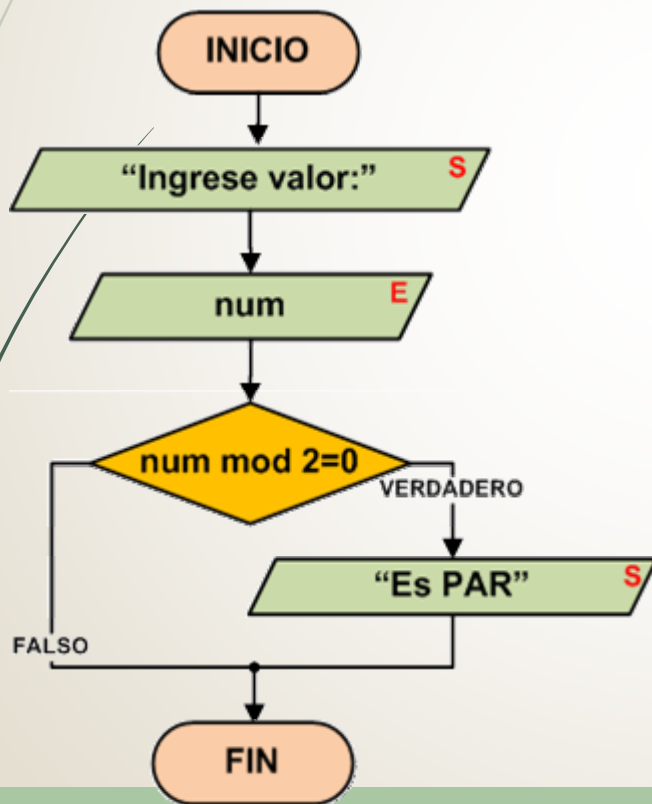
acciones

FIN_SEGUN



Ejemplo Selectivas (1)

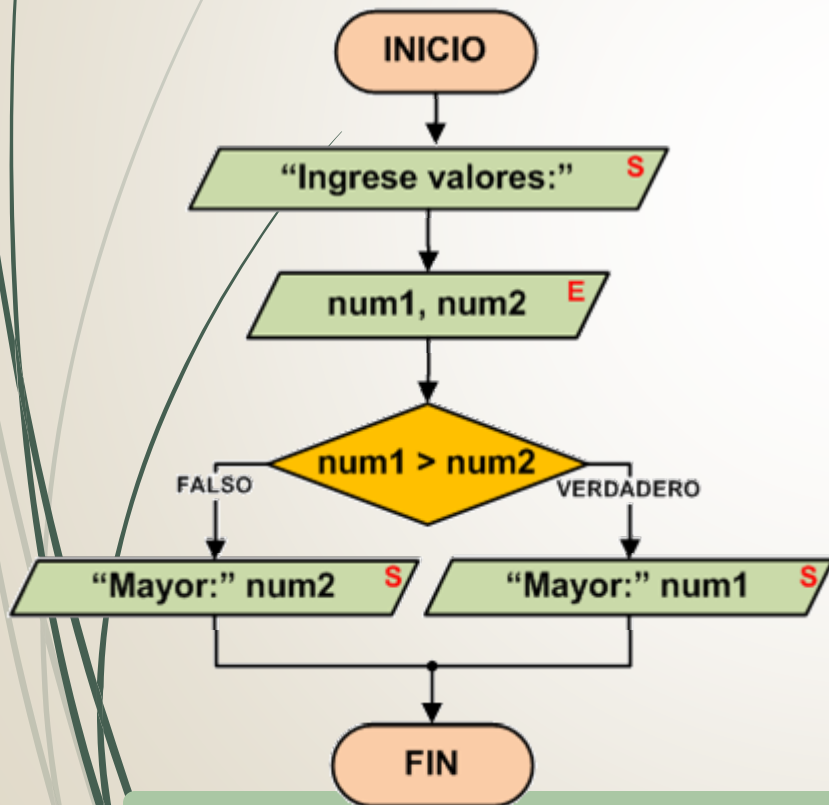
- Diseñe un algoritmo que determine si un número es par.



```
PROGRAMA ejemplo4
VARIABLES
    num: ENTERO
INICIO
    ESCRIBIR "Ingrese valor: "
    LEER num
    SI num mod 2 = 0 ENTONCES
        ESCRIBIR "El valor es PAR"
    FIN_SI
FIN
```

