

Tema 3: Introducción a la programación y Diagramas de Flujo

Informática
Grado en Ingeniería en Tecnologías
Industriales

Departamento de Ingeniería de Sistemas y Automática.
Escuela de Ingenieros. Universidad de Sevilla



Índice

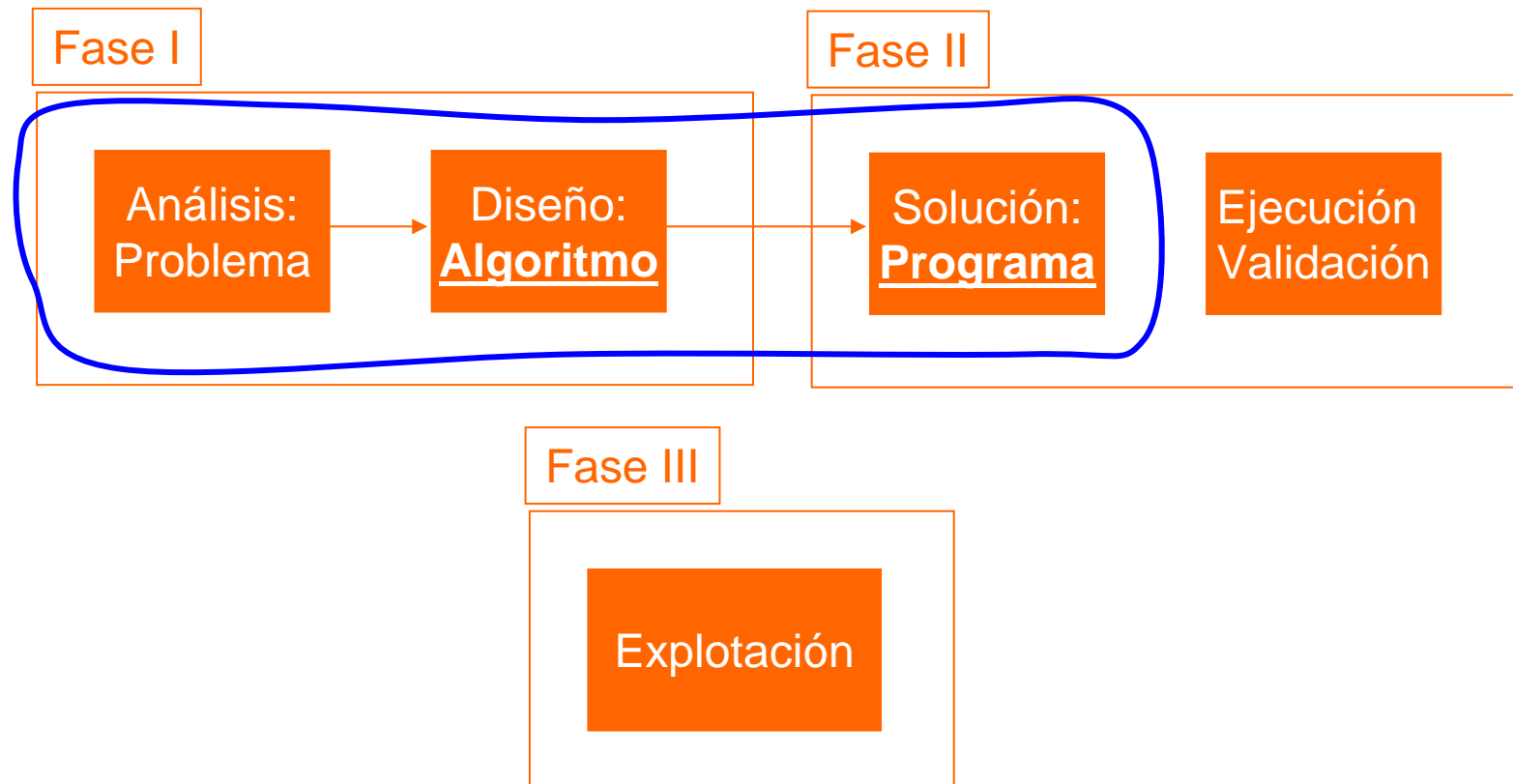
1. Objetivo
2. Definiciones: Algoritmo y programa
3. Tipos de programación: Programación estructurada
4. Herramientas para la realización de un algoritmo:
 - 4.1. Pseudocódigo
 - 4.2. Diagrama de Flujo
5. Tabla de Objetos
6. Programación estructurada: estructuras de control
7. Traza de un programa

1. Objetivo

- ❑ El objetivo principal del tema consiste en la realización de algoritmos mediante pseudocódigo y diagrama de flujo.

2. Definiciones: Algoritmo y Programa

Fases de la programación



2. Definiciones: Algoritmo y Programa

- ❑ Un **algoritmo** es un conjunto ordenado y finito de instrucciones que permite hallar la solución de un problema.
- ❑ Un **programa** es la codificación del algoritmo en algún lenguaje de programación o en lenguaje máquina.



Ejemplo de un algoritmo

- ❑ Problema: Receta para freír un huevo frito
- ❑ Algoritmo:
 1. Poner aceite en sartén
 2. Colocar sartén en fuego
 3. Romper el huevo haciendo caer el contenido en sartén
 4. Tirar cáscaras a la basura
 5. Poner sal en yema
 6. Si el huevo está sólido ir a 7, si no esperar
 7. Servir huevo, fregar sartén
 8. Fin

Ejemplo de algoritmo y programa

❑ **Problema:** Hallar el valor absoluto de un número x .

❑ **Algoritmo:**

1. Si x es positivo, el resultado es, $r \leftarrow x$
2. Si no, el resultado es $r \leftarrow -x$
3. Fin

❑ **Programa:**

```
int valor_absoluto(int x)
{
    if(x>0)
        r=x;
    else
        r=-x;
    return x;
}
```

Partes de un algoritmo

- ❑ Un algoritmo utiliza un conjunto de datos de entrada y proporciona unos datos de salida.
- ❑ **ENTRADA:** Corresponde a los datos que requiere el proceso para ofrecer los resultados esperados.
- ❑ **PROCESO:** Pasos necesarios para obtener la solución del problema o la situación planteada.
- ❑ **SALIDA:** Datos presentados por el proceso como solución, resultado.

3. Tipos de programación

- ❑ Programación imperativa o estructurada:
C, PASCAL, FORTRAN
- ❑ Programación Orientada a Objetos: Visual
C++, Java
- ❑ Programación funcional: LISP
- ❑ Programación lógica: PROLOG

3. Programación Estructurada

- ❑ Programación estructurada: consiste en un conjunto de reglas para escribir programas de tal manera que sean legibles y fáciles de modificar.
- ❑ Reglas a seguir para la programación estructurada:
 - a) Características de algoritmos estructurados
 - b) ¿Cómo construir un algoritmo estructurado?
 - c) Estructuras algorítmicas estructuradas

a) Características de un algoritmo estructurado

- ❑ **Finito:** El algoritmo debe tener un número finito de pasos.
- ❑ **Eficientes:** Deben ocupar la mínima memoria y minimizar el tiempo de ejecución.
- ❑ **Legibles:** El texto que lo describe debe ser claro, de forma que permita entenderlo y leerlo fácilmente.
- ❑ **Modificables:** Estarán diseñados de modo que sus posteriores modificaciones sean fáciles de realizar, incluso por programadores diferentes a sus propios autores.

a) Características de un algoritmo estructurado

- ❑ **Modulares:** La filosofía utilizada para su diseño debe favorecer la división del problema en módulos pequeños.
- ❑ **Único punto de entrada, único punto de salida:** A los algoritmos y a los módulos que lo integran, se entra por un solo punto (inicio) y se sale por un solo punto (fin)

b) ¿Cómo construir algoritmos estructurados?

- ☐ Definición y análisis del problema: datos de entrada y salida (resultados)
- ☐ Aplicar la técnica de: “divide y vencerás”, que consiste en descomponer el problema en subproblemas más sencillos
- ☐ Resolución de los subproblemas: realización de los algoritmos correspondientes a los subproblemas
- ☐ Depurar (prueba de validez) el algoritmo resultante

¡A PROGRAMAR SE APRENDE PROGRAMANDO!

c) Estructuras algorítmicas o de control

- ❑ **Secuenciales:** cada acción se realiza una sola vez y en un determinado orden
- ❑ **Condicionales (selectivas):** permiten seleccionar una acción a realizar entre varias alternativas
- ❑ **Iterativas (repetitivas):** una determinada acción se realiza más de una vez

4. Herramientas para la realización de algoritmos

- ❑ Existen diferentes métodos para representar un algoritmo, los procedimientos más habituales son:
 - ❑ Pseudocódigo (herramienta no gráfica)
 - ❑ Diagrama de flujo (herramienta gráfica)



4.1. Pseudocódigo

- Un **pseudocódigo** es una forma de representar un algoritmo basándose en el lenguaje natural.

Ejemplo de pseudocódigo

❑ **Problema:** leer dos números enteros y escribir la suma.

❑ **Pseudocódigo:**

1. Leer primer sumando, a .
2. Leer segundo sumando, b .
3. Hallar la suma de los sumandos, $r \leftarrow a+b$.
4. Escribir r .
5. Fin

Elementos básicos de un algoritmo

- ❑ Datos de diferente tipo: números reales, enteros, caracteres,...
- ❑ Las instrucciones que los procesan: asignación, operaciones aritméticas, lógicas,...

Ejemplo de pseudocódigo

❑ **Problema:** leer dos números enteros y escribir la suma.

❑ **Pseudocódigo:**

1. Leer primer sumando, a.
2. Leer segundo sumando, b.
3. Hallar la suma de los sumandos, $r \leftarrow a + b$.
4. Escribir r.
5. Fin

Datos de entrada
(números enteros)

Asignación

Datos de salida
Resultado
(número entero)

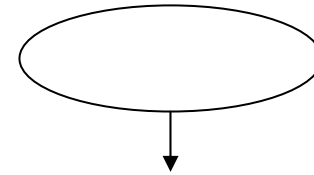
Suma aritmética

4.2. Diagramas de flujo

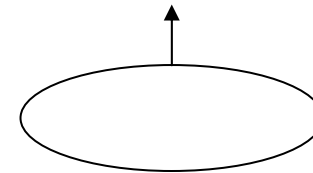
- ❑ Los Diagramas de Flujo son herramientas gráficas para representar algoritmos.
- ❑ Están formados por una serie de símbolos, que tienen al menos una flecha que viene del paso anterior y otra que va al paso siguiente.
- ❑ Los símbolos representan distintas acciones: lectura/escritura, principio, fin, salto...

Símbolos de un diagrama de flujo

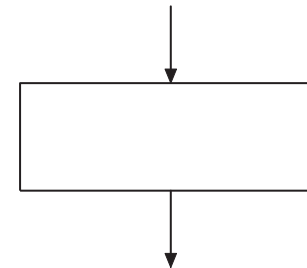
❑ **Comienzo de bloque:**



❑ **Fin de bloque:**

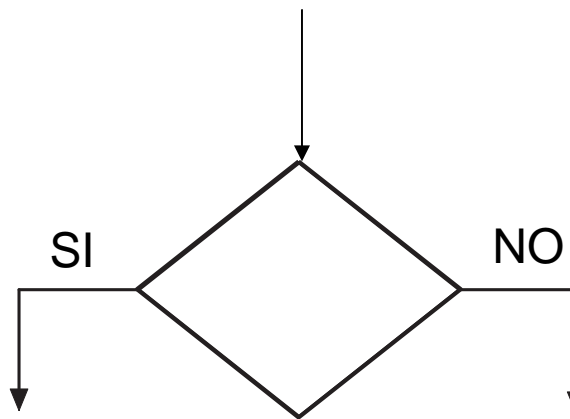


❑ **Proceso:** asignaciones,
operaciones



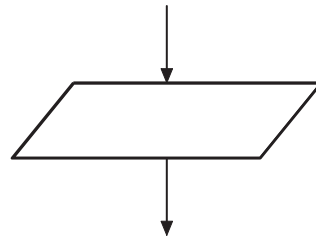
Símbolos de un diagrama de flujo

- ❑ **Bifurcación:** representa una decisión. En su interior se almacena una condición y dependiendo del resultado de la evaluación de la misma se sigue una dirección u otra. Se utiliza en las estructuras selectiva e iterativas



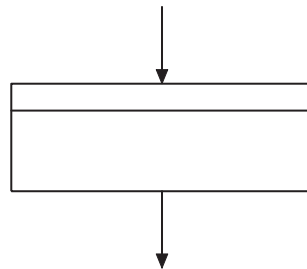
Símbolos de un diagrama de flujo

- ❑ **Entrada y salida de datos:** Se utiliza para representar la introducción de datos de entrada (lectura) y para la impresión de un resultado (salida)



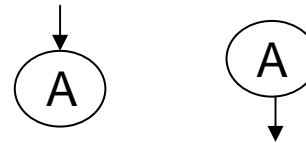
Símbolos de un diagrama de flujo

- ❑ **Módulo:** la operación es realizada por un bloque (DF) que se detalla en otro lugar. No afecta a la codificación.

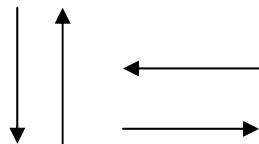


Símbolos de un diagrama de flujo

❑ **Conector:** símbolo utilizado para expresar conexión de DF



❑ **Líneas de flujo o dirección:** Expresan la dirección del flujo del diagrama



Reglas para construcción de diagramas de flujo

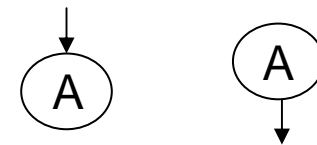
1. Todo diagrama de flujo debe tener un inicio y un fin.
2. Las líneas utilizadas para indicar la dirección de flujo del diagrama deben ser rectas, verticales y horizontales. No deben ser inclinadas y tampoco se deben cruzar.
3. Las líneas utilizadas para indicar la dirección de flujo del diagrama deben estar conectadas.

