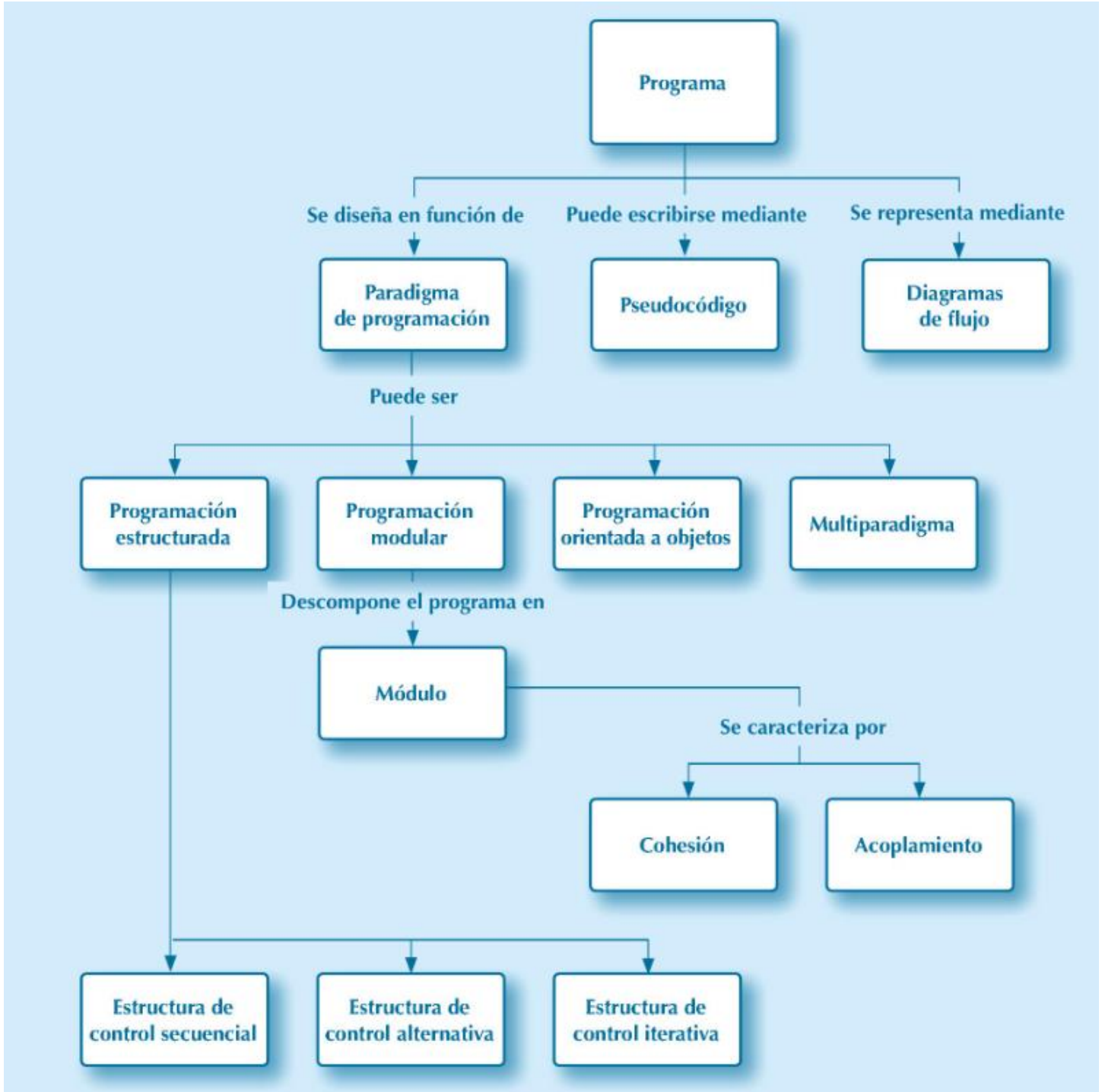
INTRODUCCIÓN A LA PROGRAMACIÓN

1º DAW

IES. MARÍA MOLINER 2023/2024


Programa
Se diseña en función de
Puede escribirse mediante
Se representa mediante
Paradigma de programación
Pseudocódigo
Diagramas de flujo
Puede ser
Programación estructurada
Programación modular
Programación orientada a objetos
Multiparadigma
Descompone el programa en
Módulo
Se caracteriza por
Cohesión
Acoplamiento
Estructura de control secuencial
Estructura de control alternativa
Estructura de control iterativa

El concepto *programación* ha ido evolucionando de forma pareja a como lo han ido haciendo los dispositivos y tecnologías.

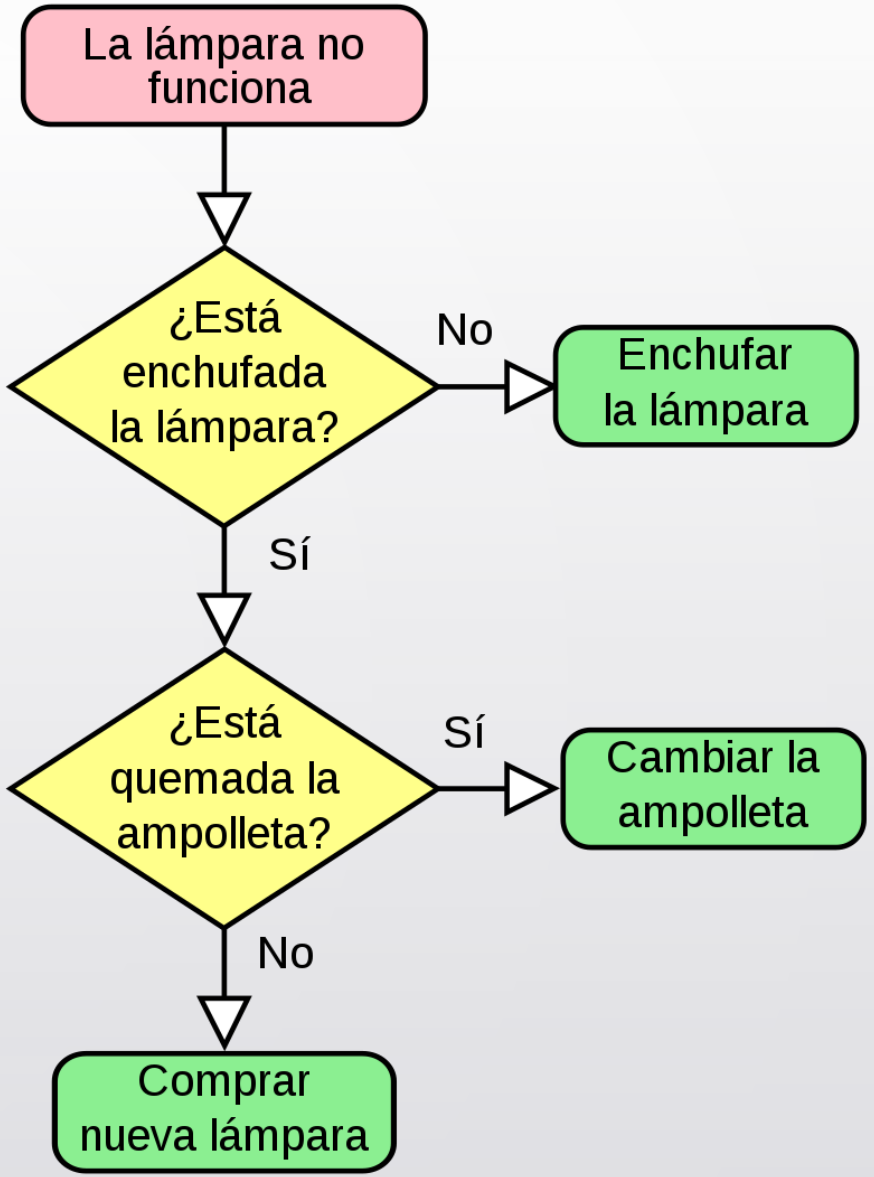
En las primeras computadoras las instrucciones de los programas eran escritas en código binario (ceros y unos).

Dada la complicación asociada a esta técnica de programación, los científicos que estudiaban esta nueva ciencia decidieron sustituir determinadas secuencias binarias por palabras que permitieran representarlas más fácilmente, lo que dio lugar al conocido *lenguaje ensamblador*.

Posteriormente aparecieron los lenguajes de alto nivel, que añadiendo una nueva capa de abstracción al proceso de programación y permitieron el desarrollo de programas cada vez más sofisticados a la vez que más fáciles de desarrollar.

DIAGRAMAS DE FLUJO

- Def: La representación gráfica de un algoritmo o proceso.
- Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva.
- Facilita la comprensión del algoritmo gracias a la descripción visual que aporta sobre el flujo de este.



DIAGRAMAS DE FLUJO

- Mediante DF se podrán representar las estructuras de control

Figura 1.15
Secuencial.

Figura 1.16
Alternativa simple.

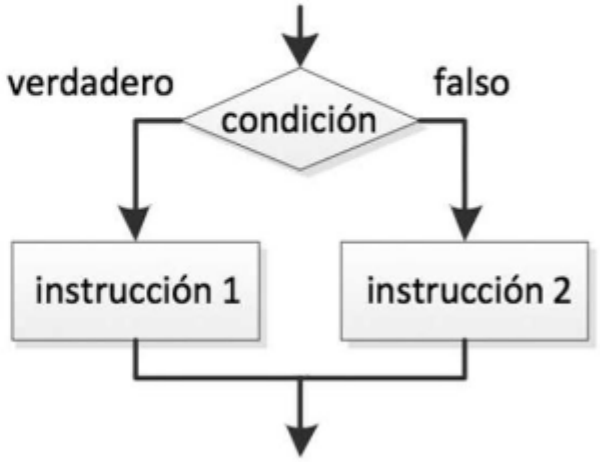


Figura 1.17
Alternativa doble.