Ejemplo Selectivas (2)

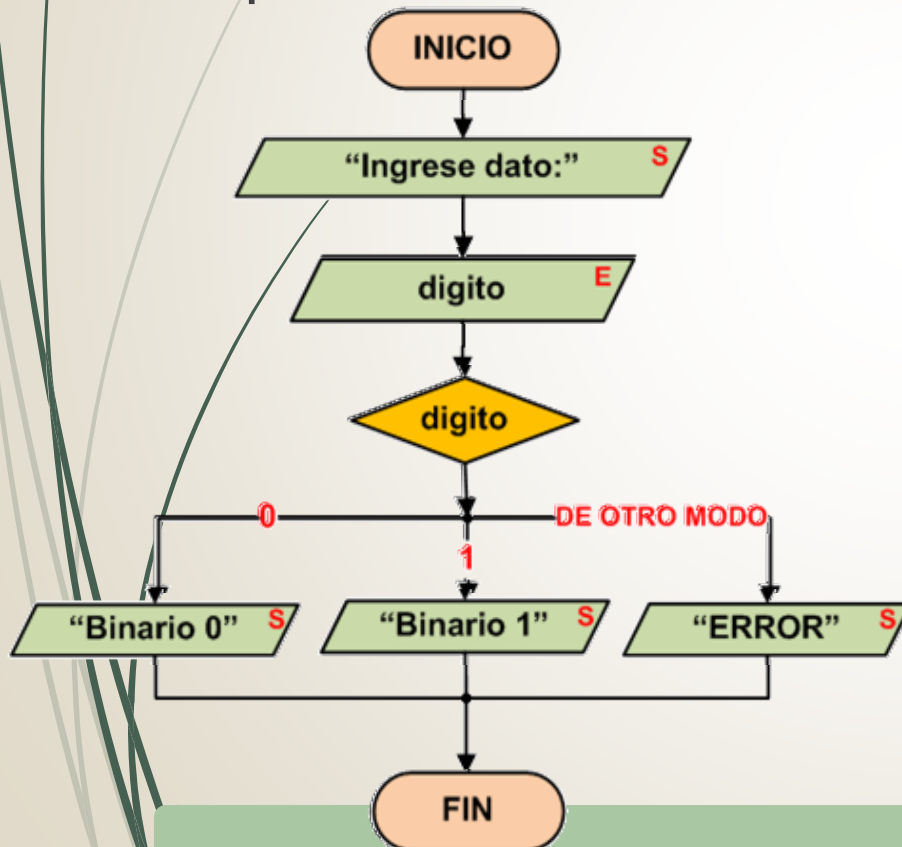
- Diseñe un algoritmo que determine el mayor de 2 valores.



```
PROGRAMA ejemplo5
VARIABLES
    num1,num2:ENTERO
INICIO
    ESCRIBIR "Ingrese valores: "
    LEER num1, num2
    SI num1 > num2 ENTONCES
        ESCRIBIR "El mayor es: " num1
    SINO
        ESCRIBIR "El mayor es: " num2
    FIN_SI
FIN
```

Ejemplo Selectivas (3)

- Diseñe un algoritmo que indique si un dígito pertenece al sistema binario (0 ó 1) o no.



```
PROGRAMA ejemplo6
VARIABLES
    digito: ENTERO
INICIO
    ESCRIBIR "Ingrese dato: "
    LEER digito
    SEGUN digito HACER
        0: ESCRIBIR "Binario 0"
        1: ESCRIBIR "Binario 1"
        De otro modo: ESCRIBIR "Error"
    FIN_SEGUN
FIN
```

Ejemplo Selectivas (4)

➡ Diseñe

- a. un algoritmo que, dados 2 valores ingresados por el usuario, determine si son iguales o no.
- b. un algoritmo que determine si un carácter ingresado por el usuario es una minúscula o no.
- c. un algoritmo que determine si un número ingresado por el usuario es impar y positivo.
- d. Diseñe un algoritmo que calcule la raíces de una ecuación cuadrática. Controle que el cálculo sea posible.