Reglas para construcción de diagramas de flujo

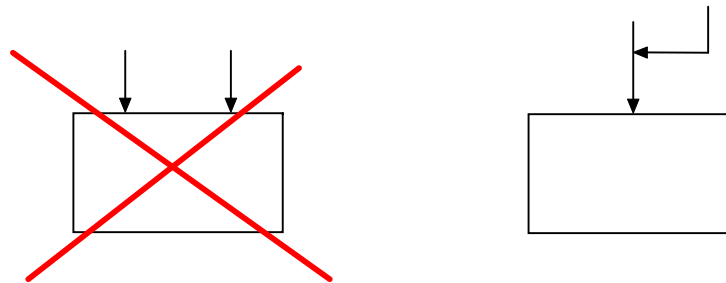
4. El diagrama de flujo debe ser construido de arriba hacia abajo (top-down)
5. La notación utilizada en el diagrama de flujo debe ser independiente del lenguaje de programación. La solución presentada en el D.F. puede escribirse posterior y fácilmente en cualquier lenguaje de programación
6. Es conveniente cuando realizamos una tarea compleja poner comentarios que expresen o ayuden a entender lo que hicimos

Reglas para construcción de diagramas de flujo

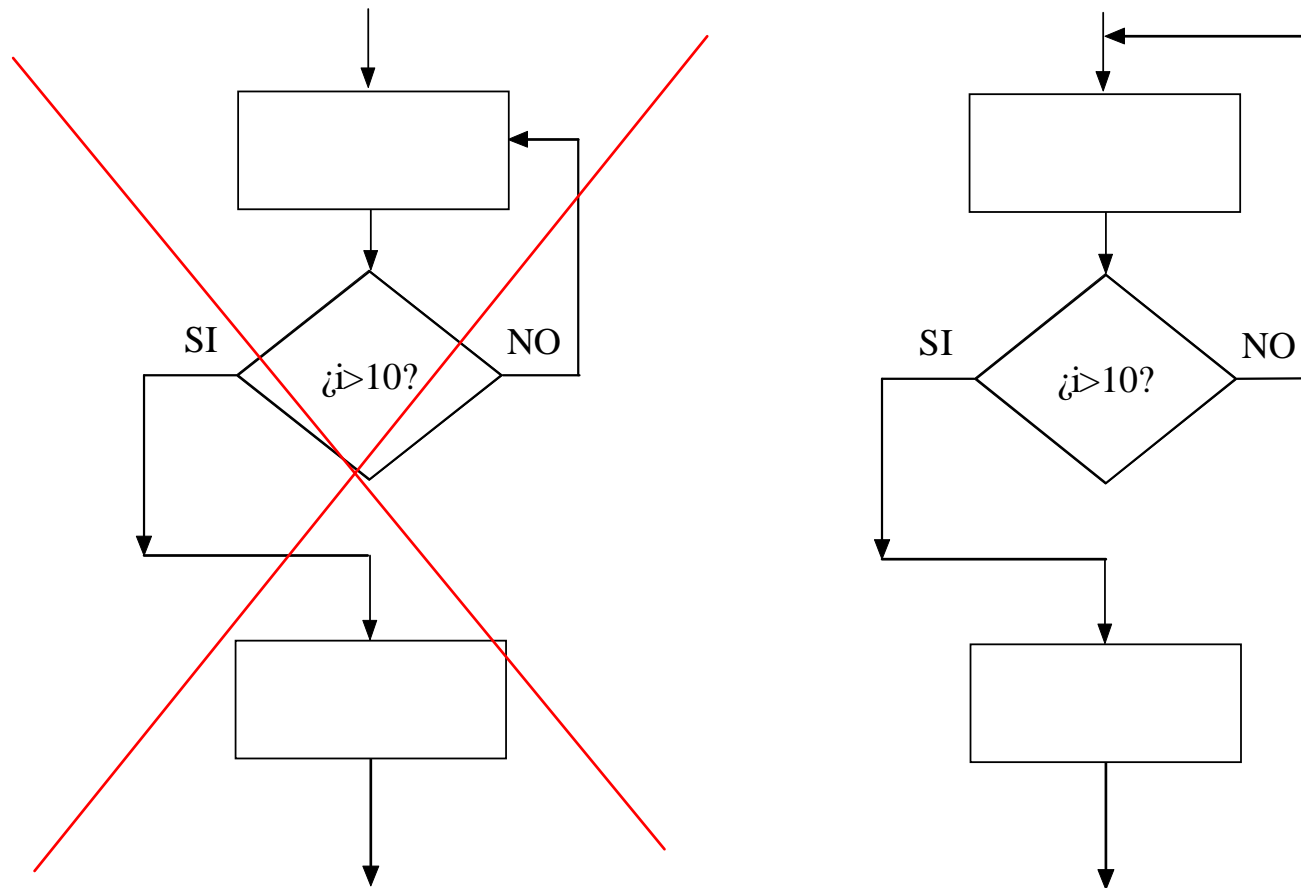
7. Si el DF requiere más de una hoja para su construcción, debemos utilizar los conectores adecuados y enumerar las páginas convenientemente.



8. No puede llegar más de una línea a un símbolo



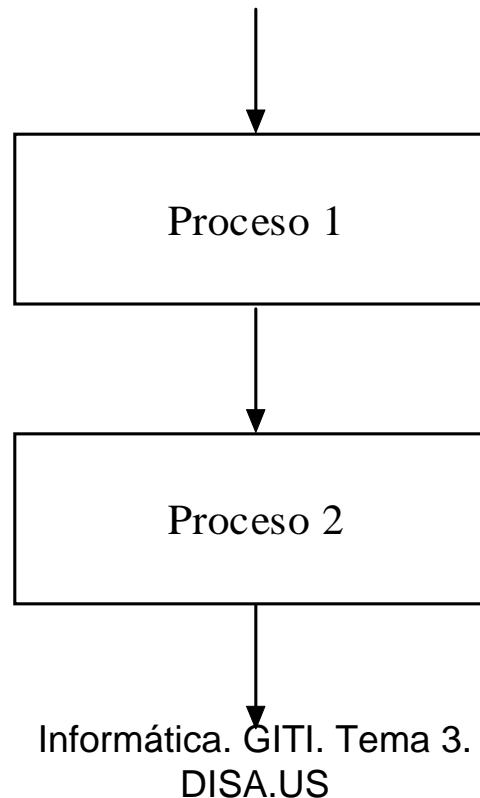
Reglas para construcción de diagramas de flujo



Reglas para construcción de diagramas de flujo

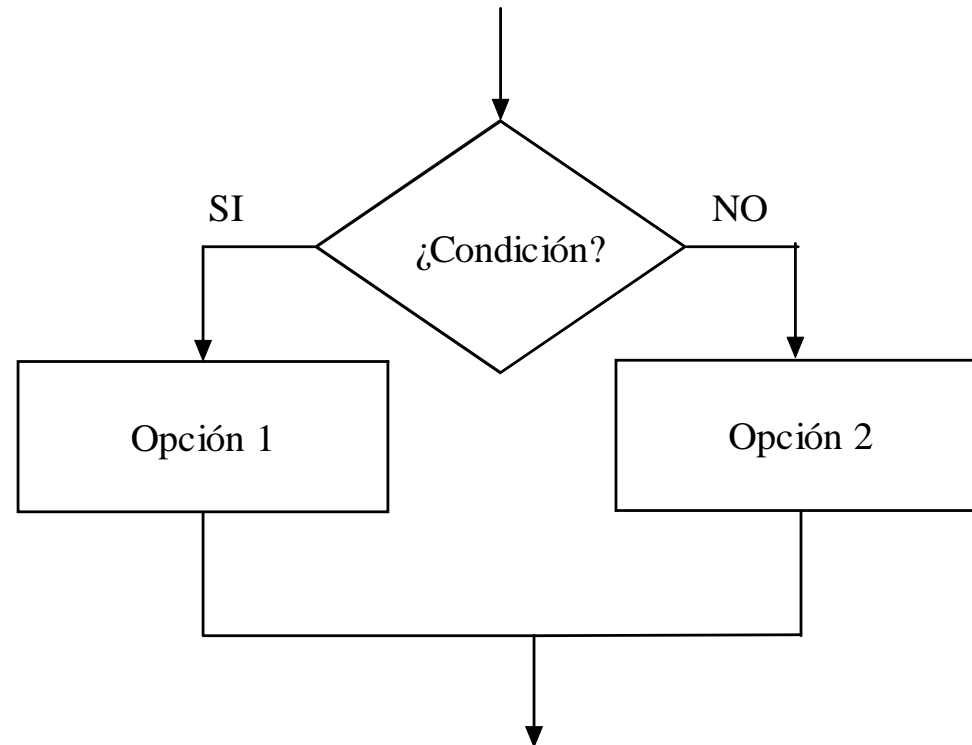
9 Utilizar sólo los bloques siguientes para realizar DF.

❑ Estructura secuencial:



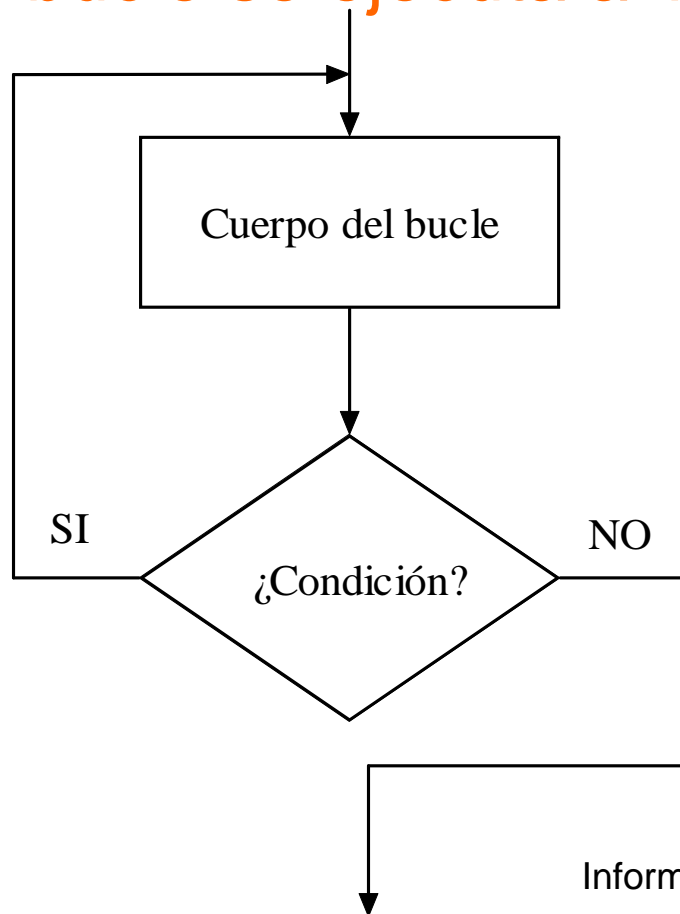
Reglas para construcción de diagramas de flujo

❑ Estructura selectiva



Reglas para construcción de diagramas de flujo

- ❑ Estructura iterativa: salida en cola (el cuerpo del bucle se ejecuta al menos una vez)

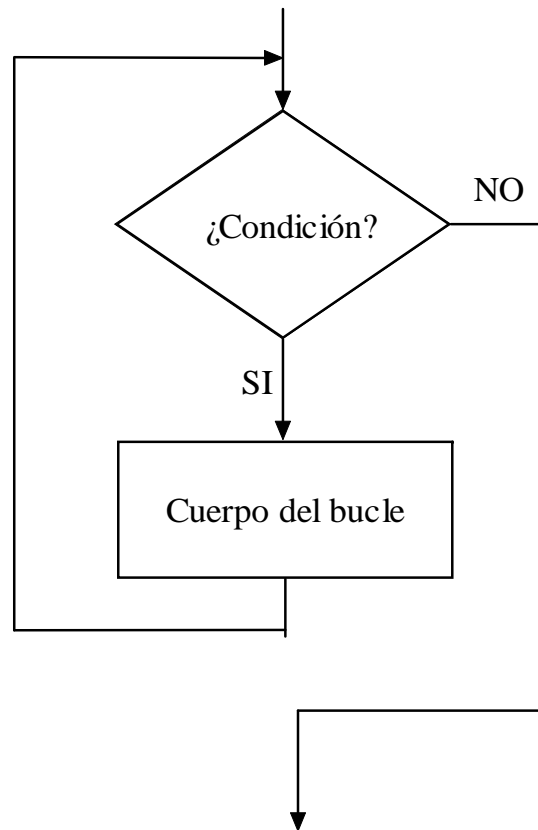


Proceso iterativo o bucle:

- **Cuerpo del bucle:**
Conjunto de operaciones que se repiten
- **Condición** de salida o parada