DIAGRAMAS DE FLUJO

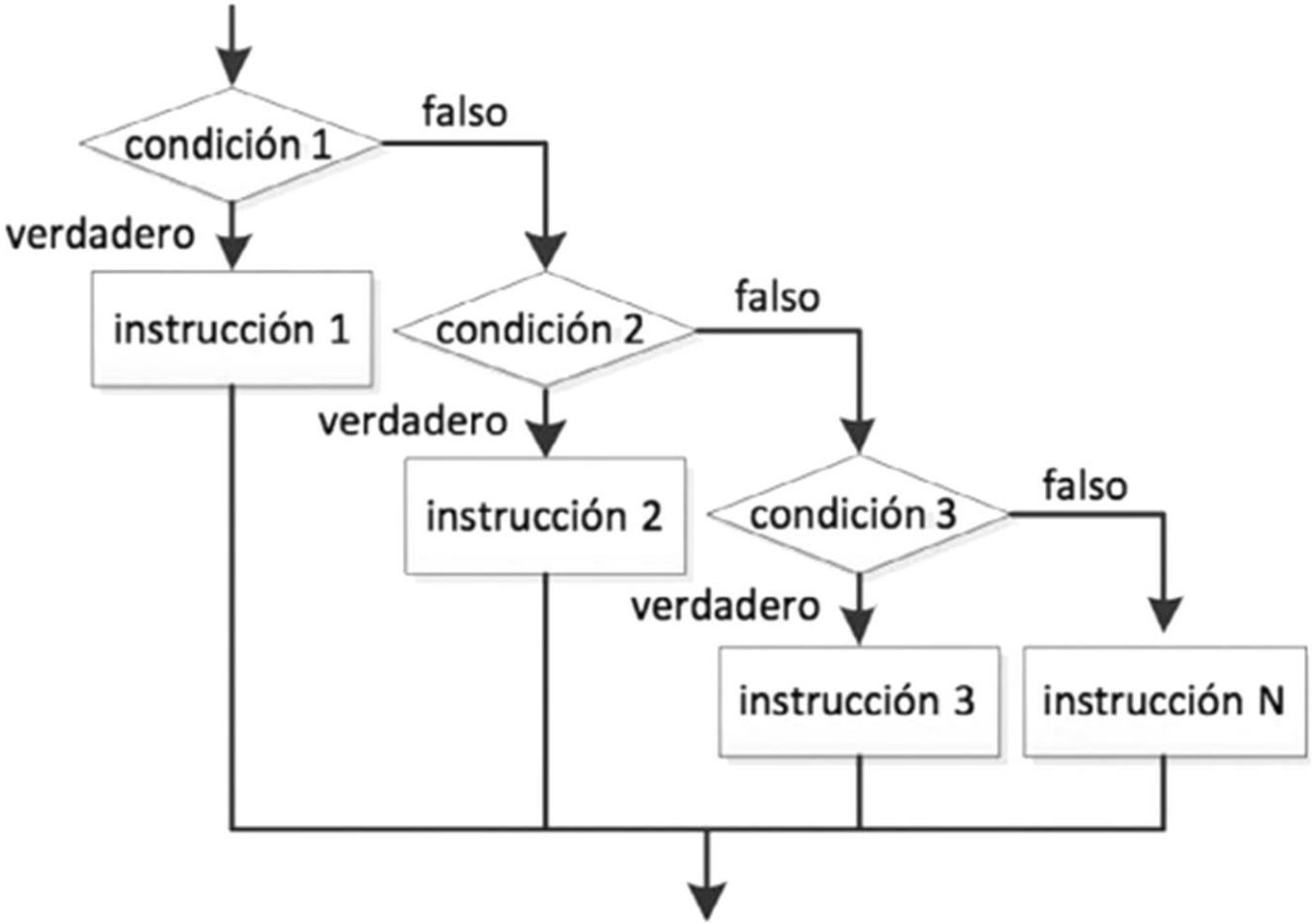


Figura 1.18
Alternativa múltiple con simples anidadas.

DIAGRAMAS DE FLUJO

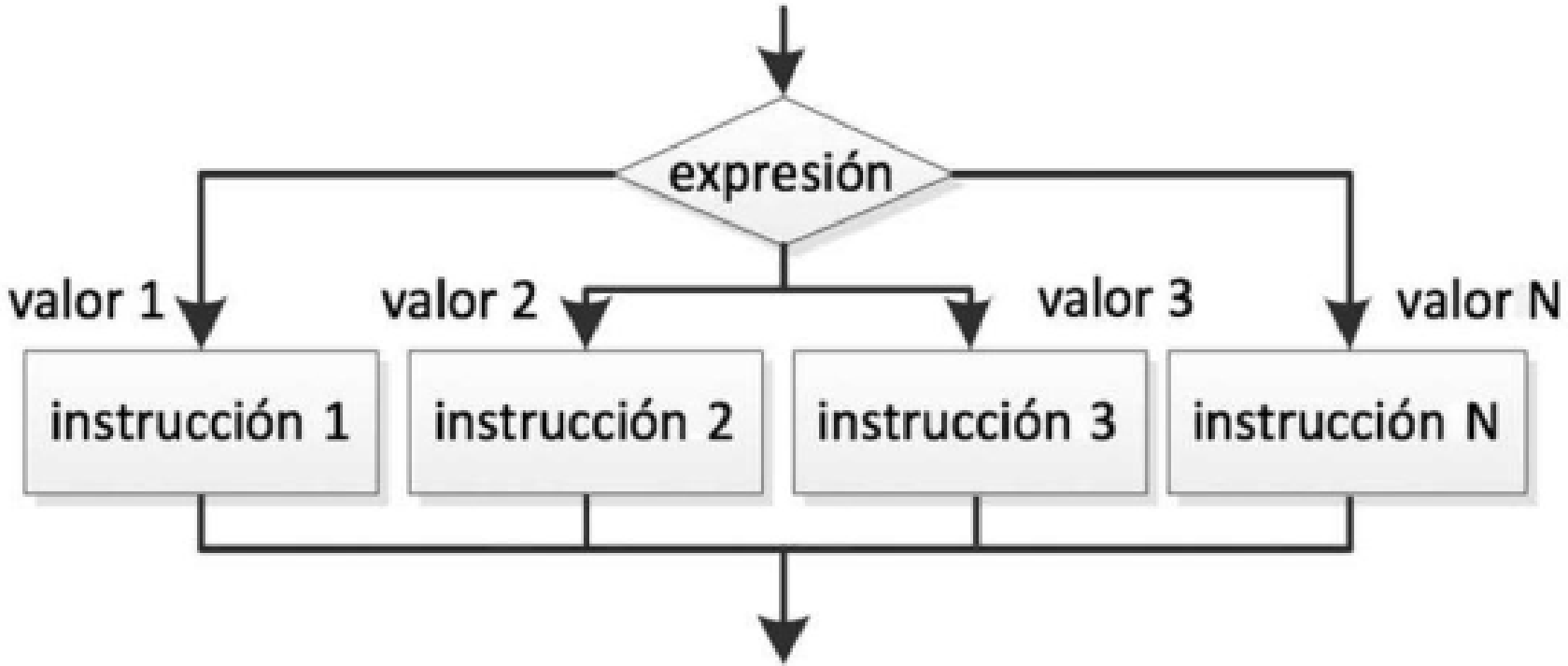


Figura 1.19
Alternativa múltiple II.

DIAGRAMAS DE FLUJO

Figura 1.20
Iterativa *while*.

Figura 1.21
Iterativa *repeat*.