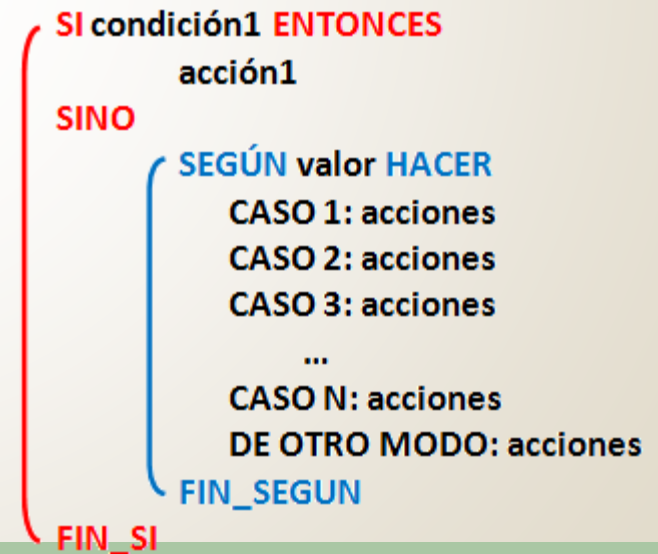
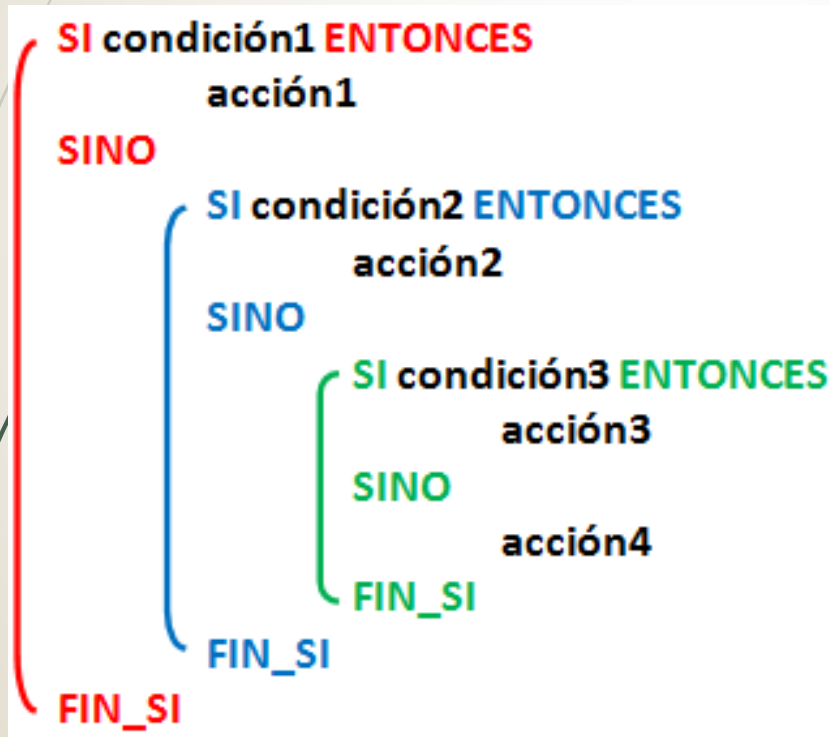


Anidamiento

- La PE permite combinar las estructuras de control básicas de manera flexible.
- Una estructura de control puede contener otras estructuras. Esto se llama *anidamiento de estructuras de control*.
- Reglas de anidamiento
 - la estructura interna debe quedar completamente incluida dentro de la externa, y
 - no puede existir solapamiento de estructuras.