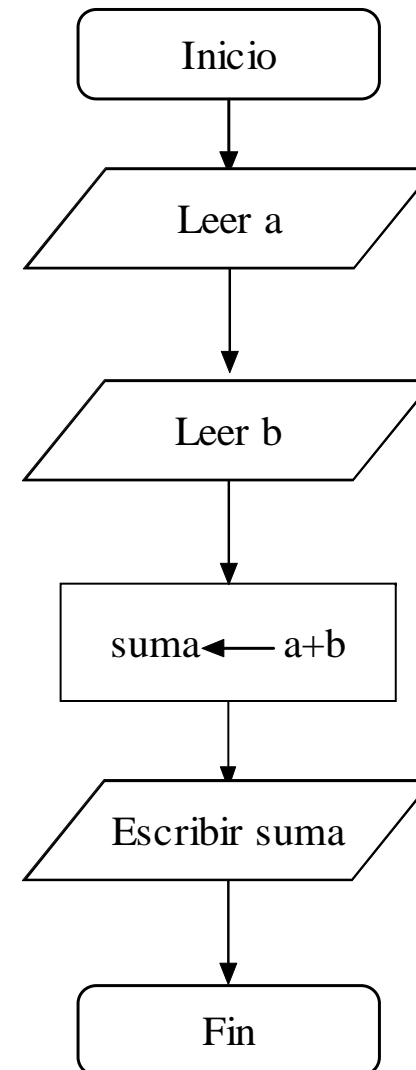
Reglas para construcción de diagramas de flujo

❑ Estructura iterativa: salida en cabeza



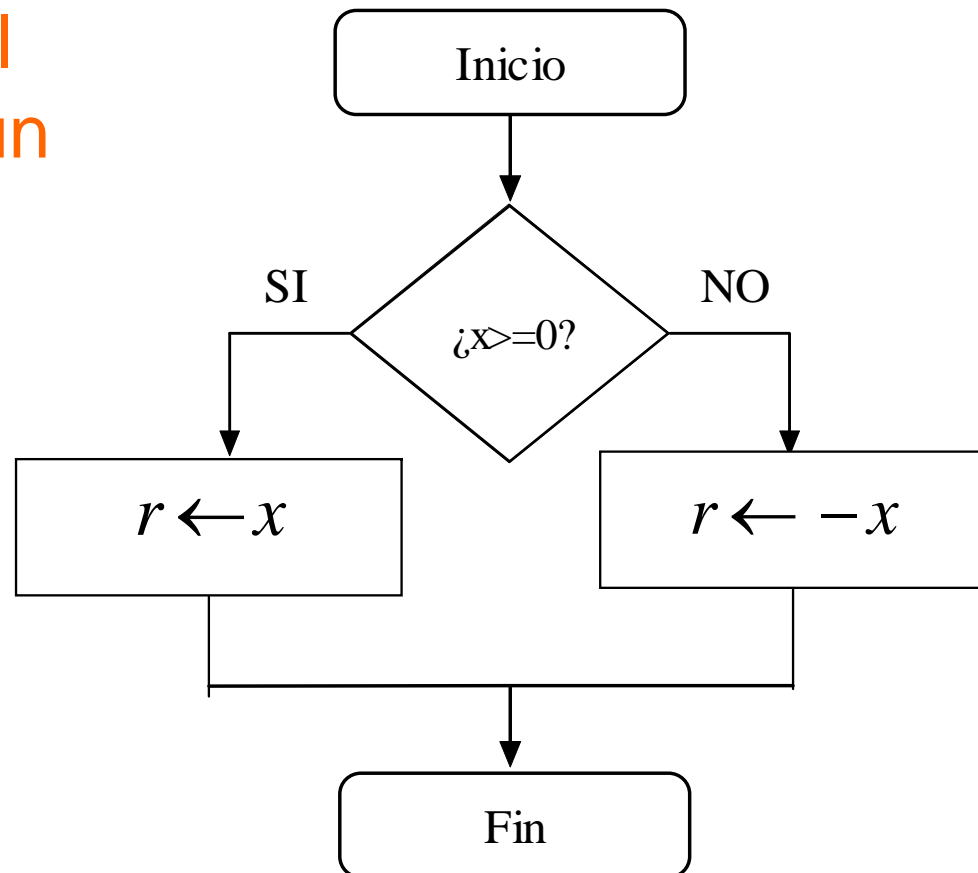
Ejemplo 1: algoritmo con estructura secuencial

- ❑ Problema: leer dos números enteros y escribir la suma.



Ejemplo 2: algoritmo con estructura selectiva

❑ Problema: Hallar el valor absoluto de un número x .



Índice

1. Objetivo
2. Definiciones: Algoritmo y programa
3. Tipos de programación: Programación estructurada
4. Herramientas para la realización de un algoritmo:
 - 4.1. Pseudocódigo
 - 4.2. Diagrama de Flujo
- 5. Tabla de Objetos**
6. Programación estructurada: estructuras de control
7. Traza de un programa

5.1. Objeto: identificador, valor y tipo

□ Un objeto (dato), es información que utiliza un algoritmo para resolver un problema.

□ Ejemplo:

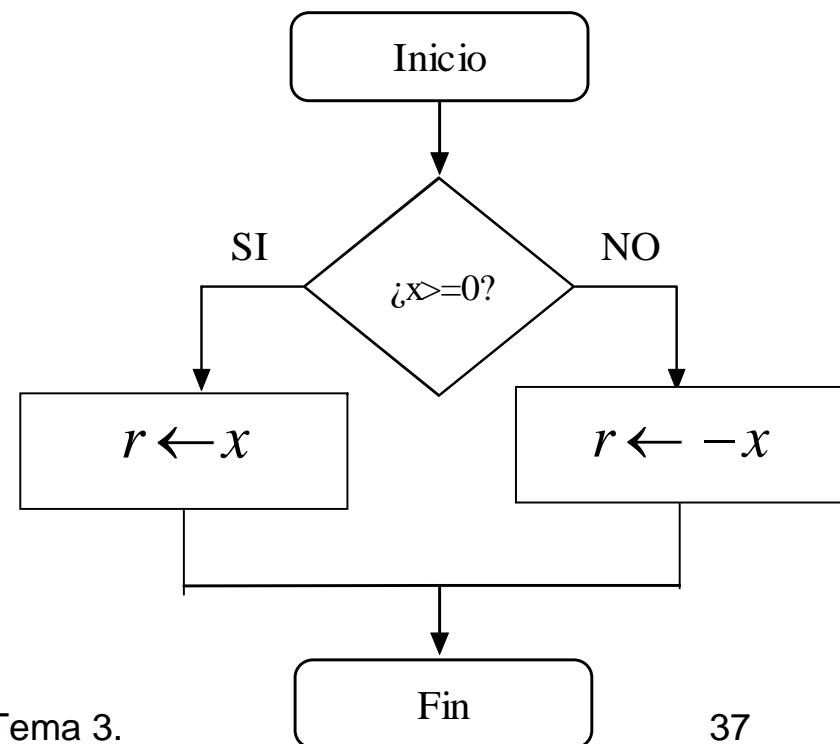
x: dato de entrada. Variable.

Número entero

r: dato de salida. Variable.

Número entero

0: constante entera



Conceptos fundamentales

- ❑ Los datos a procesar por una computadora, deben almacenarse en casillas o celdas de memoria para su posterior utilización.
- ❑ Estas casillas de memoria tienen un nombre que permite su identificación: **Identificador, nombre o etiqueta**
- ❑ El valor de las celdas de memoria puede ser **variable o constante**

Constantes

- ❑ Las constantes son datos que no cambian durante la ejecución
- ❑ Para nombrar las constantes utilizamos los identificadores
- ❑ Existen tantos tipos de constante como tipos de datos

Variables

- ❑ Las variables son objetos que pueden cambiar su valor durante la ejecución
- ❑ Para nombrar las variables utilizamos los identificadores
- ❑ Existen tantos tipos de variables como tipos de datos

Los nombres, identificadores de las constantes y variables deben ser representativos de la función que cumplen el algoritmo. Ej: suma, contador, media

Tipos de datos

☐ Simples:

- ☐ Ocupan sólo una casilla de memoria
- ☐ Con un identificador se hace referencia a un único valor a la vez
 - ☐ Una única letra, un único valor real...

☐ Complejos o Estructurados:

- ☐ Tiene varios componentes, cada uno de estos componentes puede ser a su vez un dato simple.
- ☐ Con un identificador se hace referencia a un grupo de celdas de memoria, es decir, se usa un único identificador para un conjunto de valores
- ☐ Vectores y matrices (de enteros, reales, lógicos, caracteres)

Tipos de datos simples

- ❑ **Enteros:** se utilizan para representar números enteros, positivos o negativos. Ej: 128, -45,...
- ❑ **Reales:** se utilizan para representar números reales, positivos o negativos. Ej: 7.5, -37.675,...
- ❑ **Lógicos:** se utilizan para representar valores lógicos o booleanos. Son datos que sólo pueden tomar dos valores: verdadero o falso
 - ❑ V o F
- ❑ **Caracteres:**
 - ❑ Letras del alfabeto mayúsculas o minúsculas (son diferentes)
 - ❑ Caracteres numéricos: dígitos del 0 al 9
 - ❑ Caracteres especiales: signos de puntuación, guiones, paréntesis, asteriscos...
 - ❑ El valor se indica entre comillas simples: ‘ ‘
 - ❑ Ej.: x <- ‘a’, letra <- ‘1’, x <- 1

Objeto: Identificador, valor y tipo

☐ Identificador, nombre o etiqueta:

- ☐ Cada objeto tiene un único nombre que lo identifica.

☐ Valor:

- ☐ El valor del objeto puede variar durante el algoritmo (variable) o permanecer constante (constante).

☐ Tipo:

- ☐ Los objetos pueden ser de diferentes tipos (tipos simples y complejos)

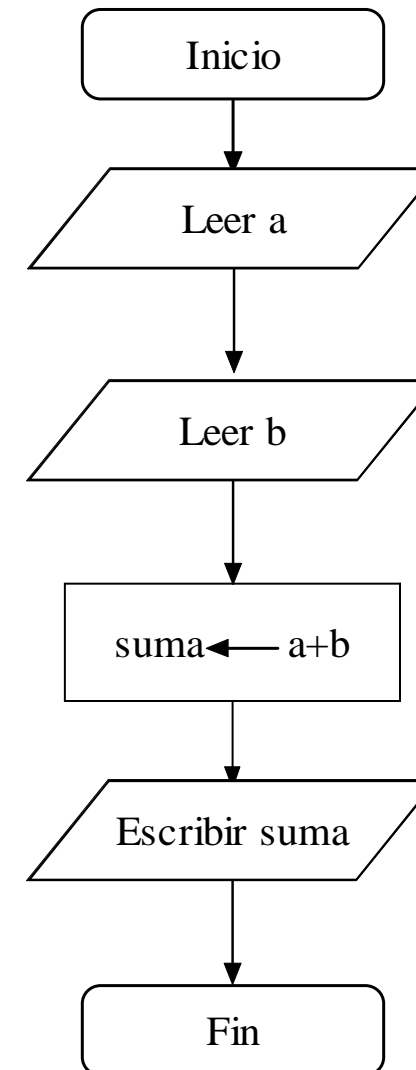
5.2. Tabla de objetos

- ❑ Una tabla de objeto es una tabla que refleja todos los objetos existentes en un algoritmo.
- ❑ En la tabla de objetos se indica el identificador, nombre o etiqueta, el valor (constante o variable) y el tipo de todos los objetos que aparecen en un algoritmo.

Objeto	Identificador o nombre	Valor	Tipo
--------	---------------------------	-------	------

Ej.:Tabla de objetos normalizada I.
Problema: leer dos números enteros y escribir la suma.

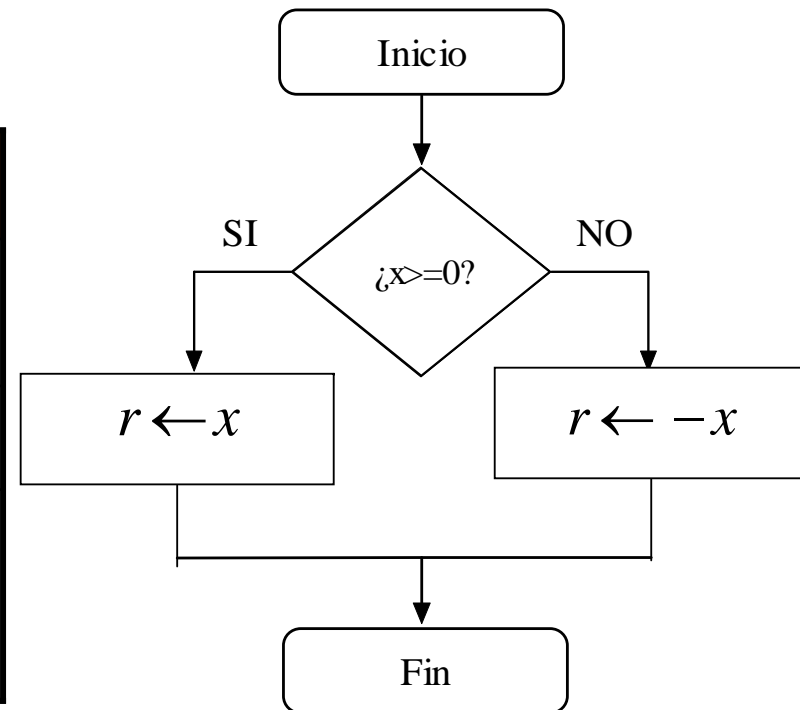
Objeto	Nombre	Valor	Tipo
Primer sumando	a	Variable	Entera
Segundo sumando	b	Variable	Entera
Resultado de la suma	suma	Variable	Entera



Ej.: Tabla de objetos normalizada II.

Problema: Hallar el valor absoluto de un número x .