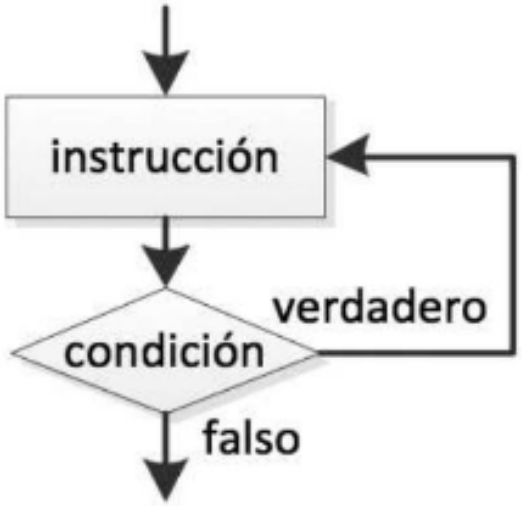
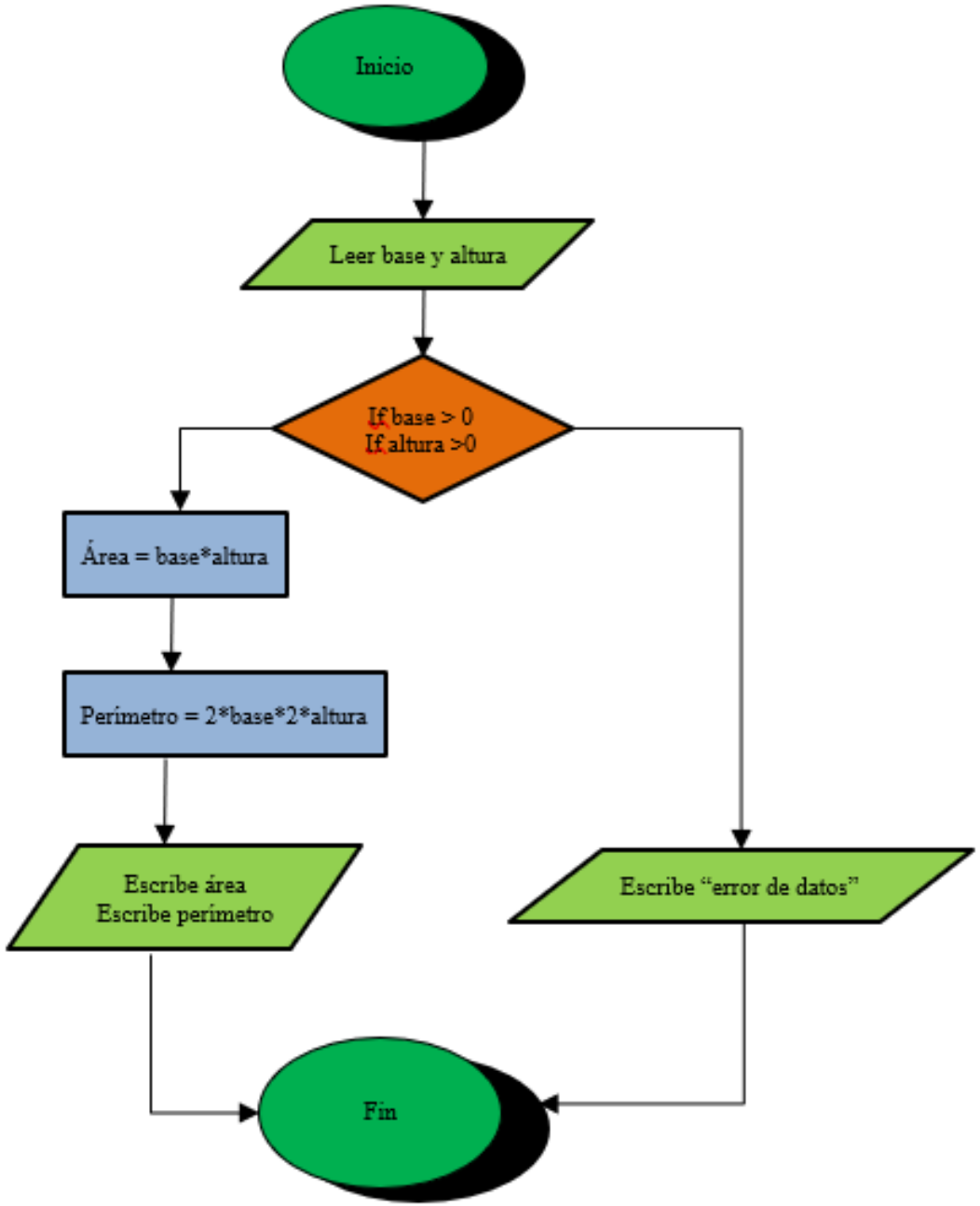
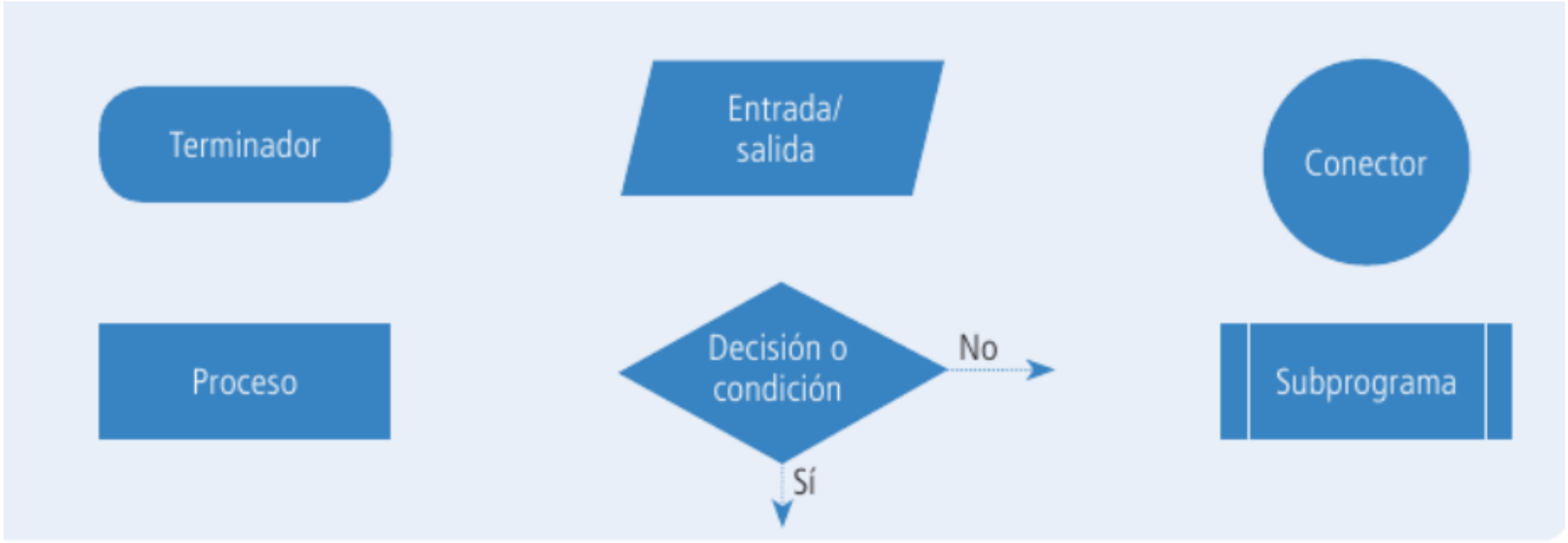


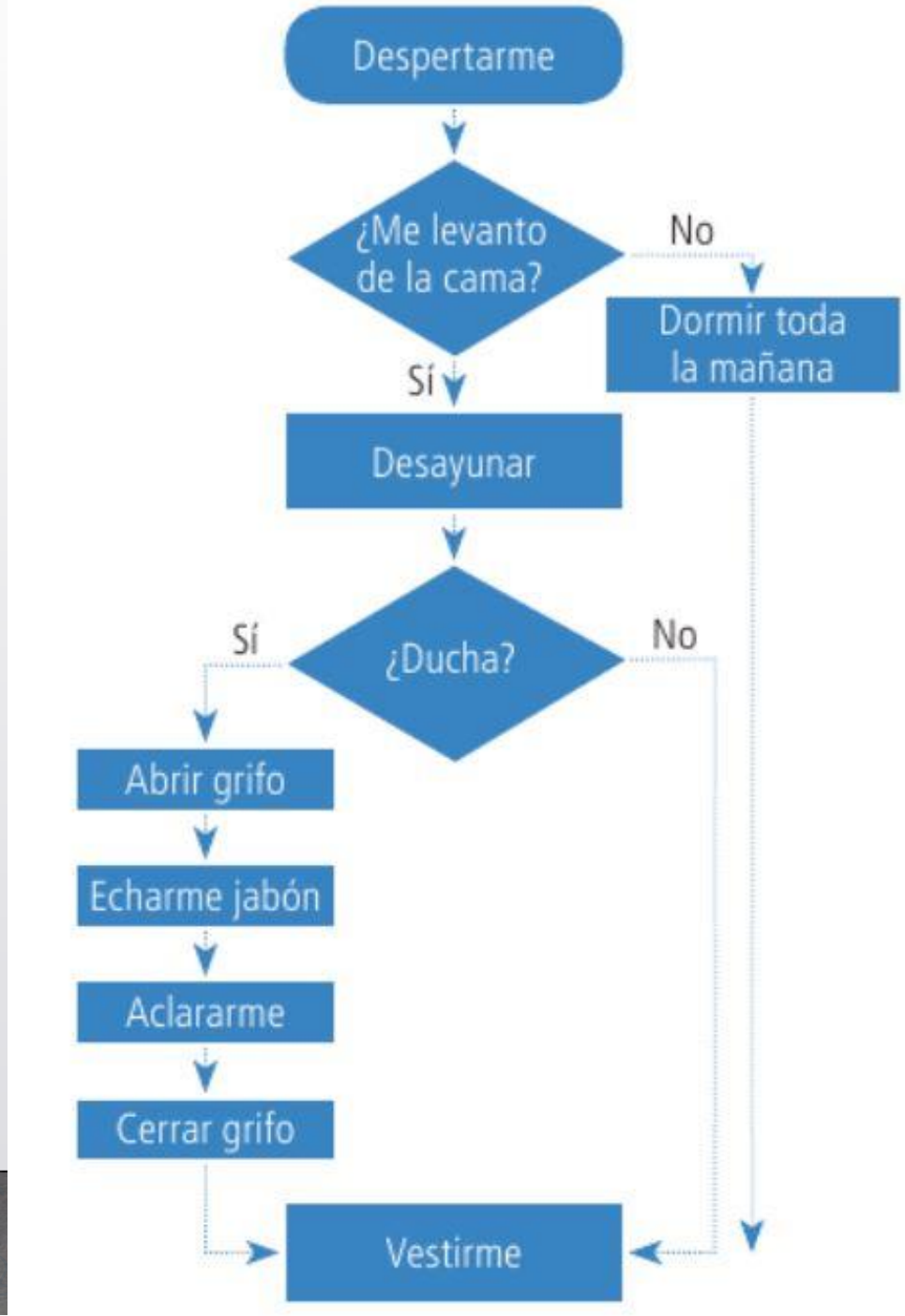
Figura 1.22
Iterativa *do while*.


Inicio
Leer base y altura
If base > 0
If altura >0
Área = base*altura
Perímetro = 2*base*2*altura
Escribe área
Escribe perímetro
Escribe "error de datos"
Fin


Terminador
Entrada/
salida
Conector
Proceso
Decisión o
condición
No
Subprograma
Sí

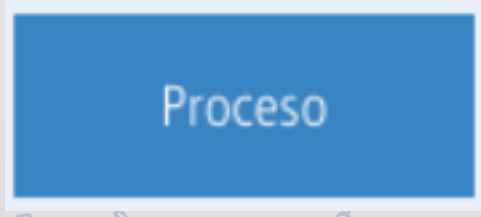
- **Terminador.** Indica el inicio o el fin del diagrama.
- **Proceso.** Representa cada una de las operaciones que se van a realizar.
- **Entrada/salida.** Indica recogida o salida de datos. En un programa normalmente la entrada se realizará a través del teclado o el ratón y la salida será en la pantalla.
- **Decisión o condición.** Se evalúa una condición y, según el resultado sea verdadero o falso, se sigue un camino u otro.
- **Conector.** Sirve para enlazar partes de un diagrama que pueden dividirse y mostrarse en varias columnas, por ejemplo.
- **Subprograma.** Representa un conjunto de tareas independientes que se ejecutarán para volver después de nuevo al flujo del algoritmo.