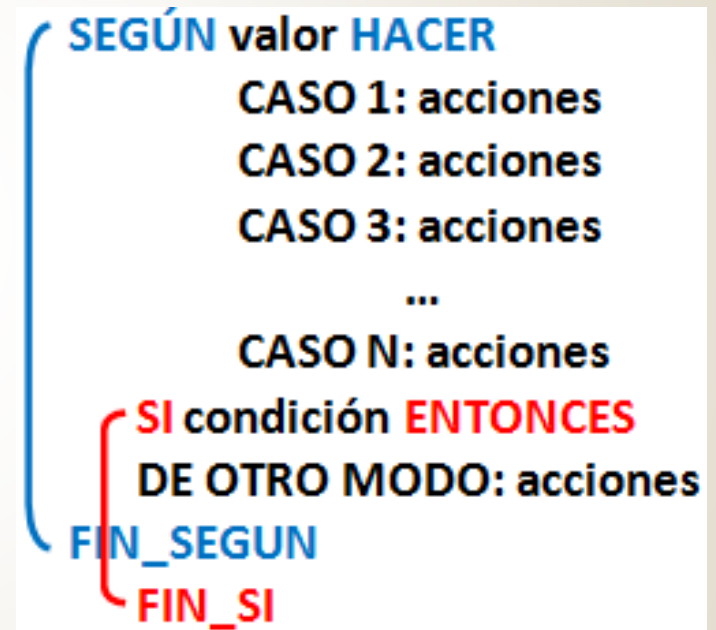
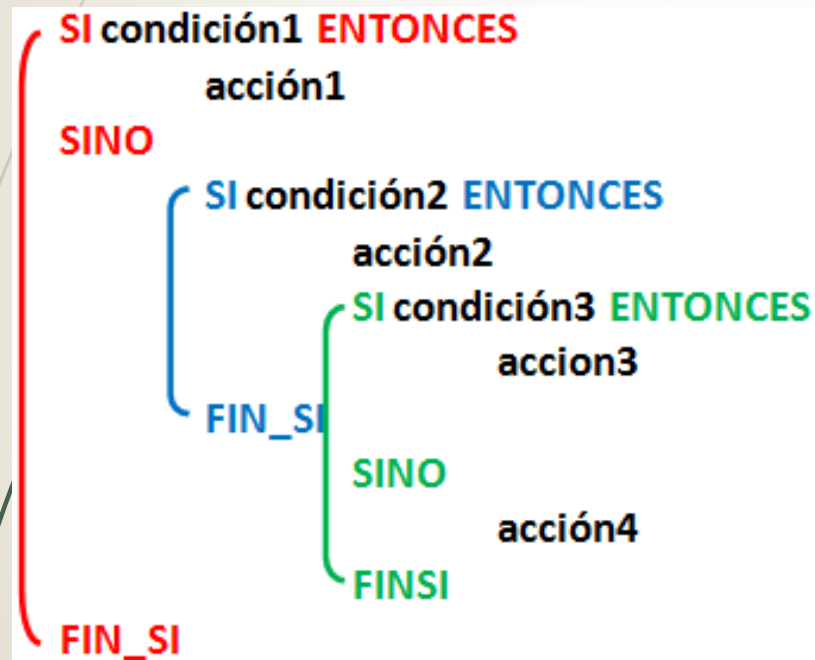
Anidamiento Válido

➤ Ejemplos



Anidamiento Inválido

➤ Ejemplos



Ejemplo Anidamiento

► Diseña

- a. un algoritmo que determine si 2 valores ingresados por el usuario son iguales o siendo distintos cuál es el mayor.
- b. un algoritmo que determine si un valor ingresado por el usuario es positivo, negativo o cero.
- c. un algoritmo que determine si un carácter ingresado por el usuario es minúscula, mayúscula o un símbolo especial.
- d. un algoritmo que, dados los lados de un triángulo, determine si se trata de un triángulo equilátero, isósceles o escaleno.