Objeto	Nombre	Valor	Tipo
Dato entrada	x	Variable	Entera
Cero	0	Constante	Entera
Resultado, valor absoluto	r	Variable	Entera



5.3. Operadores y Expresiones

- ❑ Las siguientes operaciones y expresiones son las únicas permitidas en la realización de un algoritmo:
 - a) Operaciones aritméticas
 - b) Operaciones lógicas
 - c) Operaciones relacionales
 - d) Asignación
 - e) Expresiones

a) Operaciones aritméticas

- ❑ Permiten operar con valores enteros o reales para obtener un resultado también entero o real
- ❑ Para poder realizar operaciones aritméticas necesitamos operadores aritméticos

a) Operaciones aritméticas

Operación	Operador	Ejemplo
Suma	+	$a+b$
Resta	-	$a-b$
Unario	- (cambio de signo)	$-a$
Multiplicación	*	$a*b$
División	/	a/b
Módulo (resto de división entera)	%	$a\%b$



¡Definida solo para operandos enteros!

a) Operaciones aritméticas

- ❑ La división entera es diferente a la división de reales
- ❑ Si los dos operandos son enteros el resultado es un entero
- ❑ Si los dos operandos son reales el resultado es un real
- ❑ Si uno de los dos operandos es real el resultado es real

Ej: Para a y b enteros con valores: a=9 y b=2
Para c y d reales con valores: c=9 y d=2


Operación	Resultado	Tipo Resultado
a/b	4	entero
c/d	4.5	real
a/d	4.5	real
a%b	1	entero

Precedencia o prioridad

- ❑ La precedencia o prioridad de un operador determina el orden de aplicación de los operadores de una expresión.
- ❑ Si tenemos en una expresión más de un operador, debemos aplicar primero el de mayor prioridad, resolver esa operación y así sucesivamente.
- ❑ El operador $()$ es un operador asociativo que tiene la prioridad más alta.

Precedencia operadores aritméticos

- ❑ Se evalúan primero las expresiones entre paréntesis. Si las subexpresiones se encuentran anidadas por paréntesis, primero se evalúan aquéllas que se encuentran en el último nivel de anidamiento.
- ❑ Los operadores aritméticos se aplican teniendo en cuenta la precedencia y de izquierda a derecha.

Operador	
-	 MAYOR PRIORIDAD
*, /, %	
+, -	
	MENOR PRIORIDAD

b) Operaciones lógicas

- ❑ Permiten operar con valores lógicos para obtener un resultado también lógico

Operación	Operador	Ejemplo
Negación	NO	NO a
Conjunción	Y	A Y b
Disyunción	O	A O b



MAYOR P.

MENOR P.

b) Operaciones lógicas

NO a

Valor a	Resultado
V	F
F	V

a Y b

Valor a	Valor b	Resultado
V	V	V
V	F	F
F	V	F
F	F	F

a O b

Valor a	Valor b	Resultado
V	V	V
V	F	V
F	V	V
F	F	F

c) Operaciones relacionales

- ☐ Permiten comparar dos datos del mismo tipo para obtener un resultado lógico: V o F
- ☐ Los datos pueden ser variables o constantes de tipo numérico o carácter

c) Operaciones relacionales

Operación	Operador	Ejemplo	Resultado
Igual que	=	'h'='H'	F
Distinto de	<>	'a'<>'b'	V
Menor que	<	7<15	V
Mayor que	>	22.5>11.6	V
Menor o igual que	<=	15<=15	V
Mayor o igual que	>=	35>=40	F

d) Asignación

- ❑ Sintaxis: 'variable' <- 'expresión'
- ❑ Primero se evalúa la expresión
- ❑ Después se asigna el valor a la variable
 - ❑ El valor anterior de la variable se pierde/olvida

Precedencia operadores: aritméticos, lógicos, relacionales y asignación

Operadores
()
NO, -
*, /, %
+, -
<, <=, >, >=
=, <>
Y
O
<- Asignación

MAYOR PRIORIDAD

MENOR PRIORIDAD

e) Expresiones

- ❑ Una expresión es cualquier combinación de operadores variables y constantes.
- ❑ Las expresiones sirven para manipular los objetos.
- ❑ Una expresión devuelve un valor de un tipo determinado
- ❑ Ejemplos:
 - ❑ $a + 23 * b - z$
 - ❑ NO x
 - ❑ $a > b$ Y $a < c$
 - ❑ $(z \leq 8)$ Y NO $(k - 6 * h)$

e) Ejemplos expresiones

$$9 + 7 * 8 - 36 / 5$$

$$9 + 56 - 36 / 5$$

$$9 + 56 - 7 \text{ (división entera)}$$

$$65 - 7$$

$$58$$

$$8 \% 2 * 3 + 1$$

$$8 \% 6 + 1$$

$$2 + 1$$

$$3$$

$$2 * 5 + 7.8 / 4 \leq 2.5 / 6$$

$$10 + 7.8 / 4 \leq 2.5 / 6$$

$$10 + 1.95 \leq 0.41$$

$$11.95 \leq 0.41$$

F

Operadores aritméticos, lógicos y relacionales

- ❑ **NOTA**: Los operadores aritméticos, lógicos y relacionales mostrados en las transparencias son los únicos permitidos para la realización de algoritmos. Excepto en casos particulares que se indique lo contrario

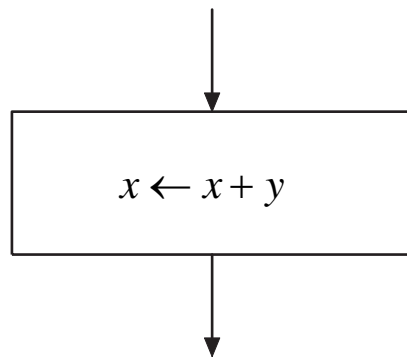
6. Estructuras de control de flujo (Programación Estructurada)

- ☐ Bloque/sentencia de asignación
- ☐ Bloque/sentencia de entrada
- ☐ Bloque/sentencia de salida
- ☐ Inicio, fin y módulo
- ☐ Estructuras de control:
 - ☐ Secuencial
 - ☐ Selectiva
 - ☐ Iterativa

Bloque/sentencia de asignación

❑ Sintaxis: 'variable' <- 'expresión'

❑ Símbolo



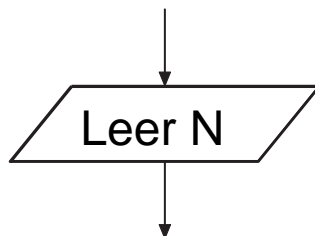
- En cada símbolo de proceso de DF pueden introducirse varias asignaciones.
- Al llegar el flujo a un bloque de asignaciones se realizan cada una de ellas, en el orden en el que aparecen y posteriormente se activa el bloque indicado por la flecha saliente.

Bloque/sentencia de entrada

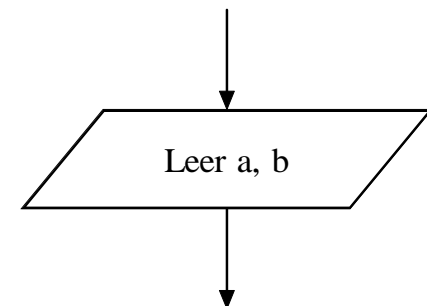
❑ Sintaxis: Leer 'variable'

- ❑ Se recibe un único valor (por teclado)
- ❑ Se asigna el valor a la variable
 - ❑ El valor anterior de la variable se pierde/olvida

❑ Símbolo

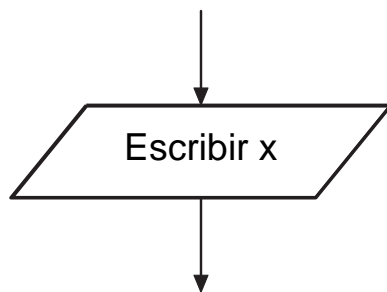


- En cada símbolo de entrada de DF pueden introducirse varias lecturas.
- Al llegar el flujo a un bloque de entrada se realizan todas las lecturas y posteriormente se activa el bloque indicado por la flecha saliente.

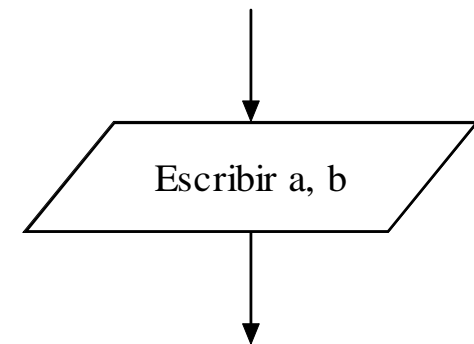


Bloque/sentencia de salida

- ❑ Sintaxis: Escribir 'variable'
 - ❑ Se devuelve el valor de la variable
 - ❑ Se escribe en pantalla
 - ❑ Forma de devolver resultados de un algoritmo*
- ❑ Símbolo



- En cada símbolo de salida de DF pueden introducirse varias escrituras.
- Al llegar el flujo a un bloque de salida se realizan todas las escrituras y posteriormente se activa el bloque indicado por la flecha saliente

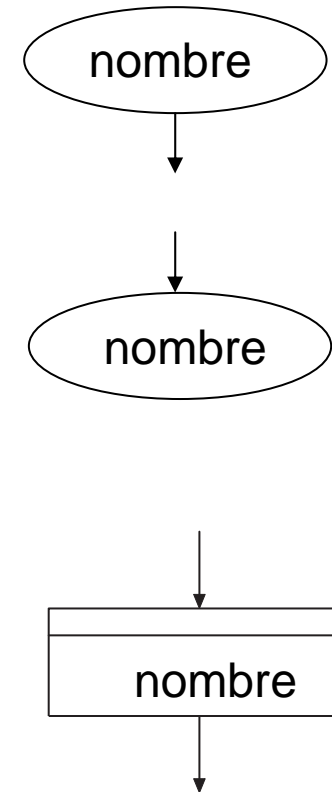


Inicio, fin y módulo

❑ Inicio

❑ Fin

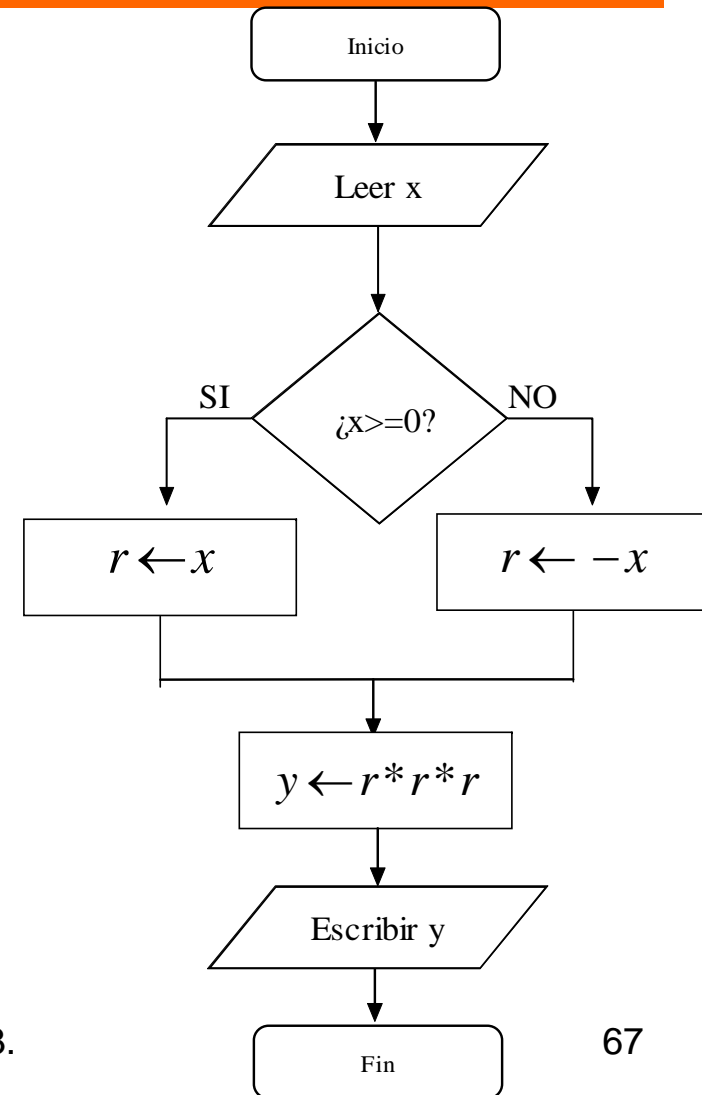
❑ **Módulo:** la operación es realizada por un bloque (DF) que se detalla en otro lugar. No afecta a la codificación.