Terminador
Entrada/
salida
Conector
Proceso
Decisión o
condición
No
Sí
Subprograma



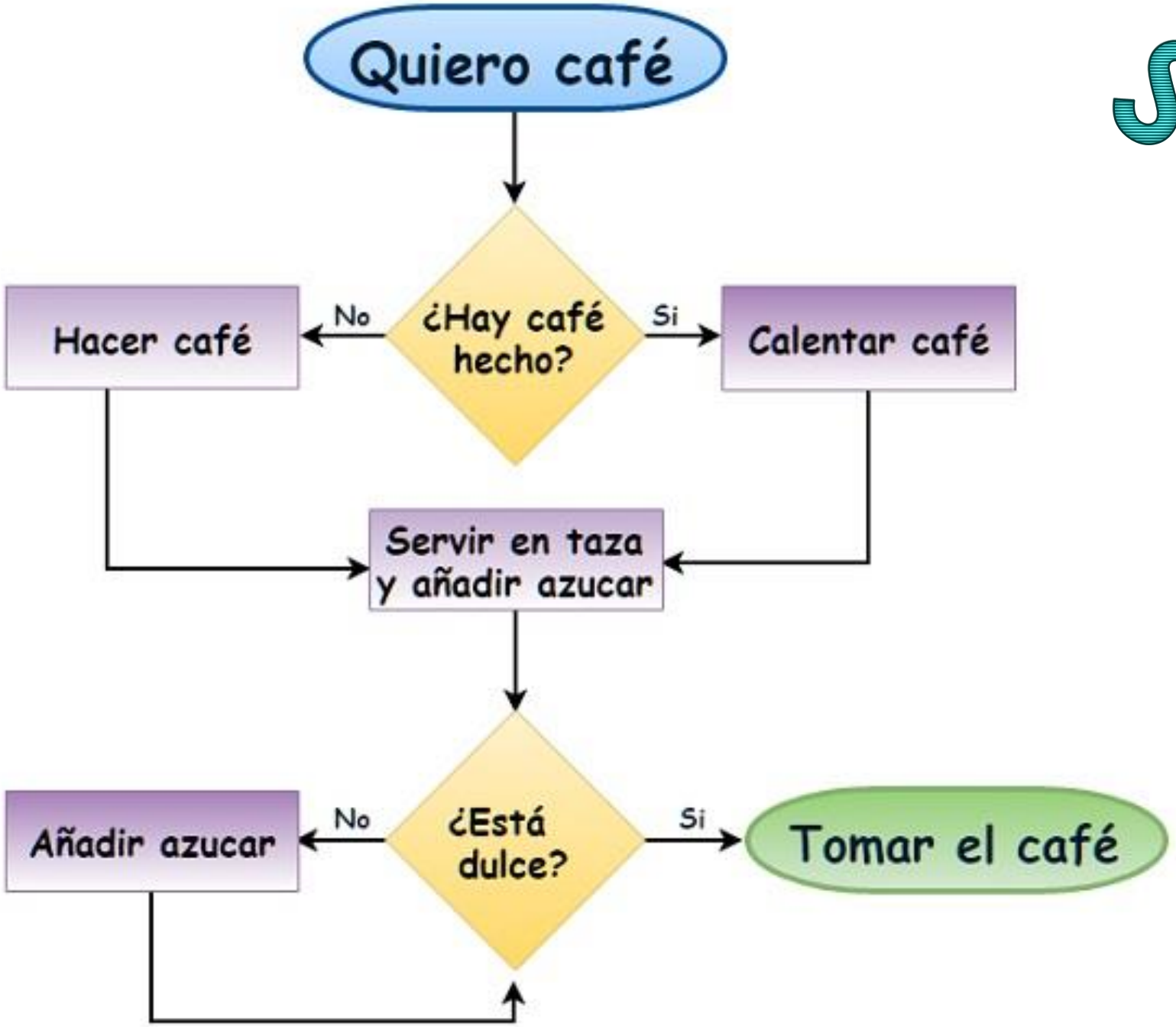
REALIZA UN DIAGRAMA DE FLUJO

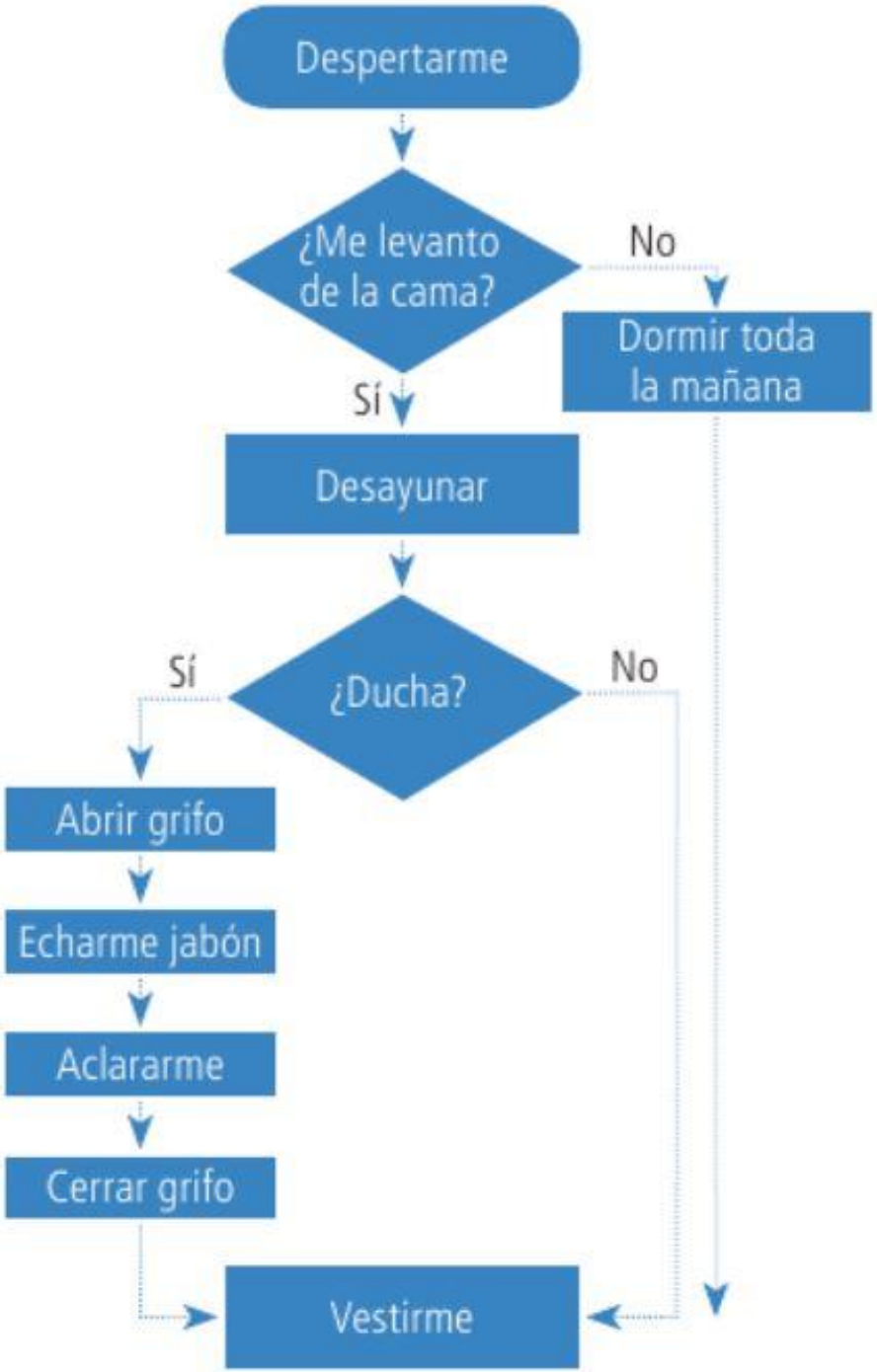
- ¿HAY CAFÉ PREPARADO?
- ¿ESTÁ SUFICIENTEMENTE DULCE?