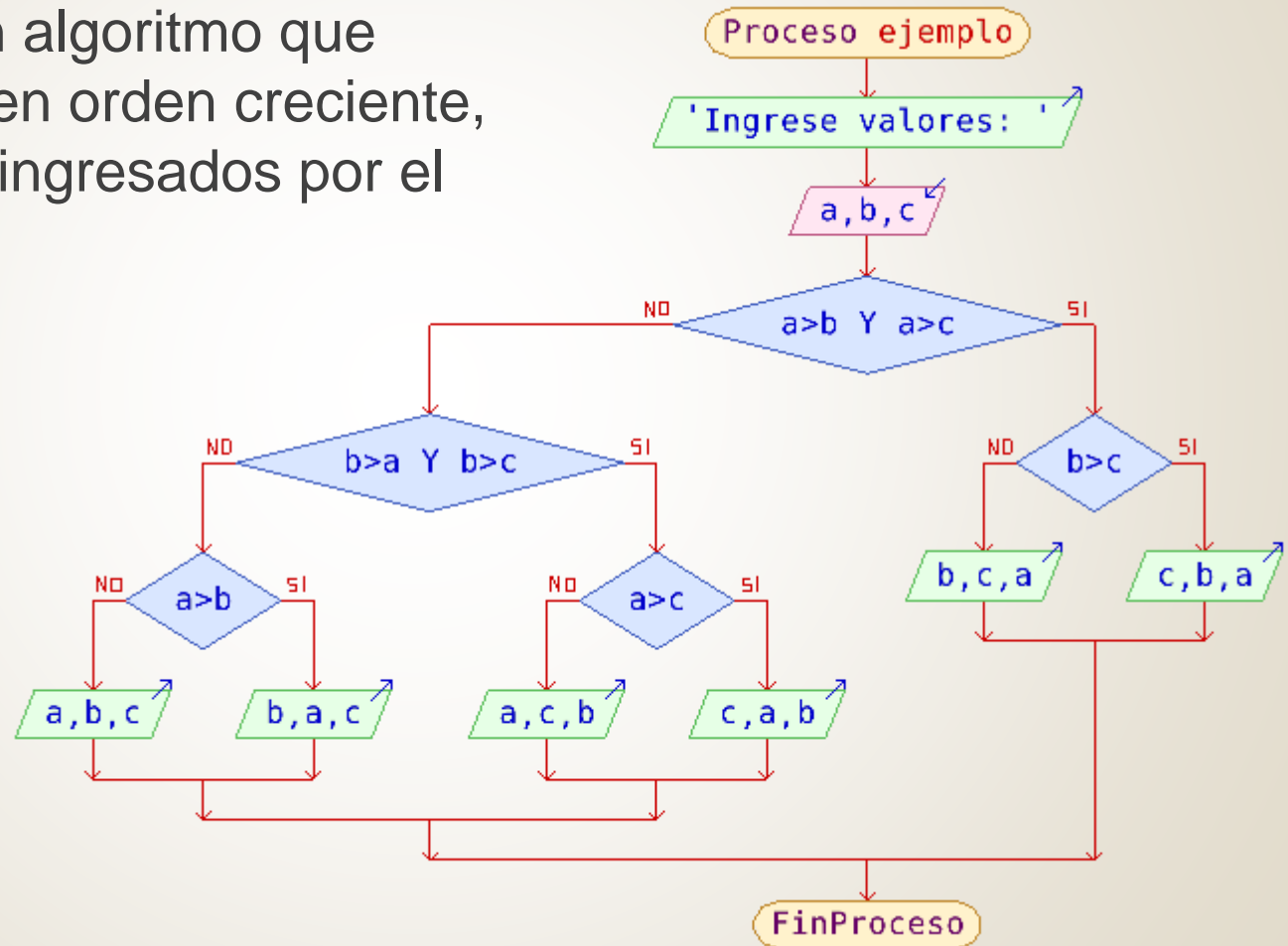
Prueba de Escritorio (1)

- La prueba de escritorio permite **comprobar, en tiempo de diseño**, el comportamiento de un algoritmo.
- Consiste en **analizar instrucción a instrucción** el algoritmo, registrando los cambios de las variables y condiciones.
- Es conveniente utilizar **datos representativos del problema** y también valores **de excepción o no esperados**.