Uso del símbolo módulo:

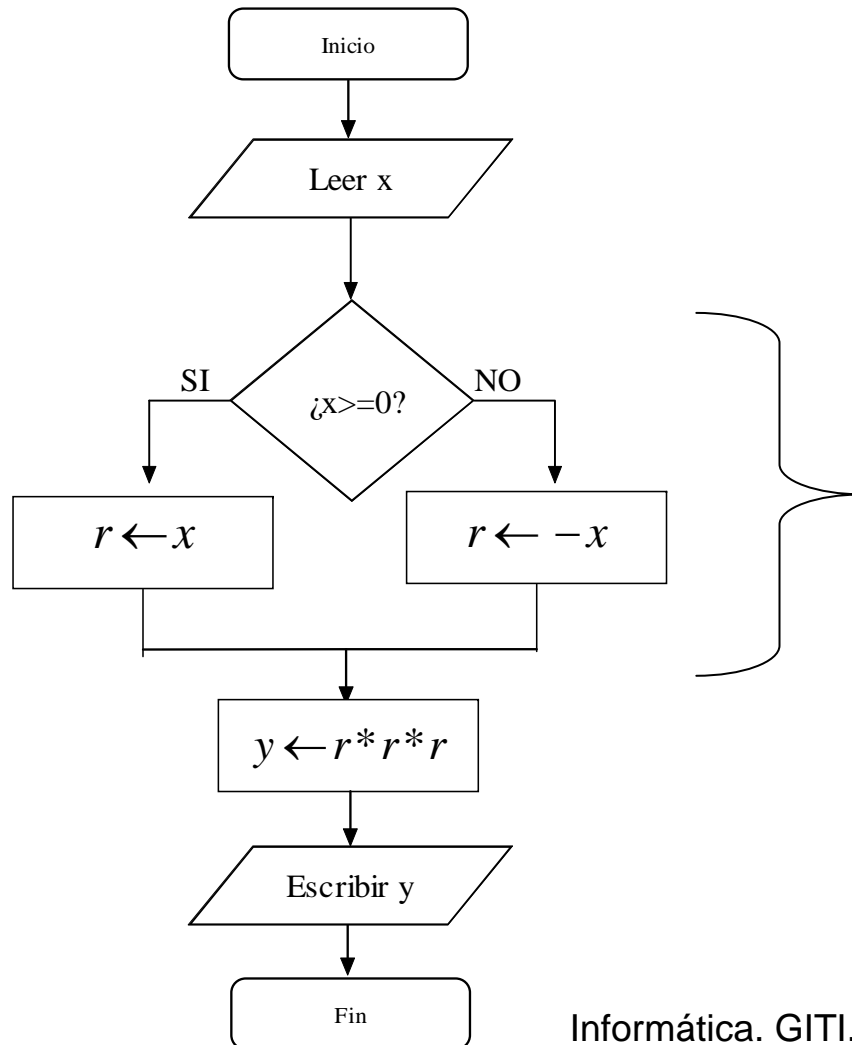
Problema: Leer un número entero x y calcular $y=|x|^3$. Escribir el resultado.

Objeto	Nombre	Valor	Tipo
Dato entrada	x	Variable	Entera
Cero	0	Constante	Entera
$ x $	r	Variable	Entera
Resultado, $ x ^3$	y	Variable	Entera

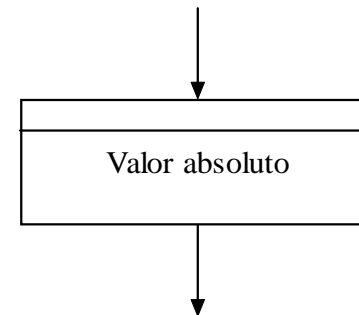


Uso del símbolo módulo:

Problema: Leer un número entero x y calcular $y=|x|^3$. Escribir el resultado.

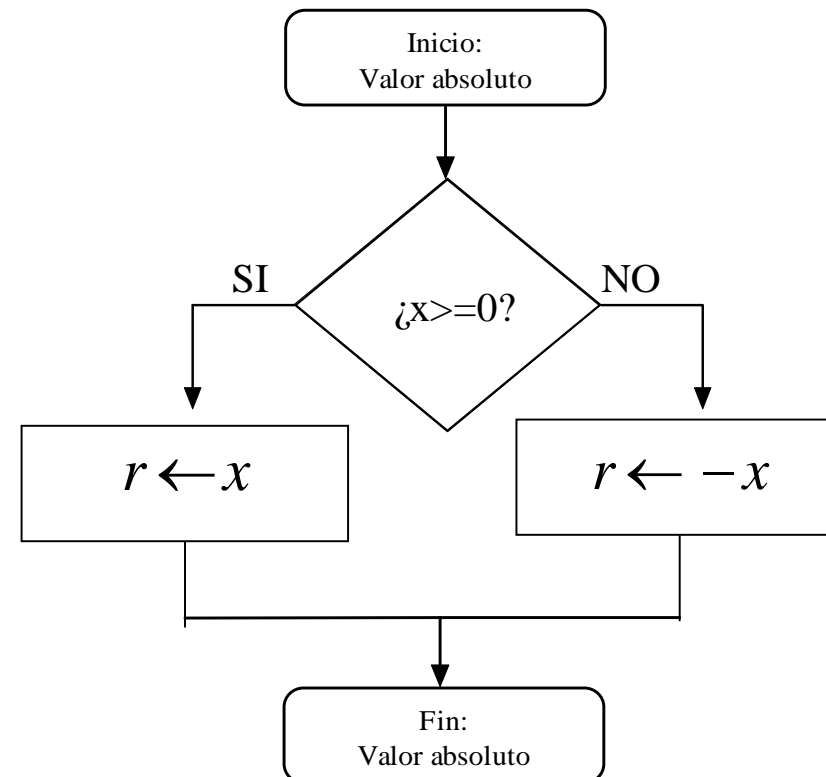
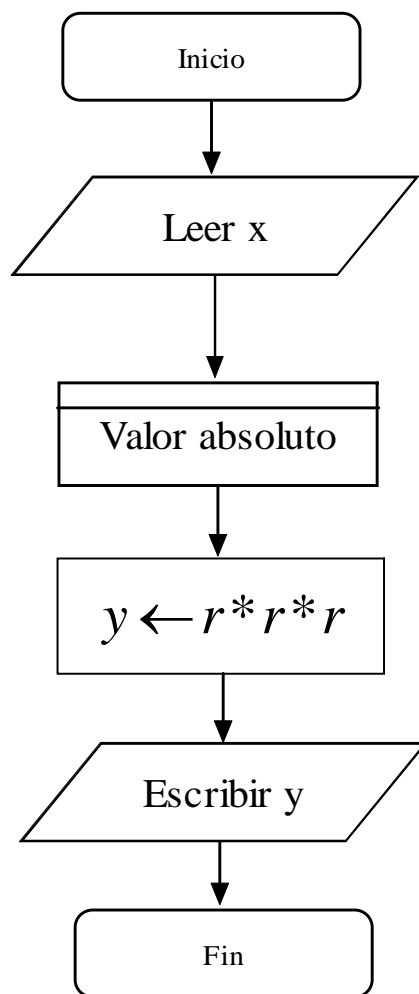


Módulo



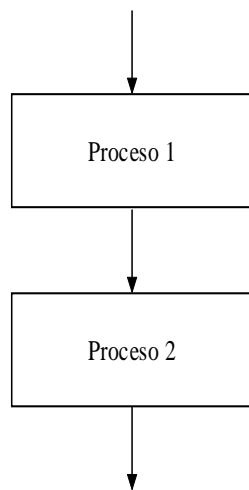
Uso del símbolo módulo:

Problema: Leer un número entero x y calcular $y=|x|^3$. Escribir el resultado.

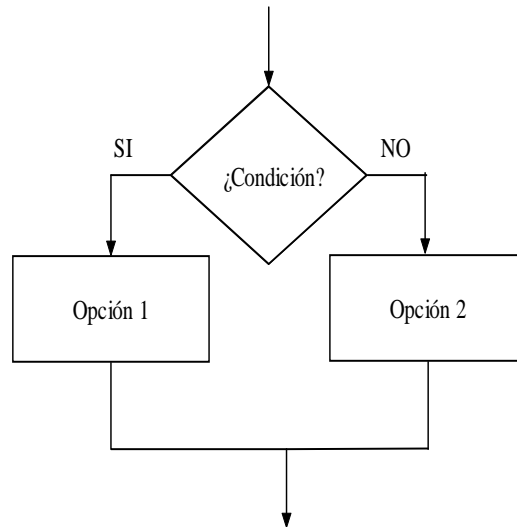


- Comparten TODAS las variables (las variables se deben llamar igual)
- El módulo se puede usar más de una vez en el mismo algoritmo

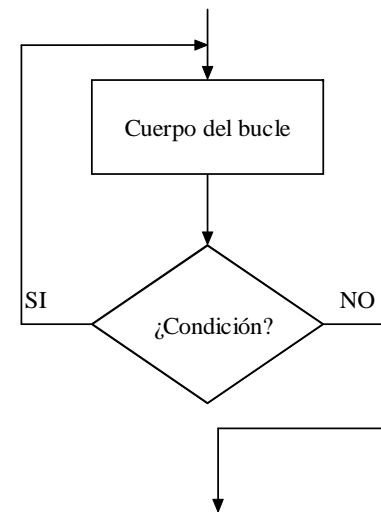
Estructuras de control



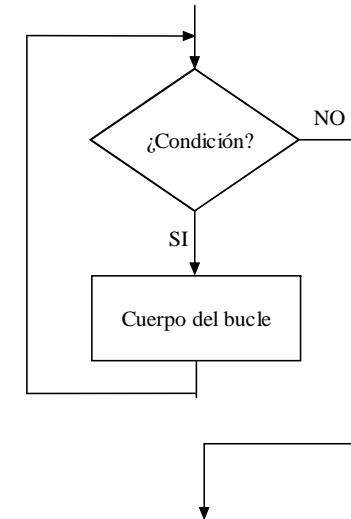
Secuencial



Selectiva



**Iterativa I
(salida en cola)**



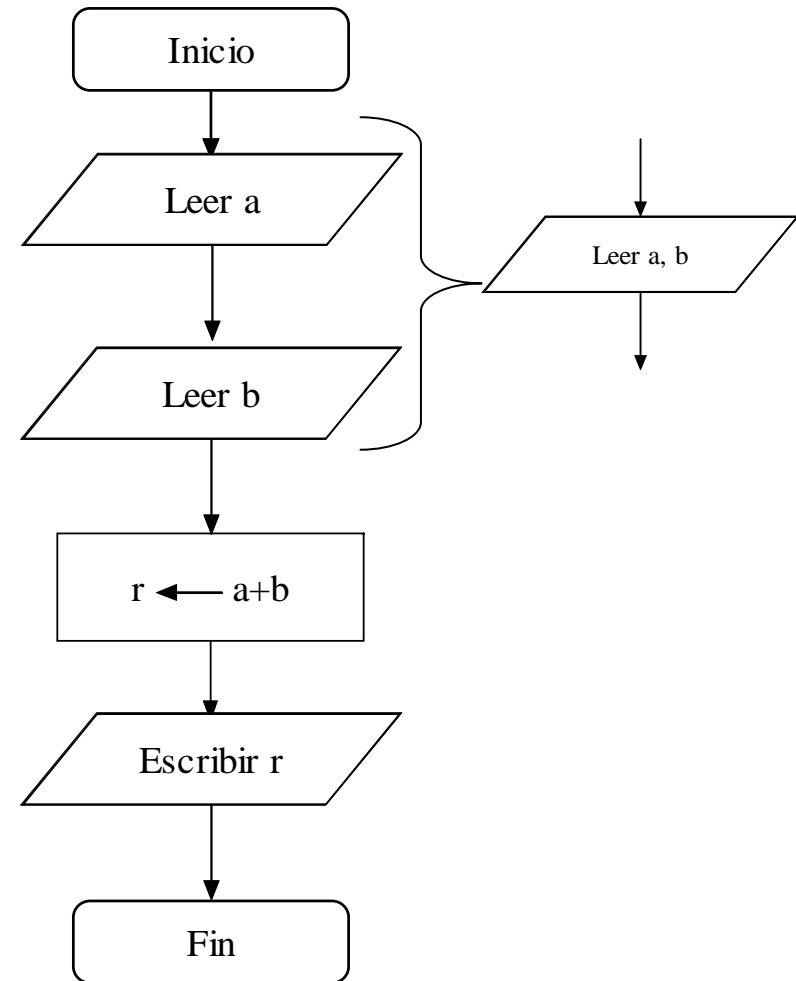
**Iterativa II
(salida en cabeza)**

Estructura secuencial: Ejemplo 1

Problema: leer dos números reales y escribir la suma.

Diagrama de Flujo y Tabla de Objetos:

Objeto	Nombre	Valor	Tipo
Dato entrada	a	Variable	Real
Dato entrada	b	Variable	Real
Resultado, la suma	r	Variable	Real

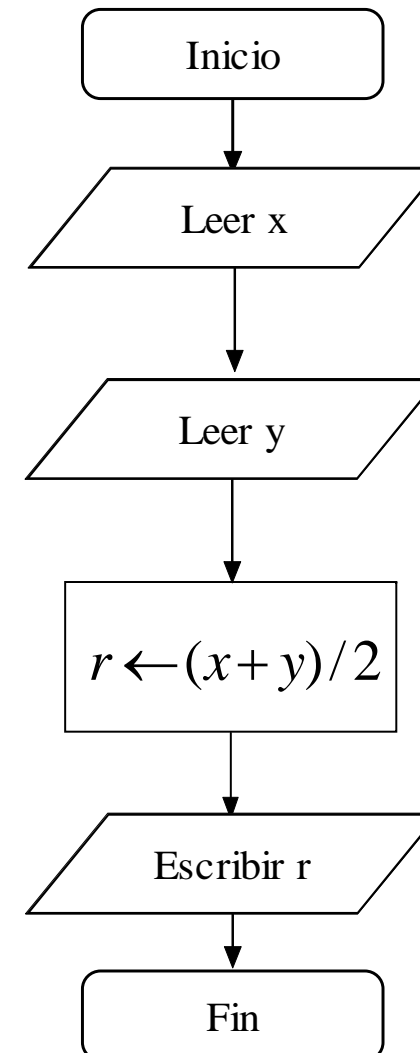


Estructura secuencial: Ejemplo 2

Problema: leer dos números reales y escribir la media aritmética.