https://www.lucidchart.com/

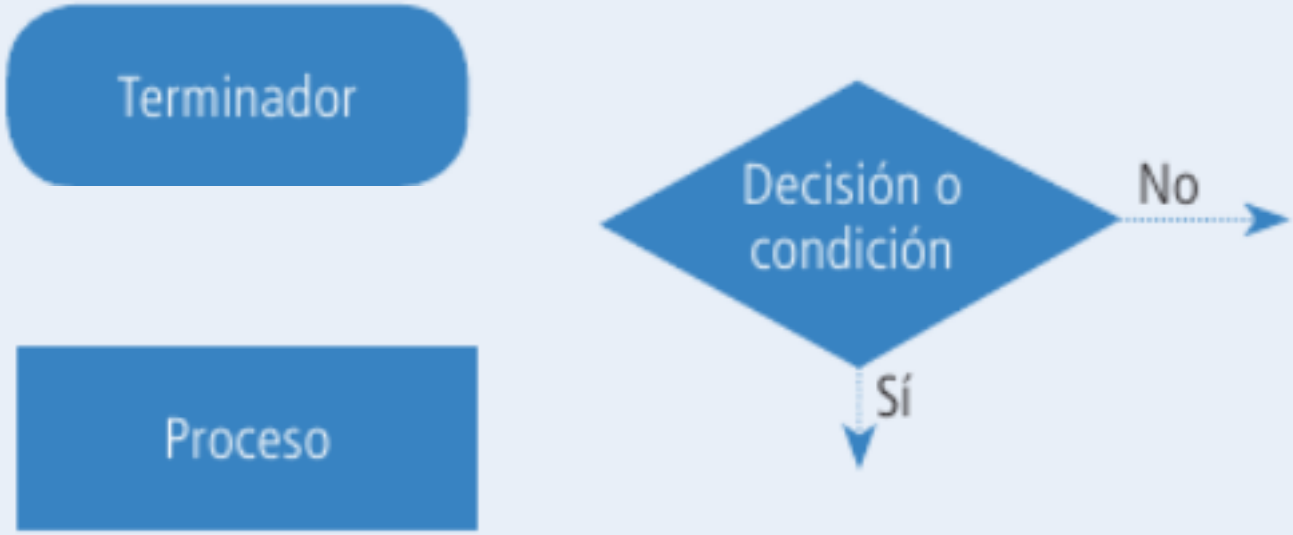
SOLUCIÓN

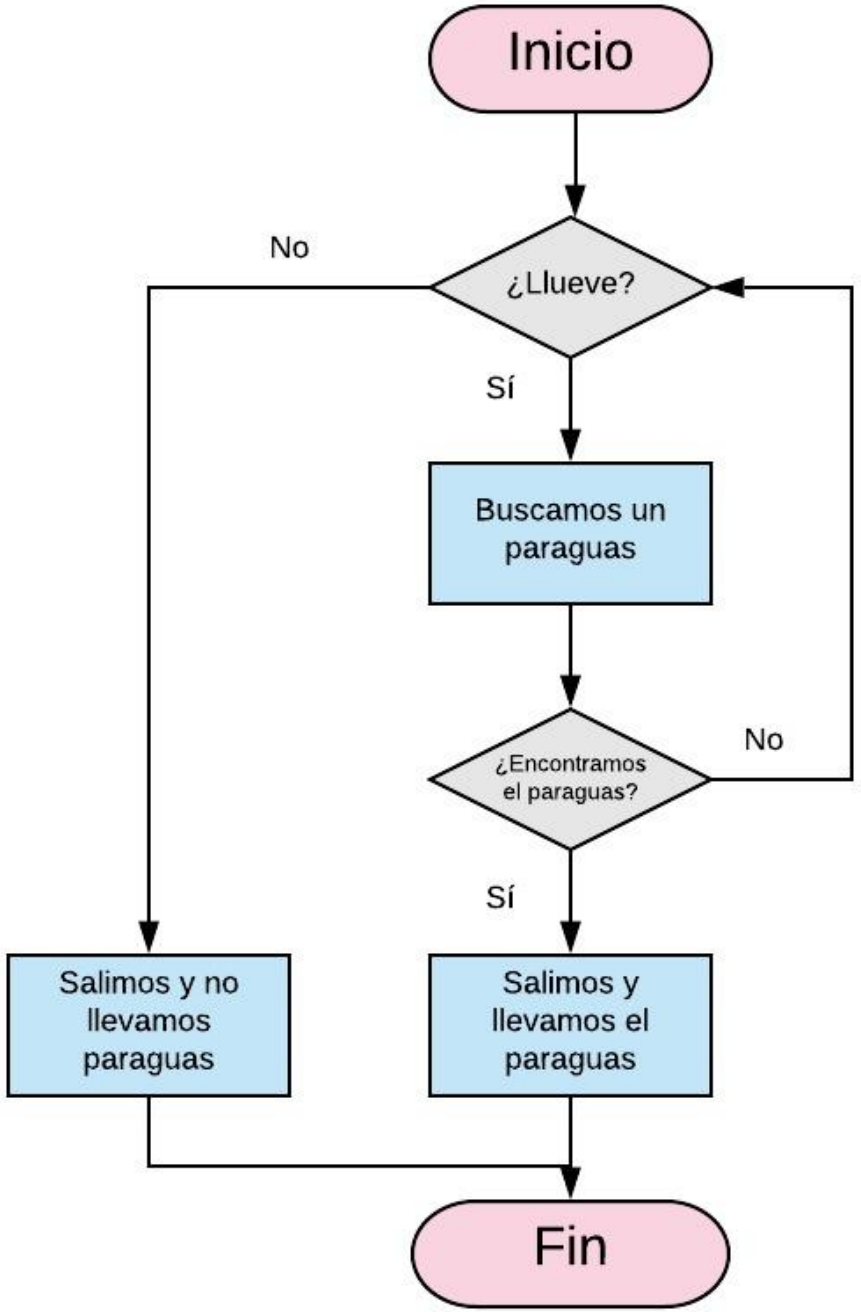

Quiero café
¿Hay café hecho?
No
Si
Hacer café
Calentar café
Servir en taza y añadir azucar
¿Está dulce?
No
Si
Añadir azucar
Tomar el café



2 - REALIZA UN DIAGRAMA DE FLUJO: VERIFICA SI AL SALIR LLEVO O NO PARAGUAS.

- ¿LLUEVE?
- ¿ENCONTRÉ EL PARAGUAS?

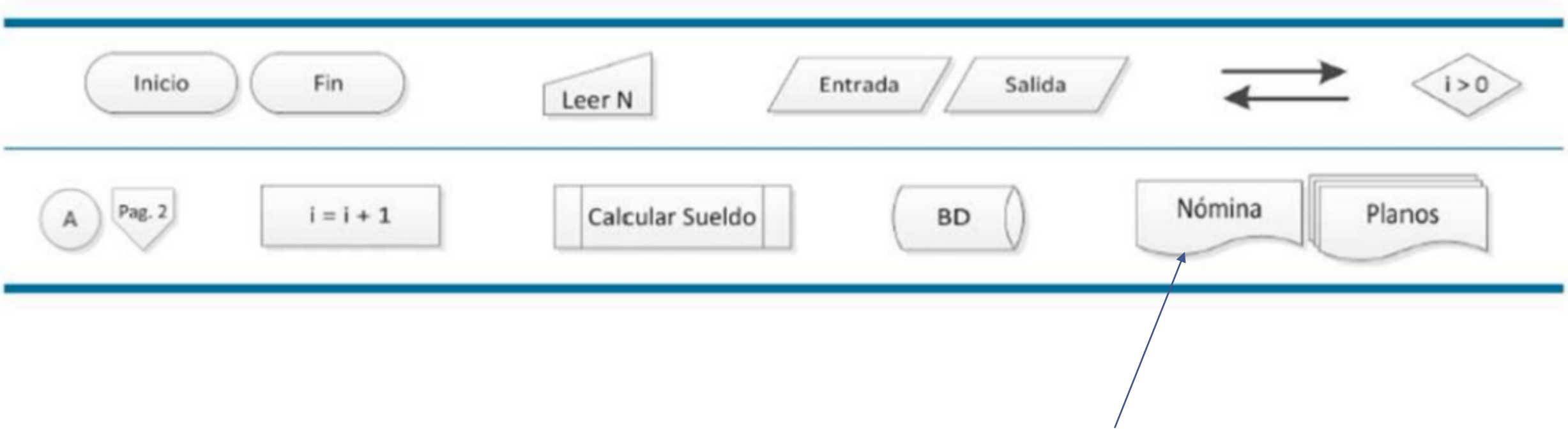



Inicio
¿Llueve?
No
Sí
Buscamos un paraguas
¿Encontramos el paraguas?
No
Sí
Salimos y no llevamos paraguas
Salimos y llevamos el paraguas
Fin

SOLUCIÓN

Diagramas de flujo

- Representación gráfica de un algoritmo o proceso.
- Simbología:



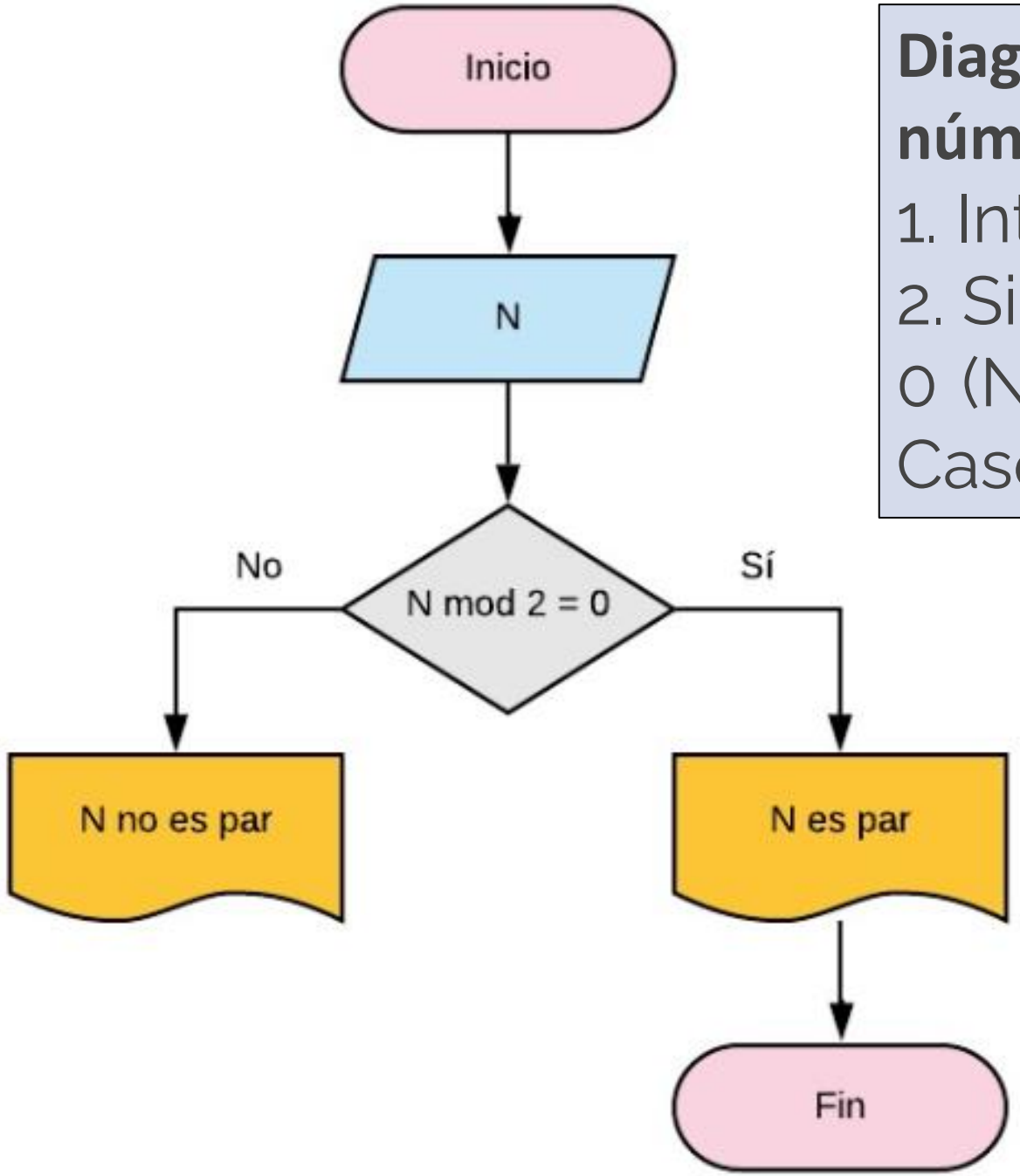
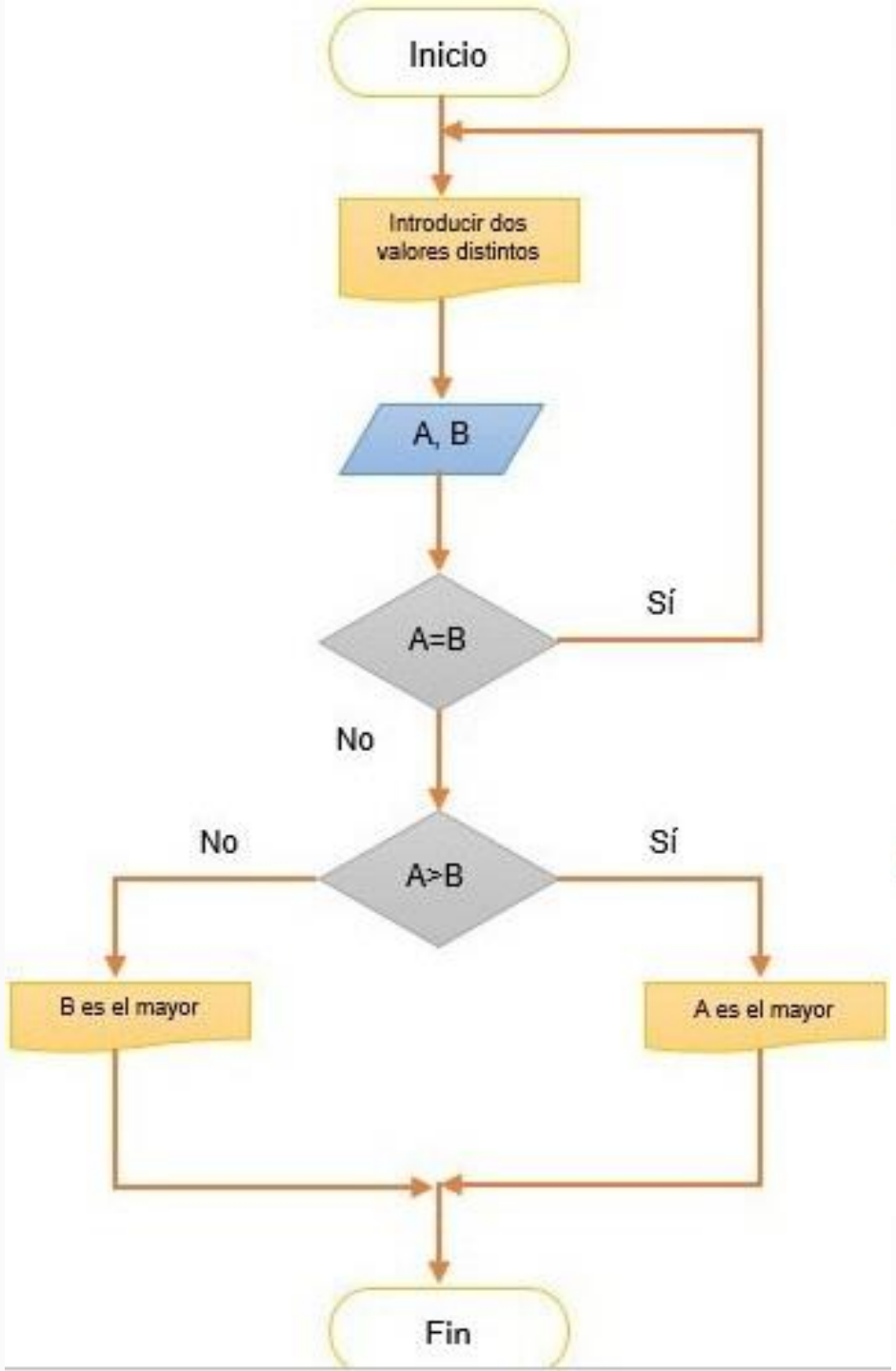


Diagrama de Flujo para saber si un número es par o impar

1. Introducimos un número “N”
2. Si “N” se divide entre 2 y el resto es 0 (N mod 2 = 0), entonces “N” es par. Caso contrario es impar

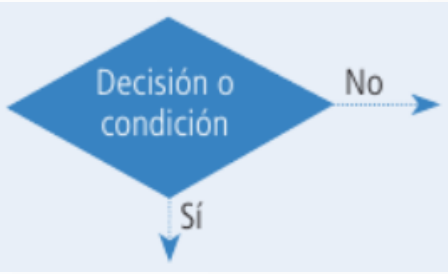
Entrada General
Entrada/Salida de datos en General(en esta guía, solo la usaremos para la Entrada).

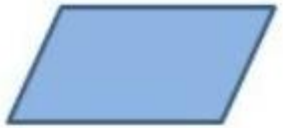
Salida Impresa
Indica la presentación de uno o varios resultados en forma impresa.



EJERCICIO RESUELTO:
Diagrama de flujo con la comparativa de dos números y devuelva el mayor

Inicio/Final
Se utiliza para indicar el inicio y el final de un diagrama; del Inicio sólo puede salir una línea de flujo y al Final sólo debe llegar una línea.





Entrada General
Entrada/Salida de datos en General(en esta guía, solo la usaremos para la Entrada).

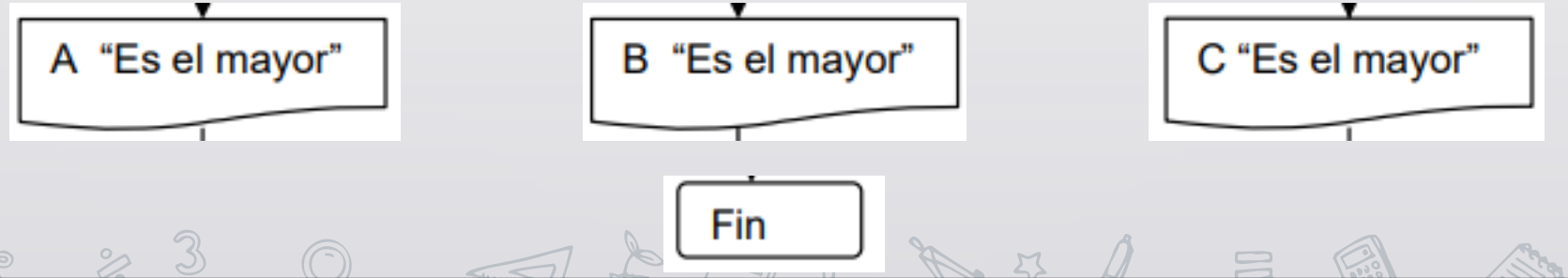
Salida Impresa
Indica la presentación de uno o varios resultados en forma impresa.

EJERCICIO: Realiza un diagrama de flujo con la comparativa de tres números y devuelva el mayor

PISTA

1. Introducir tres valores: “A” , “B” y “C”
2. Leer los valores
3. Si A > B y A> C entonces escribimos que “A” es el mayor…
4. Si B > A Y B > C entonces…

…



Programación estructurada

- Origen en la década de los 60.
- Desciende del paradigma de programación imperativa.
- Mejora la calidad del software.
- Agiliza el desarrollo.
- Reduce costes.
- Presenta dificultades a la hora de abordar proyectos de gran tamaño.

Programación estructurada

- Basado en el *teorema del programa estructurado* (*Corrado* Böhm - Giuseppe Jacopini) según el cual todo programa se puede escribir empleando únicamente 3 tipos básicos de estructuras de control:

  - *Secuencial*: las instrucciones se ejecutan una tras otra.
  - *Alternativa*: las expresiones son evaluadas y dependiendo del resultado se decide cual será la siguiente en ser ejecutada.
  - *Iterativa*: se repetirá un conjunto de instrucciones hasta que una condición sea cierta.

Pseudocódigo

- Técnica basada en un lenguaje cercano a un lenguaje de programación, cuyo objetivo es el desarrollo de algoritmos fácilmente interpretables por un programador.
- No se trata de un lenguaje de programación real, aunque utiliza un conjunto limitado de expresiones que permiten representar las estructuras de control y los módulos descritos en los paradigmas de programación estructurada y modular.
- No existe una sintaxis estándar para su escritura.

Pseudocódigo

Aritméticos		Relacionales (Usados para formar condiciones)		Lógicos (Usados para formar condiciones)	
+	Suma				
-	Resta	=	Igual	and	y lógico (conjunción)
*	Multiplicación	<	Menor	or	o lógico (disyunción)
/	División real	≤	Menor o igual	no	Negación lógica
div	División entera	>	Mayor	Especiales	
mod o ÷	Resto o módulo	≥	Mayor o igual	←	Asignación
^	Potencia	<>	Distinto	//	Comentario

Pseudocódigo

Palabras reservadas

Inicio	Si no	Otro	Para	En
Fin	Según	Mientras	Hasta	Procedimiento
Si	Hacer	Repetir	Incremento	Función
Entonces	Caso	Hasta que	Cada	Imprimir
Leer	Retornar			

Tipos de datos

Carácter	Cadena	Entero	Real	Booleano

Pseudocódigo - Estructuras de control secuenciales

Describen bloques de instrucciones que son ejecutadas en orden de aparición (secuencialmente).
Los bloques pueden estar delimitados por las expresiones *Inicio-Fin* o estar contenidos en otras estructuras.

```
Inicio
  <instrucción1>
      …
  <instrucciónN>
Fin
```

Dadas
dos variables numéricas
A y B,
que el usuario debe
teclear.