Prueba de Escritorio (2)

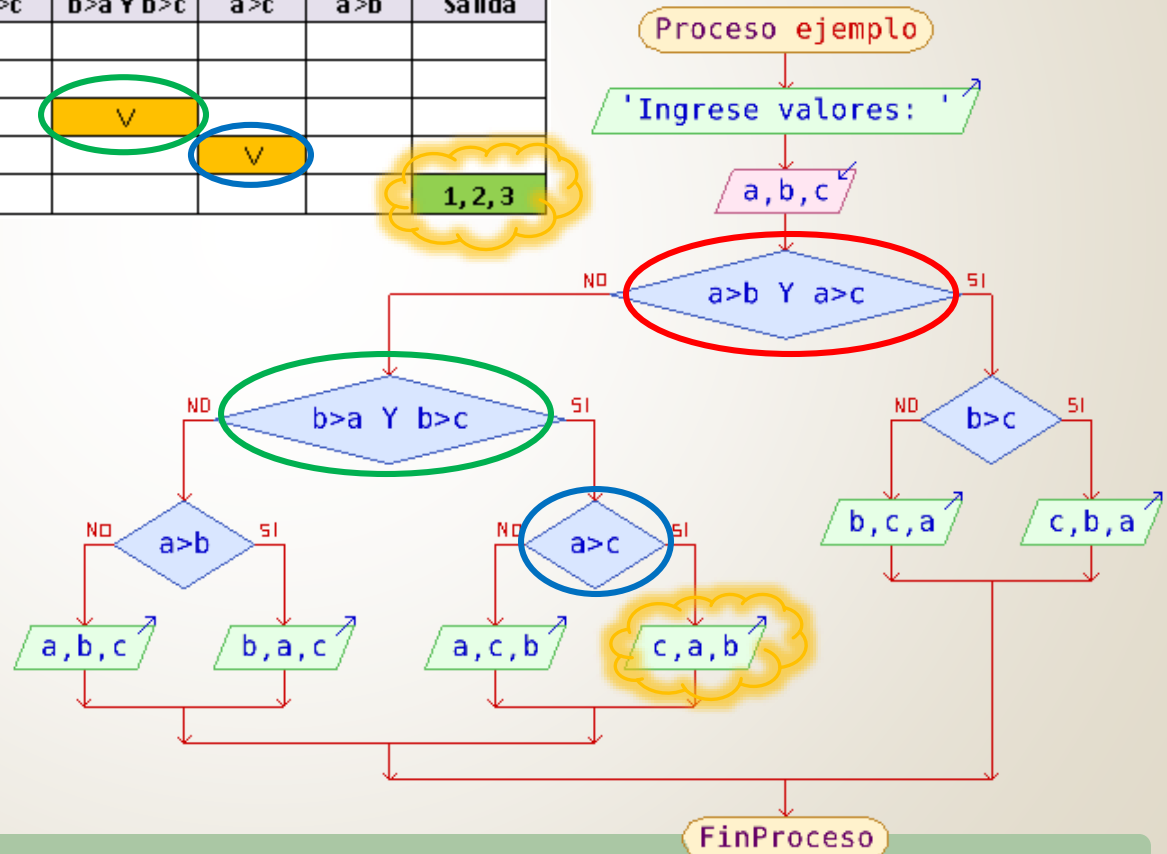
- Diseñe un algoritmo que muestre, en orden creciente, 3 valores ingresados por el usuario.



Prueba de Escritorio (3)