Diagrama de Flujo y Tabla de Objetos:

Objeto	Nombre	Valor	Tipo
Dato entrada	x	Variable	Real
Dato entrada	y	Variable	Real
Dos	2	Constante	Entera
Resultado, la media aritmética	r	Variable	Real

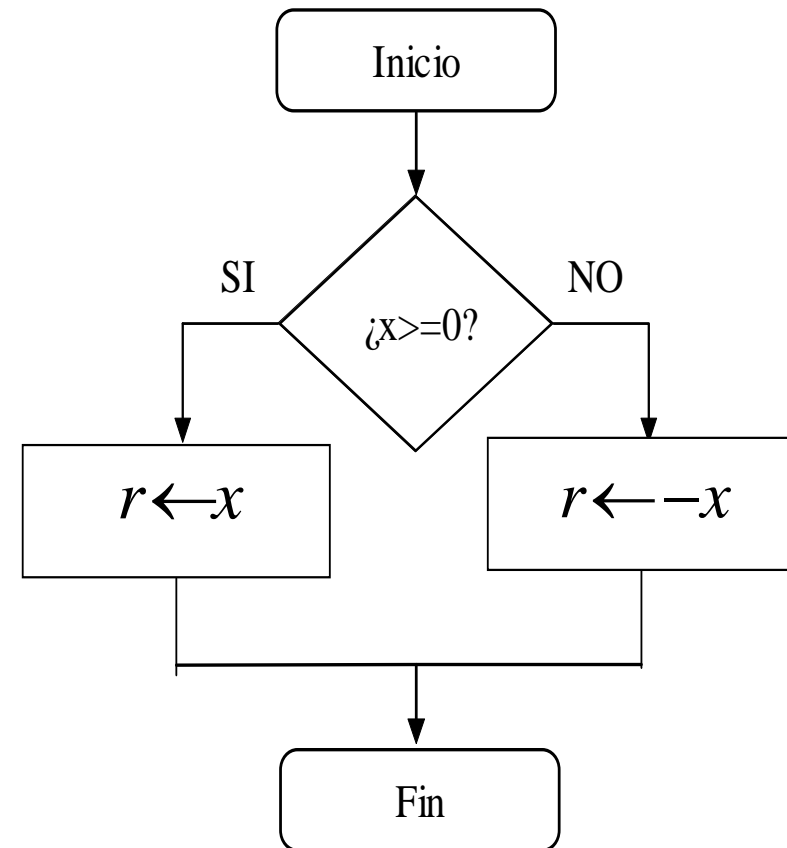


Estructura condicional: Ejemplo 1

Problema: Hallar el valor absoluto de un número x .

Diagrama de Flujo y Tabla de Objetos:

Objeto	Nombre	Valor	Tipo
Dato entrada	x	Variable	Entera
Cero	0	Constante	Entera
Resultado, valor absoluto	r	Variable	Entera



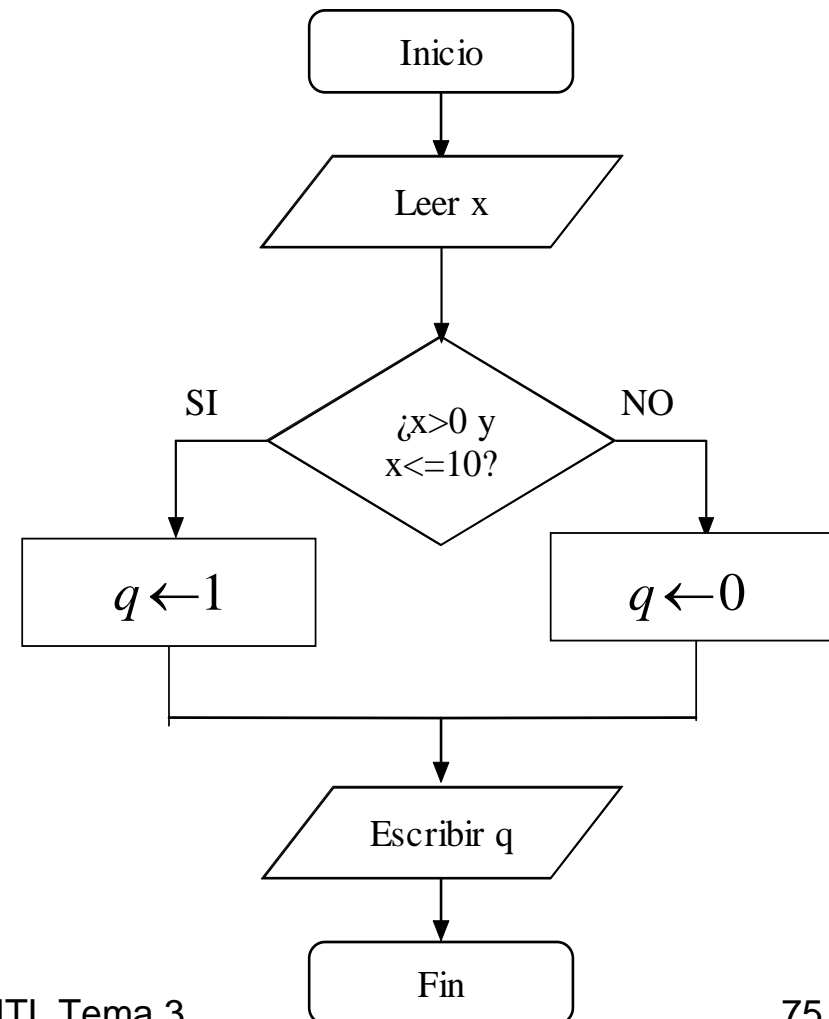
Estructura condicional: Ejemplo 2

Problema: Leer un número real del teclado. Calcular el valor de q , sabiendo que si el valor leído se encuentra en el intervalo $(0,10]$, el resultado q toma el valor de uno, en caso contrario toma el valor de cero. Escribir el resultado.

Estructura condicional: Ejemplo 2

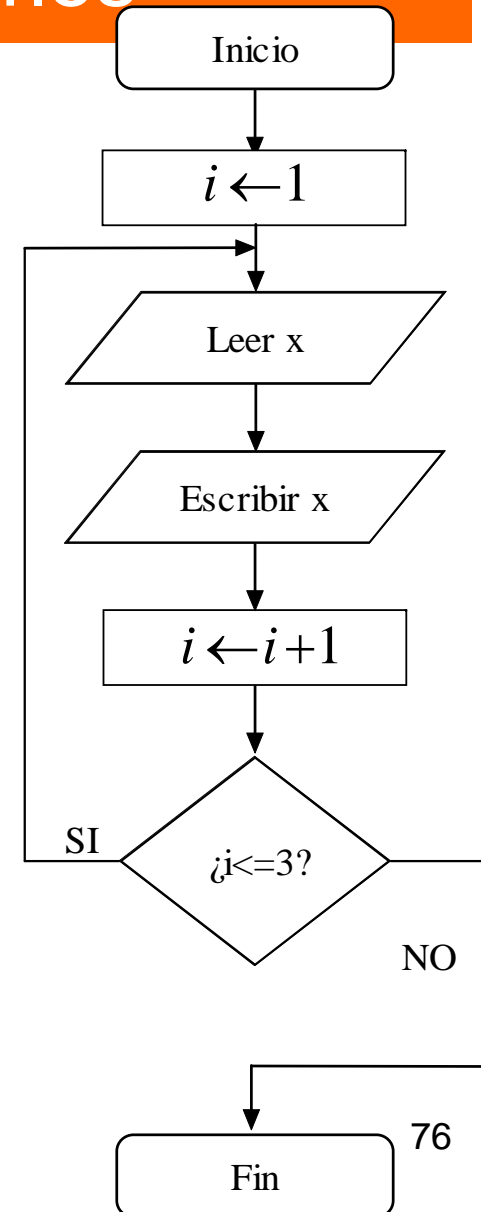
Diagrama de Flujo y Tabla de Objetos:

Objeto	Nombre	Valor	Tipo
Dato entrada	x	Variable	Real
Cero	0	Constante	Entera
Uno	1	Constante	Entera
Diez	10	Constante	Entera
Resultado	q	Variable	Entera

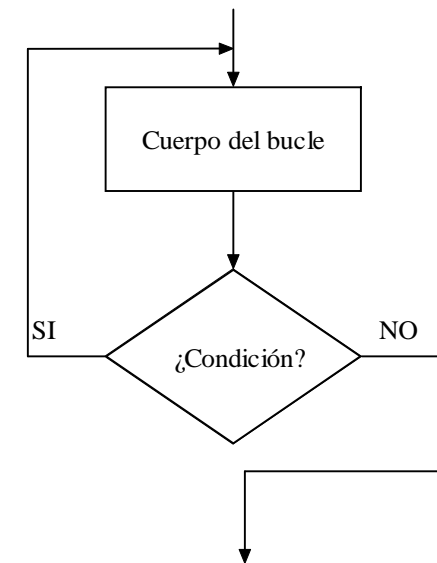
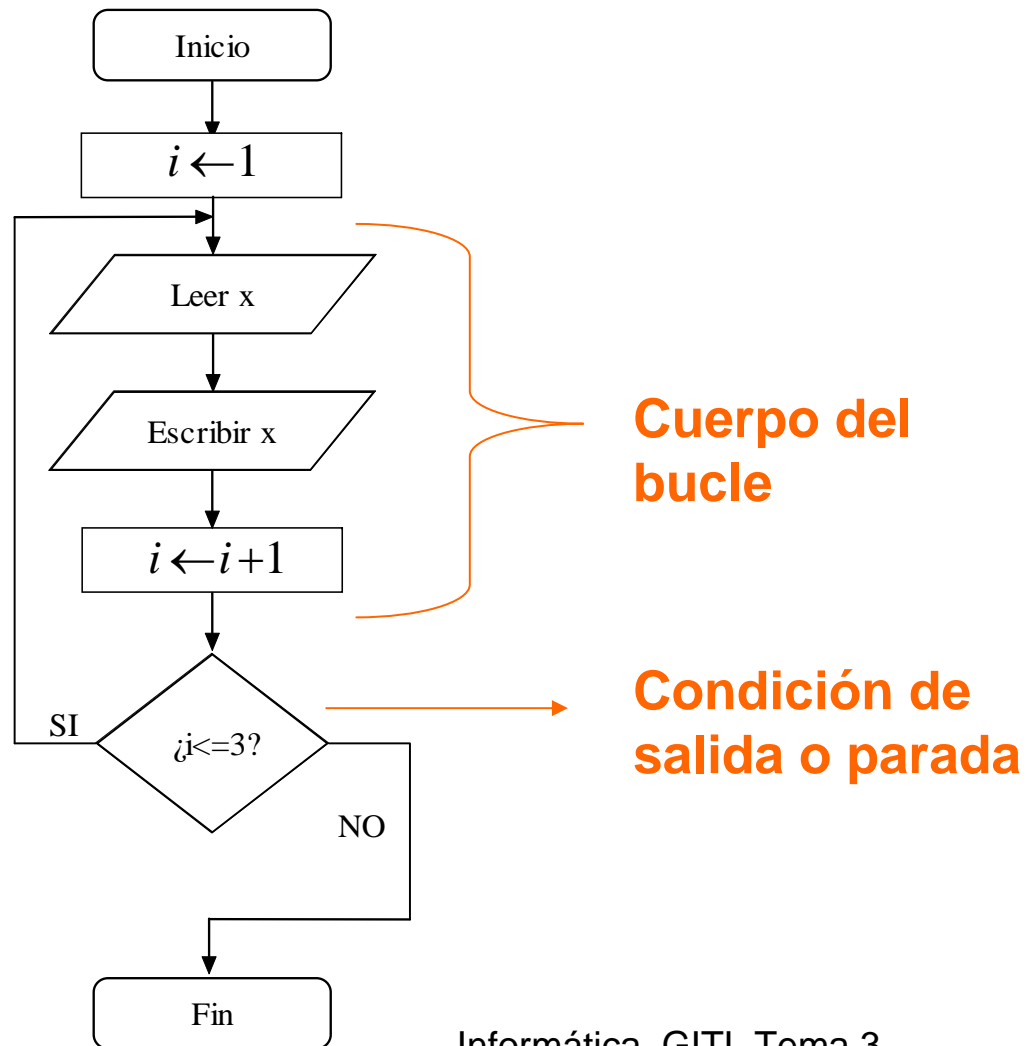


Estructura iterativa salida en cola: Leer tres números reales y escribirlos

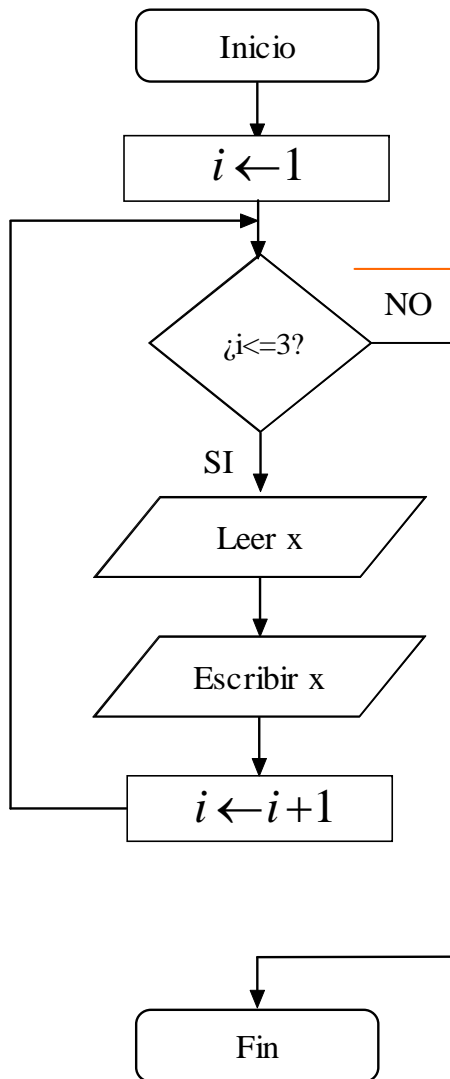
Objeto	Nombre	Valor	Tipo
Dato	x	Variable	Real
Uno	1	Constante	Entera
Tres	3	Constante	Entera
Contador	i	Variable	Entera