Se pide realizar un
algoritmo en
pseudocódigo que
intercambie los valores
de ambas variables y
muestre cuanto valen al
final las dos variables

(recuerda la asignación).

```
Proceso ejercicio_1
    Escribir "Introduce el valor de A"
    Leer A
    Escribir "Introduce el valor de B"
    Leer B
    C<-A
    A<-B
    B<-C
    Escribir "A vale " A " y B vale " B
FinProceso
```

Algoritmo que lea dos números, calculando y escribiendo:

el valor de su
- suma
- resta
- producto
- división

Comentarios → //

```
Proceso ejercicio_2
    Escribir "Introduce el primer numero"
    Leer numero1
    Escribir "Introduce el segundo numero"
    Leer numero2
    //inicializamos la variable resultado a 0 (recomendable)
    resultado<-0
    //sumamos los numeros y escribimos su resultado
    resultado<-numero1+numero2
    Escribir resultado
    //restamos los numeros y escribimos su resultado
    resultado<-numero1-numero2
    Escribir resultado
    //multiplicamos los numeros y escribimos su resultado
    resultado<-numero1*numero2
    Escribir resultado
    //dividimos los numeros y escribimos su resultado
    resultado<-numero1/numero2
    Escribir resultado
FinProceso
```

```
Leer numero1
Escribir "Introduce el segundo numero"
Leer numero2
//inicializamos la variable resultado a 0 (recomendable)
resultado<-0
//sumamos los numeros y escribimos su resultado
resultado<-numero1+numero2
Escribir resultado
//restamos los numeros y escribimos su resultado
resultado<-numero1-numero2
Escribir resultado
//multiplicamos los numeros y escribimos su resultado
resultado<-numero1*numero2
Escribir resultado
//dividimos los numeros y escribimos su resultado
resultado<-numero1/numero2
```

Análisis del problema

- Leer el radio de un circunferencia y calcular e imprimir su superficie y su longitud.

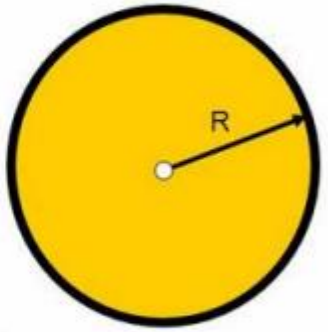
	Especificaciones
Entradas:	Radio de la circunferencia (Variable RADIO).
Salidas:	Superficie de la circunferencia (Variable SUPERFICIE). Longitud de la circunferencia (Variable LONGITUD)
Variables:	RADIO, SUPERFICIE, LONGITUD de tipo REAL.

CÍRCULO

PERIMETRO.

El perímetro de un círculo es la longitud de la circunferencia.

$$P = 2.\pi.R$$



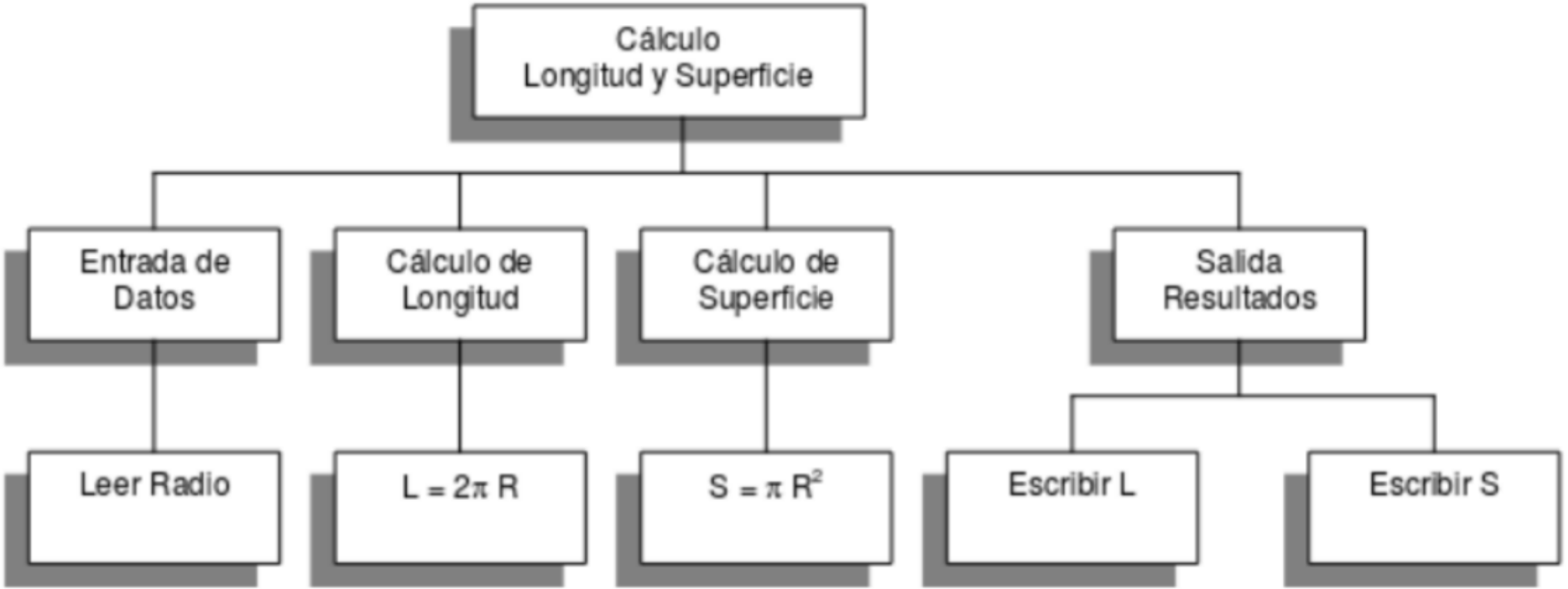
ÁREA

El área del círculo es la medida de la superficie que hay dentro de la circunferencia´.

$$A = \pi.r^2$$

ALGORITMOS

- DEBE SER: **PRECISOS**, **ESTAR DEFINIDOS** Y **FINITOS**
- **DISEÑO DE ALGORITMOS: 1º → DIVIDE Y VENERÁS**



Diseño de algoritmos

- Un **diagrama de flujo** es una de las técnicas de representación gráfica de algoritmos más antiguas.
- El **pseudocódigo**, nos permite una aproximación del algoritmo al lenguaje natural y por tanto una redacción rápida del mismo.

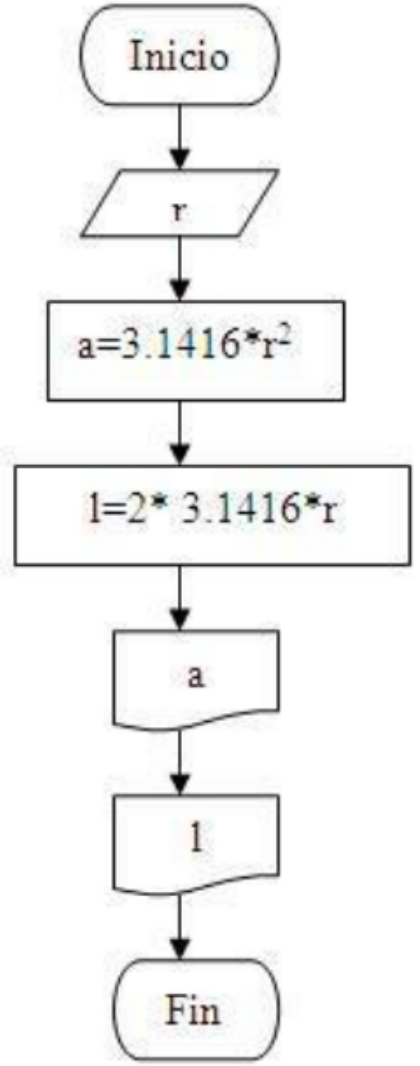
Fórmulas:

$a = 3.1416\, r^2$

$l = 2 * 3.1416 * r$

Pseudocódigo y Diagrama de flujo

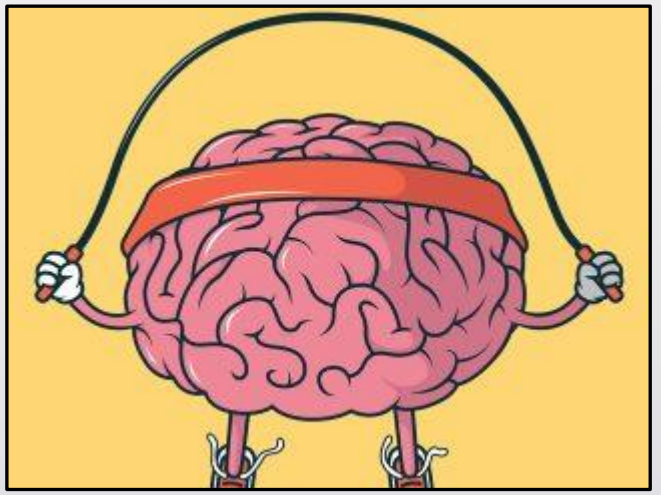
Inicio
  Leer r
  $a = 3.1416 * r^2$
  $l = 2 * 3.1416 * r$
  Escribir a
  Escribir l
Fin algoritmo



REPASO INICIAL

- Queremos un programa que solicite el perímetro de un rectángulo y además, calcule su área.

- Realiza:
  - un diagrama de flujo
  - el pseudocódigo del programa



Pseudocódigo - Estructuras de control alternativas

Encaminan el flujo de ejecución hacia un bloque de instrucciones u otro en base a la evaluación que se realiza sobre una condición determinada.