- Realice la prueba de escritorio para los valores $a=2$, $b=3$ $c=1$

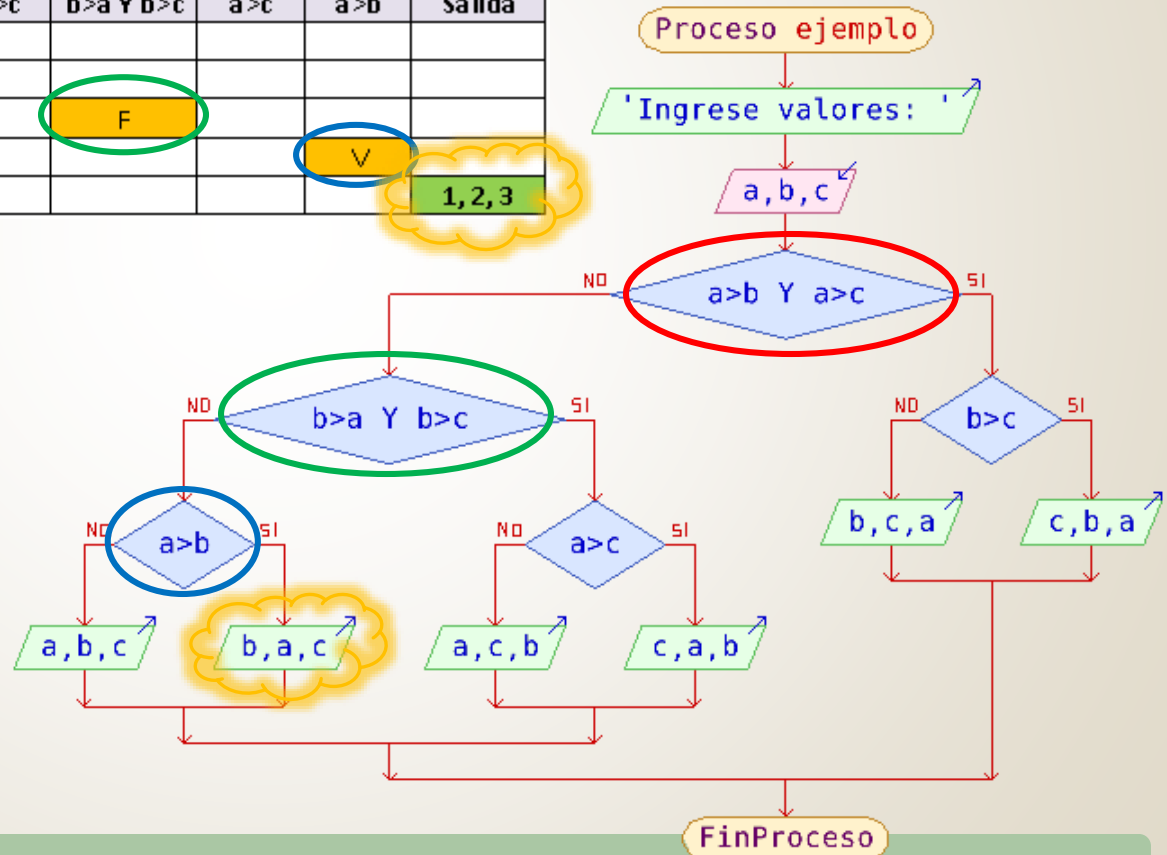
| Paso | a | b | c | $a > b \text{ Y } a > c$ | $b > c$ | $b > a \text{ Y } b > c$ | $a > c$ | $a > b$ | Salida |
|------|---|---|---|--------------------------|---------|--------------------------|---------|---------|--------|
| 1 | 2 | 3 | 1 | F | | | | | |
| 2 | | | | | V | | | | |
| 3 | | | | | | V | | | |
| 4 | | | | | | | V | | |
| 5 | | | | | | | | | 1,2,3 |



Prueba de Escritorio (4)