Estructura iterativa salida en cola: Leer tres números reales y escribirlos

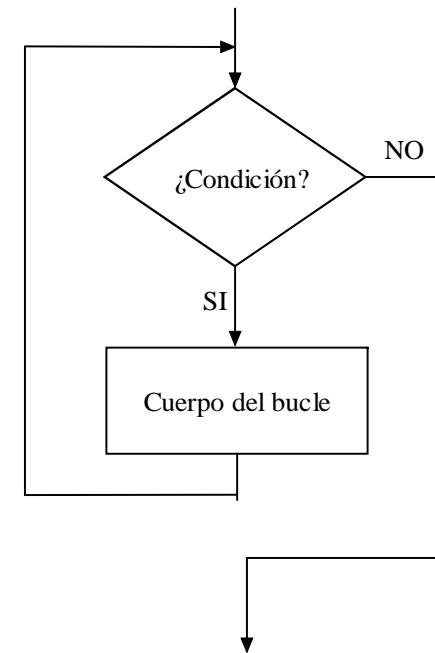


Estructura iterativa salida en cabeza: Leer tres números reales y escribirlos



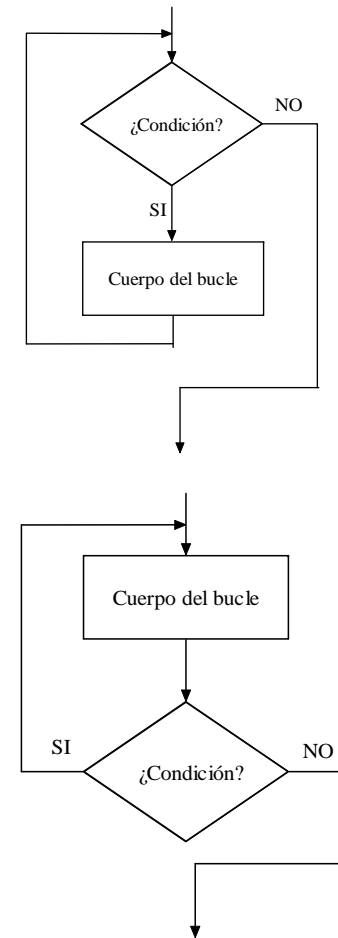
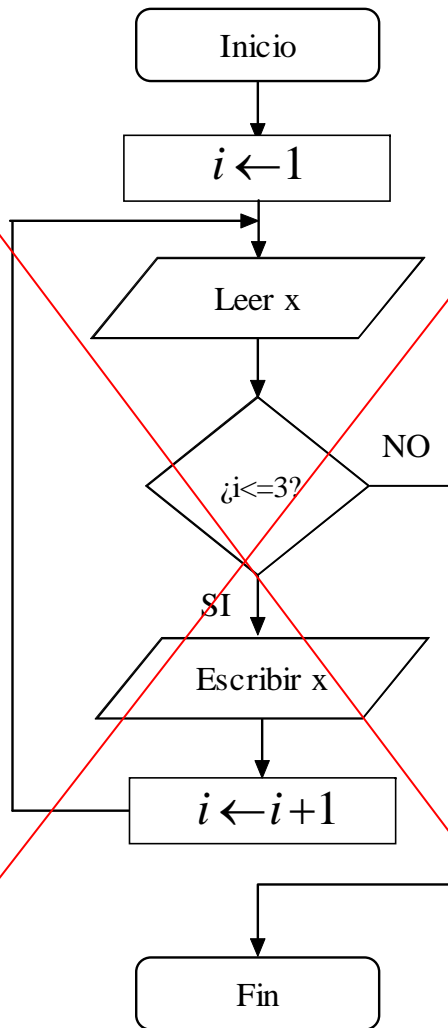
Condición de salida o parada

Cuerpo del bucle



PROGRAMACIÓN NO ESTRUCTURADA:

Leer tres números reales y escribirlos



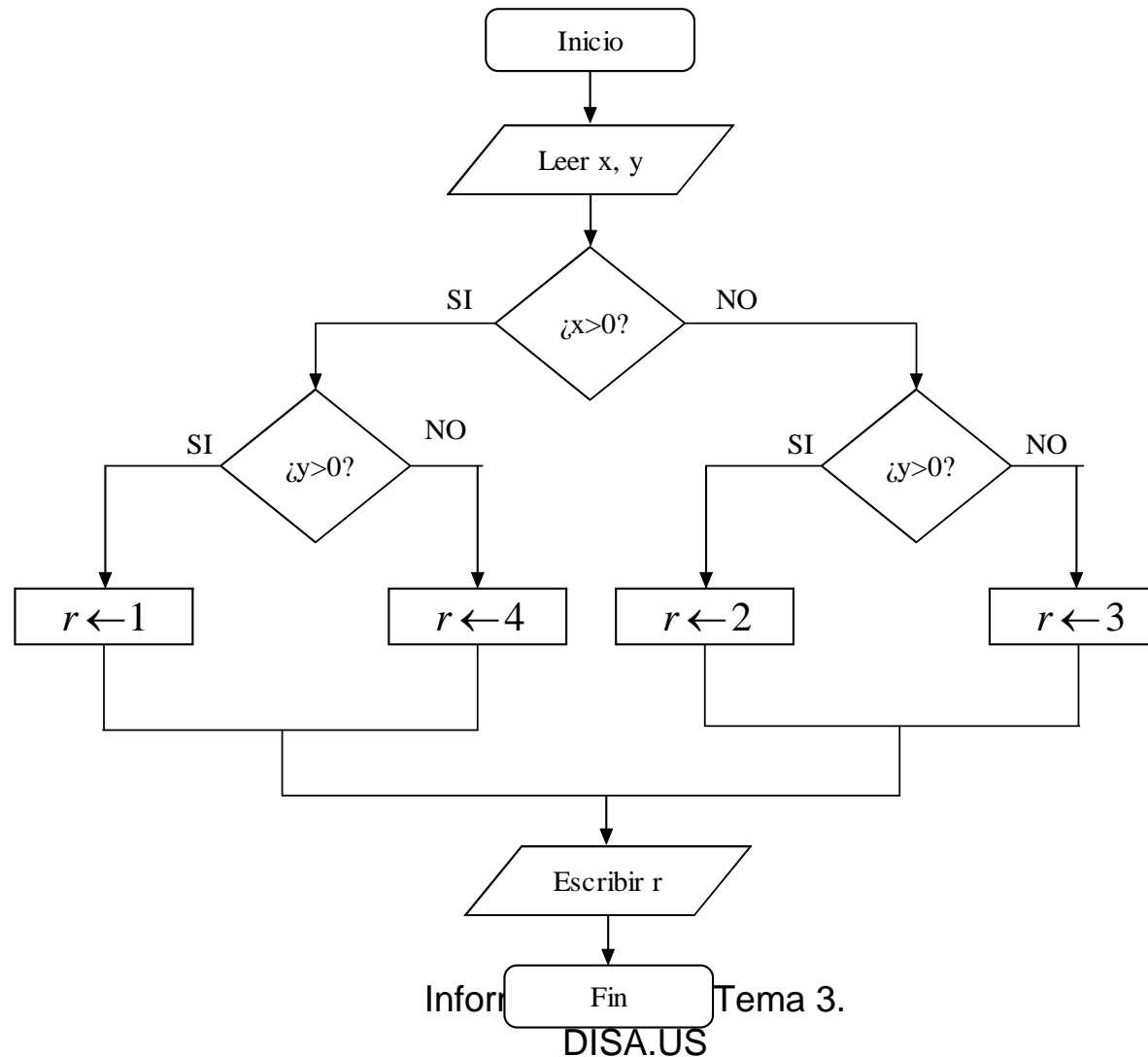
Estructuras de control anidadas

- ❑ Las estructuras de control se pueden anidar
- ❑ Dentro de una estructura selectiva puede haber otra estructura del mismo tipo o una estructura iterativa.
- ❑ En el cuerpo del bucle de una estructura iterativa puede haber otra estructura condicional o iterativa (bucles anidados).

Estructuras de control anidadas

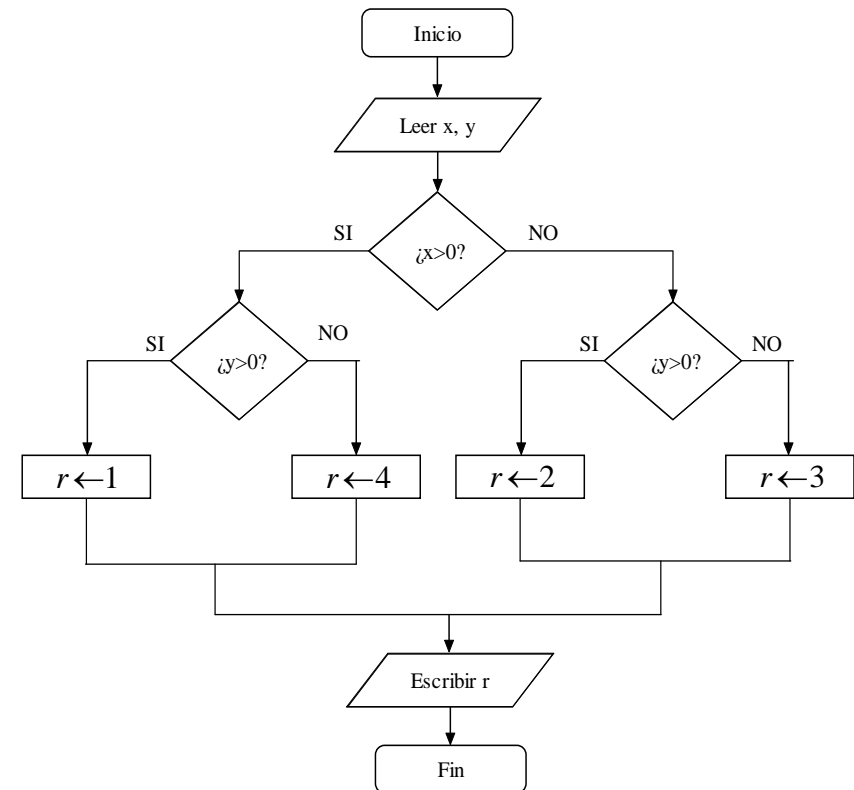
- ❑ *Diseñar un algoritmo para calcular y escribir el cuadrante al que pertenece un punto del plano cuyas coordenadas (x, y) se proporcionan. El resultado ha de ser un número entero r de 1 a 4*

Estructuras de control anidadas



Estructuras de control anidadas

Objeto	Nombre	Valor	Tipo
Dato	x	Variable	Real
Dato	y	Variable	Real
Cero	0	Constante	Entera
Uno	1	Constante	Entera
Dos	2	Constante	Entera
Tres	3	Constante	Entera
Cuatro	4	Constante	Entera
Resultado	r	Variable	Real



7. Traza o tabla de ejecución

- ❑ La traza de un algoritmo permite seguir el valor de las diferentes variables a medida que el algoritmo se va implementando
- ❑ La traza de un algoritmo normalizado la realizaremos en una tabla:
 - ❑ 1º Columna sentencia/condición evaluada
 - ❑ Una columna por cada variable del algoritmo
 - ❑ Cada fila muestra el valor de cada variable al terminar de evaluarse la correspondiente sentencia/condición
 - ❑ Se utiliza una fila por cada paso del algoritmo
 - ❑ Al ejecutarse la sentencia de Inicio el valor de todas las variables es desconocido
 - ❑ Al ejecutarse la sentencia Fin el algoritmo termina

7. Traza o tabla de ejecución

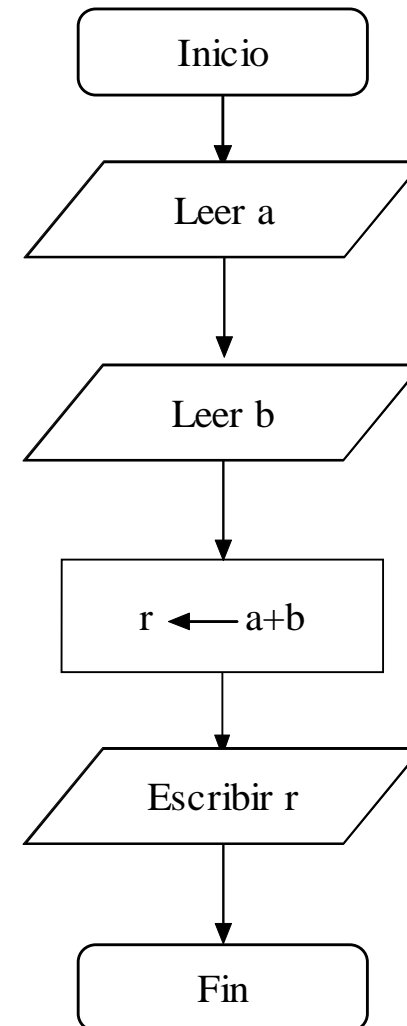
- ❑ La traza permite comprobar si un algoritmo funciona correctamente
 - ❑ Sólo es útil para analizar algoritmos sencillos
 - ❑ Con algoritmos más complejos, utilizaremos métodos basados en la traza pero más eficientes (ej.herramienta Debug)
- ❑ La traza de un algoritmo depende de los datos de entrada
 - ❑ Para diferentes datos leídos de teclado, se obtienen diferentes trazas
 - ❑ Para analizar si un algoritmo funciona correctamente hay que probar todos los posibles casos de entradas

7. Traza o tabla de ejecución

- ❑ El número de sentencias/condiciones evaluadas por un algoritmo es variables y desconocido a priori
 - ❑ La dimensión de la traza depende de las entradas
- ❑ Al evaluar una condición, es recomendable marcar si la condición era verdadera o falsa

Problema I: Leer dos números reales y escribir la suma.