- Realice la prueba de escritorio para los valores $a=2$, $b=1$ $c=3$

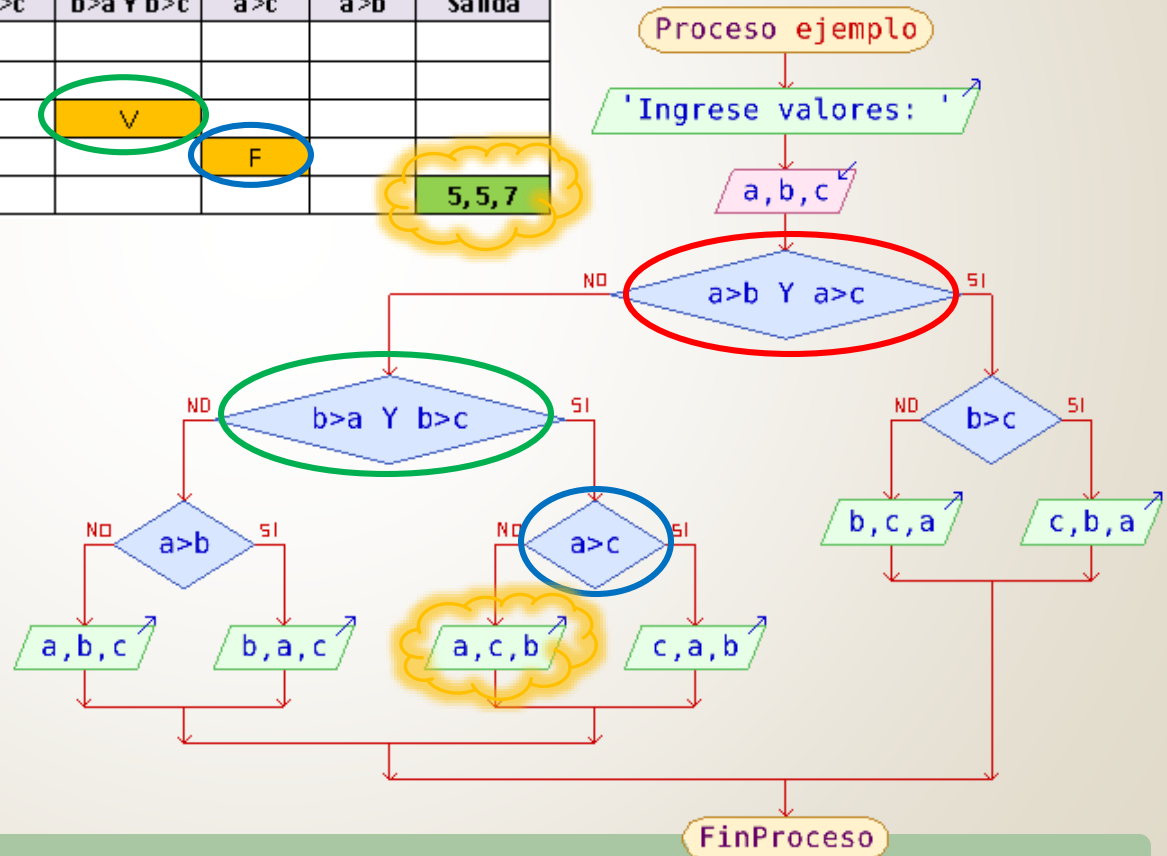
| Paso | a | b | c | $a > b \text{ Y } a > c$ | $b > c$ | $b > a \text{ Y } b > c$ | $a > c$ | $a > b$ | Salida |
|------|---|---|---|--------------------------|---------|--------------------------|---------|---------|--------|
| 1 | 2 | 1 | 3 | F | | | | | |
| 2 | | | | | | F | | | |
| 3 | | | | | | | | V | |
| 4 | | | | | | | | | |
| 5 | | | | | | | | | 1,2,3 |



Prueba de Escritorio (5)

- Realice la prueba de escritorio para los valores $a=5$, $b=7$ $c=5$

| Paso | a | b | c | $a > b \text{ Y } a > c$ | $b > c$ | $b > a \text{ Y } b > c$ | $a > c$ | $a > b$ | Salida |
|------|---|---|---|--------------------------|---------|--------------------------|---------|---------|--------|
| 1 | 5 | 7 | 5 | F | | | | | |
| 2 | | | | V | | | | | |
| 3 | | | | | | | | | |
| 4 | | | | | | | | | |
| 5 | | | | | | | | | 5,5,7 |



Bibliografía

- Sznajdleder, Pablo Augusto. Algoritmos a fondo. Alfaomega. 2012.
- López Román, Leobardo. Programación estructurada y orientada a objetos. Alfaomega. 2011.
- De Giusti, Armando *et al.* Algoritmos, datos y programas, conceptos básicos. Editorial Exacta, 1998.
- Joyanes Aguilar, Luis. Fundamentos de Programación. Mc Graw Hill. 1996.
- Joyanes Aguilar, Luis. Programación en Turbo Pascal. Mc Graw Hill. 1990.