Instrucción	a	b	r
Inicio	-	-	-
Leer a	5	-	-
Leer b	5	3	-
$r \leftarrow a + b$	5	3	8
Escribir r	5	3	8
Fin	5	3	8



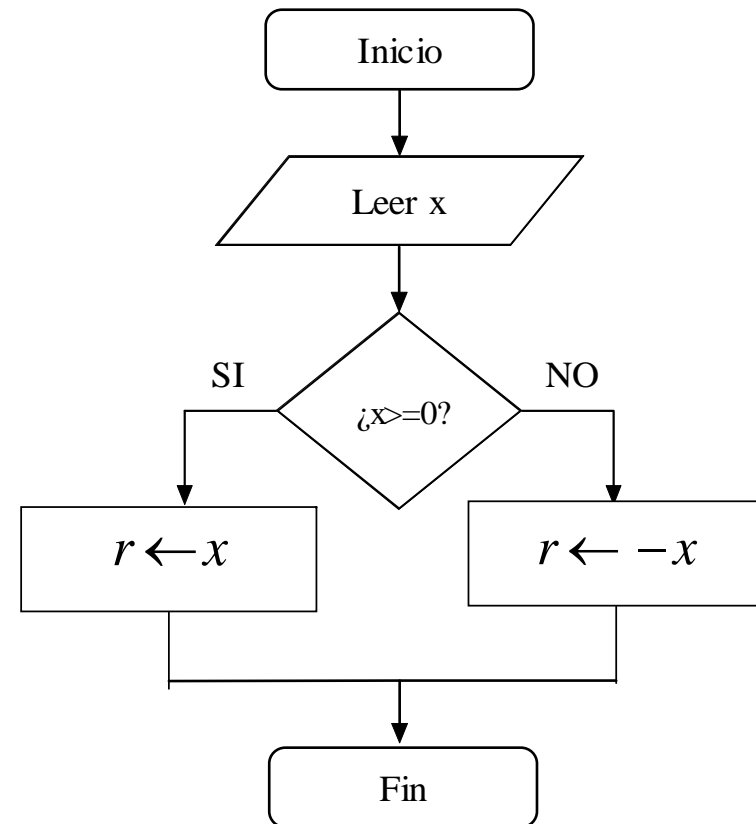
Problema II: Hallar el valor absoluto de un número x

Instrucción	x	r
Inicio	-	-
Leer x	10	-
¿ $x \geq 0$?	10	10
Fin	10	10

SI

Instrucción	x	r
Inicio	-	-
Leer x	-20	-
¿ $x \geq 0$?	-20	20
Fin	-20	20

NO

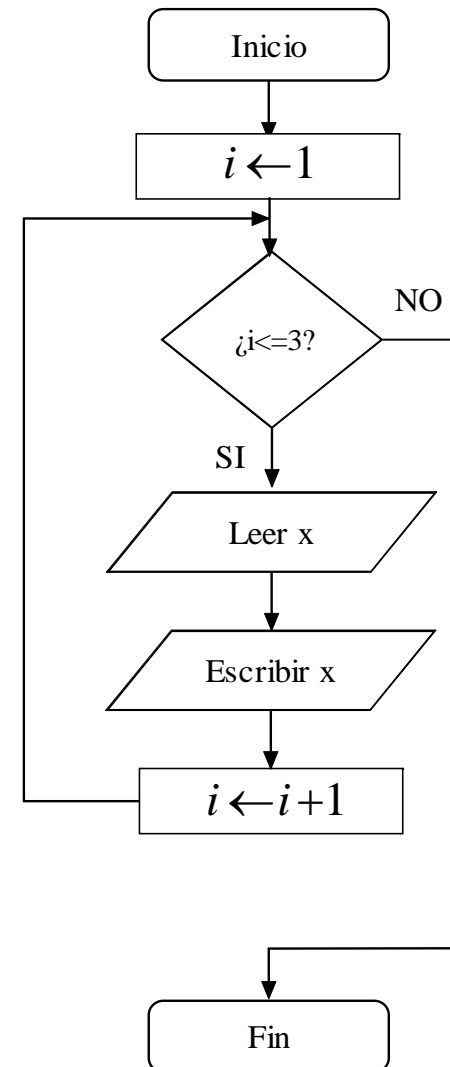


Problema III: Leer tres números reales y escribirlos

Instrucción	i	x
Inicio	-	-
$i < -0$	1	-
1. ¿ $i \leq 3$?	1	-
Leer x	1	3.5
Escribir x	1	3.5
$i \leftarrow i + 1$	2	3.5
2. ¿ $i \leq 3$?	2	7.9
Leer x	2	7.9
Escribir x	2	7.9

SI

SI

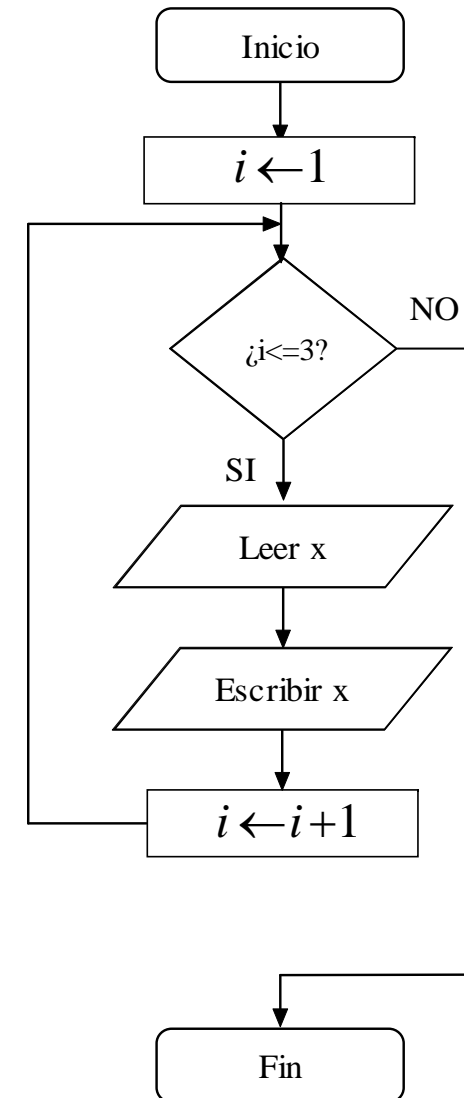


Problema III: Leer tres números reales y escribirlos

Instrucción	i	x
$i \leftarrow i + 1$	3	7.9
3. ¿ $i \leq 3$?	3	7.9
Leer x	3	2.1
Escribir x	3	2.1
$i \leftarrow i + 1$	4	2.1
4. ¿ $i \leq 3$?	4	2.1
Fin	4	2.1

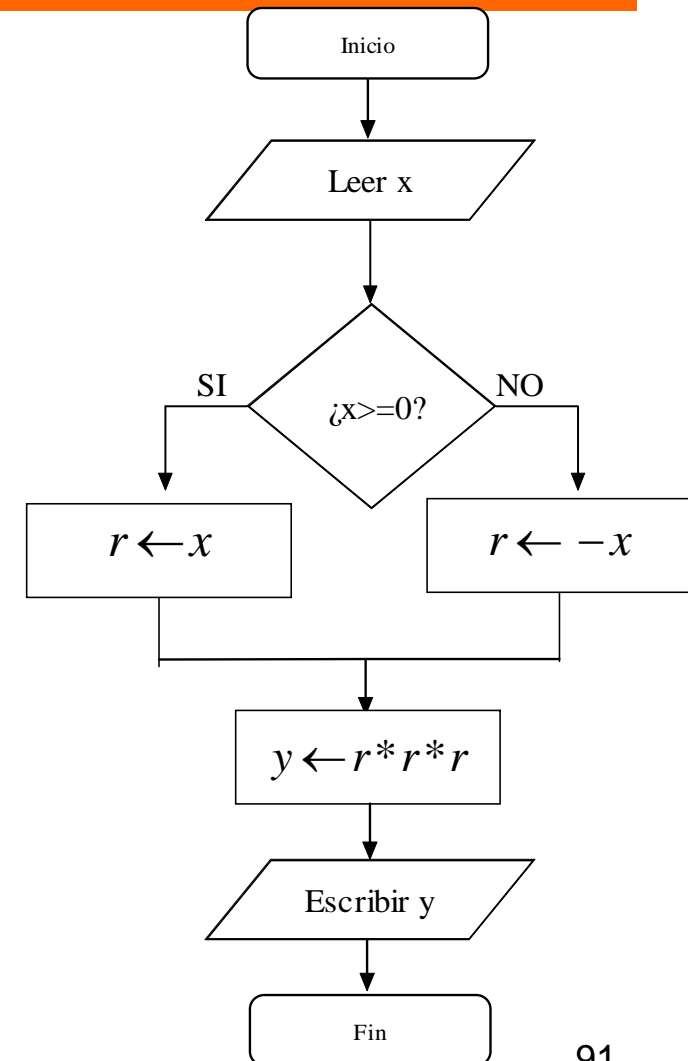
SI

NO



Problema IV: Leer un número entero x y calcular $y=|x|^3$. Escribir el resultado

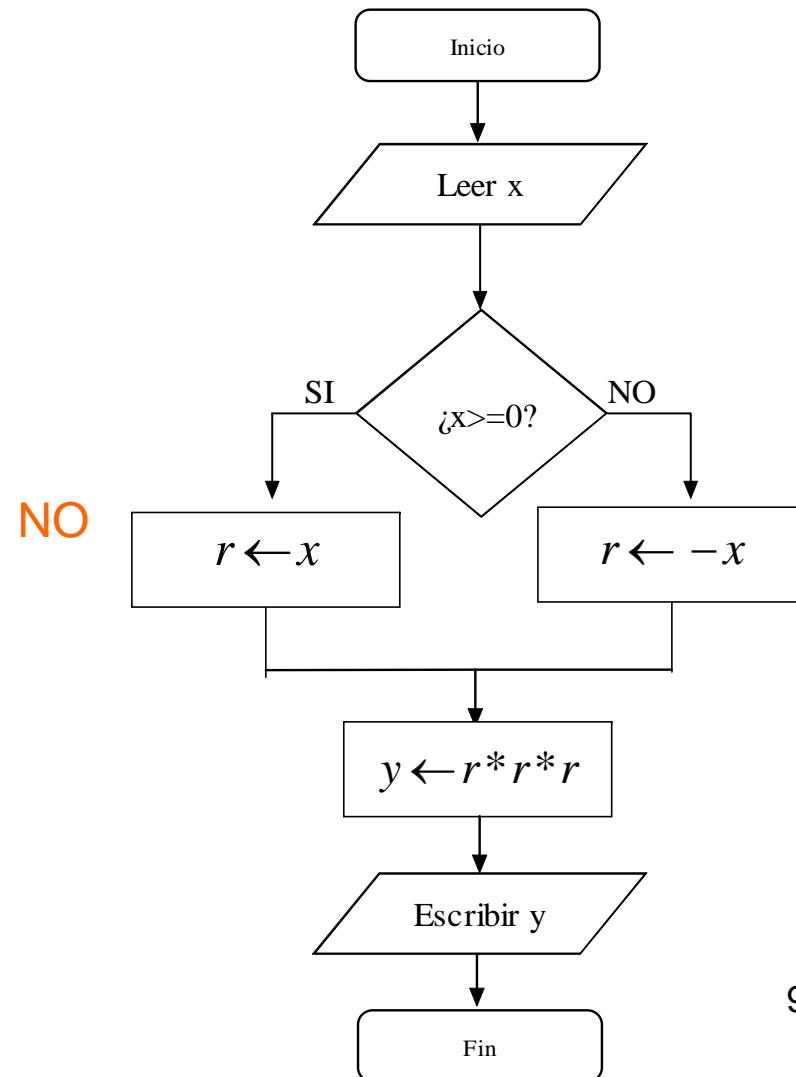
Objeto	Nombre	Valor	Tipo
Dato	x	Variable	Entera
Cero	0	Constante	Entera
x	r	Variable	Entera
Resultado, $ x ^3$	y	Variable	Entera



Problema IV

Número leído por teclado -3

Instrucción	x	r	y
Inicio	-	-	-
Leer x	-3	-	-
¿x>=0?	-3	-	-
r <- -x	-3	3	-
y <- r*r*r	-3	3	27
Escribir y	-3	3	27
Fin	-3	3	27



Problema IV

Número leído por teclado 2

Instrucción	x	r	y
Inicio	-	-	-
Leer x	2	-	-
¿x>=0?	2	-	-
r <- x	2	2	-
y <- r*r*r	2	2	8
Escribir y	2	2	8
Fin	2	2	8

SI