Simple

```
Si <condición> entonces
   <instrucción1>
       …
   <instrucciónX>
Fin Si
```

Doble

```
Si <condición> entonces
  <instrucción1>
       …
Si no
  <instrucción2>
       …
Fin Si
```

Algoritmo que lea tres números distintos y nos diga cual de ellos es el mayor

Usaremos la estructura condicional **Si** y los operadores **lógicos**.

```
Proceso ejercicio_4
    Escribir "Introduce el primer numero"
    Leer numero1
    Escribir "Introduce el segundo numero"
    Leer numero2
    Escribir "Introduce el tercer numero"
    Leer numero3
    //comparamos el numero1 con el numero2 y numero3
    //Si las dos condiciones son verdaderas el numero1 es el mayor
    Si (numero1>numero2 Y numero1>numero3) Entonces
       Escribir "el numero " numero1 " es el mayor"
       //si el numero1 no es el mayor,
       //comparamos el numero2 con el numero3
    Sino
       Si (numero2>numero3) Entonces
          Escribir "El numero " numero2 " es el mayor"
       Sino
          Escribir "El numero " numero3 " es el mayor"
       FinSi
    FinSi
FinProceso
```

Aritméticos		Relacionales (Usados para formar condiciones)		Lógicos (Usados para formar condiciones)	
+	Suma	=	Igual	and	y lógico (conjunción)
-	Resta	<	Menor	or	o lógico (disyunción)
*	Multiplicación	≤	Menor o igual	no	Negación lógica
/	División real	>	Mayor	**Especiales**	
div	División entera	≥	Mayor o igual	←	Asignación
mod o ÷	Resto o módulo	<>	Distinto	//	Comentario
^	Potencia				

Ejercicio 1:

- Diseñar un algoritmo que pida por teclado **tres números**; si el primero es negativo, debe imprimir el producto de los tres y si no lo es, imprimirá la suma.

Aritméticos		Relacionales (Usados para formar condiciones)		Lógicos (Usados para formar condiciones)	
+	Suma	=	Igual	and	y lógico (conjunción)
-	Resta	<	Menor	or	o lógico (disyunción)
*	Multiplicación	≤	Menor o igual	no	Negación lógica
/	División real	>	Mayor	**Especiales**	
div	División entera	≥	Mayor o igual	←	Asignación
mod o ÷	Resto o módulo	<>	Distinto	//	Comentario
^	Potencia				

SOLUCIÓN

```
Proceso ejercicio_5
   Escribir "Introduce el primer numero"
   Leer numero1
   Escribir "Introduce el segundo numero"
   Leer numero2
   Escribir "Introduce el tercer numero"
   Leer numero3
   //si el numero1 es menor que 0,
   //multiplicara los numero y sino los sumara
   Si (numero1<0) Entonces
       resultado<-numero1*numero2*numero3
   Sino
       resultado<-numero1+numero2+numero3
   FinSi
   Escribir resultado
FinProceso
```

Ejercicio 2:

- Un colegio desea saber qué porcentaje de niños y qué porcentaje de niñas hay en el curso actual. Diseñar un algoritmo para este propósito (recuerda que para calcular el porcentaje puedes hacer una regla de 3).

Aritméticos		Relacionales (Usados para formar condiciones)		Lógicos (Usados para formar condiciones)	
+	Suma	=	Igual	and	y lógico (conjunción)
-	Resta	<	Menor	or	o lógico (disyunción)
*	Multiplicación	≤	Menor o igual	no	Negación lógica
/	División real	>	Mayor	**Especiales**	
div	División entera	≥	Mayor o igual	←	Asignación
mod o ÷	Resto o módulo	<>	Distinto	//	Comentario
^	Potencia				

SOLUCIÓN

```
Proceso Ejercicio_7
    Escribir "Introduce el numero de niños"
    Leer numero_niños
    Escribir "Introduce el numero de niñas"
    Leer numero_niñas
    //calculamos el porcentaje
    porcentaje_niños<-numero_niños*100/(numero_niños+numero_niñas)
    porcentaje_niñas<-100-porcentaje_niños
    Escribir "Hay un " porcentaje_niños " % de niños
    Escribir "Hay un " porcentaje_niñas " % de niñas"
FinProceso
```

Ejercicio 3:

- Una tienda ofrece un descuento del 15% sobre el total de la compra durante el mes de **octubre**. Dado un mes y un importe, calcular cuál es la cantidad que se debe cobrar al cliente.

Aritméticos		Relacionales (Usados para formar condiciones)		Lógicos (Usados para formar condiciones)	
+	Suma	=	Igual	and	y lógico (conjunción)
-	Resta	<	Menor	or	o lógico (disyunción)
*	Multiplicación	≤	Menor o igual	no	Negación lógica
/	División real	>	Mayor	**Especiales**	
div	División entera	≥	Mayor o igual	←	Asignación
mod o ÷	Resto o módulo	<>	Distinto	//	Comentario
^	Potencia				

SOLUCIÓN

```
Proceso ejercicio_8
    Escribir "escribe el importe de la compra"
    Leer importe
    Escribir "Introduce el mes"
    Leer mes
    //Si el mes es octubre, se aplicara el descuento
    Si (mes="octubre") Entonces
       total<-importe*0.85
    Sino
       total<-importe
    FinSi
    Escribir total
FinProceso
```

Ejercicio 4:

- Realizar un algoritmo que dado un número entero, visualice en pantalla si es **par** o **impar**. En el caso de ser 0, debe visualizar "el número no es par ni impar" (para que un numero sea par, se debe dividir entre dos y que su resto sea 0)

Aritméticos		Relacionales (Usados para formar condiciones)		Lógicos (Usados para formar condiciones)	
+	Suma				
-	Resta	=	Igual	and	y lógico (conjunción)
*	Multiplicación	<	Menor	or	o lógico (disyunción)
/	División real	≤	Menor o igual	no	Negación lógica
div	División entera	>	Mayor	Especiales	
mod o ÷	Resto o módulo	≥	Mayor o igual	←	Asignación
^	Potencia	<>	Distinto	//	Comentario

SOLUCIÓN

```
Proceso ejercicio_9
    Escribir "Introduce un numero"
    Leer numero
    Si (numero=0) Entonces
        Escribir "El " numero " no es par ni impar"
    Sino
      //comprobamos si el numero es par
      Si (numero MOD 2=0) Entonces
          Escribir "El " numero " es par"
      Sino
          Escribir "El " numero " no es par"
      FinSi
    FinSi
FinProceso
```

Pseudocódigo - Estructuras de control alternativas

Múltiple →

```
Según <expresión> hacer
  Caso <valor1>
       <instrucciones1>
  Caso <valor2>
       <instrucciones2>
  Caso <valor3>
       <instrucciones3>
  Otro caso
       <instruccionesN>
Fin Según
```

```
Si <condición1> entonces
  <instrucciones1>
Si no
  Si <condición2> entonces
       <instrucciones2>
  Si no
       Si <condición3>
  entonces
                <instrucciones3>
       Si no
                <instruccionesN>
       Fin Si
       Fin Si
Fin Si
```

Pseudocódigo - Estructuras de control iterativas

Ejecutan un bloque de instrucciones mientras se cumpla una condición.

While

```
Mientras <condición>
  Hacer
  <instrucciones>
Fin Mientras
```

Repeat

```
Repetir
   <instrucciones>
Hasta Que
   <condición>
```

Do-While

```
Hacer
   <instrucciones>
Mientras
   <condición>
```

Ejecuta <instrucciones> como mínimo una vez

Ejercicio 5:

- Modificar el algoritmo anterior, de forma que si se teclea un cero, se vuelva a pedir el número por teclado (así hasta que se teclee un número mayor que cero)

Aritméticos		Relacionales (Usados para formar condiciones)		Lógicos (Usados para formar condiciones)	
+	Suma	=	Igual	and	y lógico (conjunción)
-	Resta	<	Menor	or	o lógico (disyunción)
*	Multiplicación	≤	Menor o igual	no	Negación lógica
/	División real	>	Mayor	**Especiales**	
div	División entera	≥	Mayor o igual	←	Asignación
mod o ÷	Resto o módulo	<>	Distinto	//	Comentario
^	Potencia				

SOLUCIÓN

```
Proceso ejercicio_10
    Escribir "Introduce un numero"
    Leer numero
    //Hasta que no se introduzca un numero mayor que 0 no saldra del bucle
    Mientras (numero<=0) hacer
        Escribir "escribe un numero mayor que 0"
        Leer numero
    FinMientras
    Si (numero MOD 2=0) Entonces
        Escribir "El " numero " es par"
    Sino
        Escribir "El " numero " no es par"
    FinSi
FinProceso
```

Ejercicio *Según, caso, caso, caso…fin según*:

- Escribe un programa que diga al usuario qué día de la semana es dado un número dado de día (1-7).
- Realiza todas las validaciones correspondientes.

Ejercicio 6:

- Algoritmo que nos diga si una persona puede acceder a cursar un ciclo formativo de grado superior o no. Para acceder a un grado superior, si se tiene un título de bachiller, en caso de no tenerlo, se puede acceder si hemos superado una prueba de acceso.

Aritméticos		Relacionales (Usados para formar condiciones)		Lógicos (Usados para formar condiciones)	
+	Suma	=	Igual	and	y lógico (conjunción)
-	Resta	<	Menor	or	o lógico (disyunción)
*	Multiplicación	≤	Menor o igual	no	Negación lógica
/	División real	>	Mayor	**Especiales**	
div	División entera	≥	Mayor o igual	←	Asignación
mod o ÷	Resto o módulo	<>	Distinto	//	Comentario
^	Potencia				

SOLUCIÓN

```
Proceso ejercicio_11
    Escribir "¿Tienes el titulo de bachiller?"
    Leer bachiller
    si (bachiller="si") Entonces
        Escribir "Puedes acceder al grado superior"
    Sino
        Escribir "¿Tienes la prueba de acceso superada?"
        Leer prueba_acceso
        si (prueba_acceso="si") Entonces
            Escribir "Puedes acceder al grado superior"
        Sino
            Escribir "No puedes acceder a un grado superior"
        FinSi
    FinSi
FinProceso
```

Pseudocódigo - Estructuras de control iterativas

For incremental

```
entero i
Para i ← 1 Hasta N
  Incremento 1 Hacer
  <instrucciones>
Fin Para
```

For decremental

```
entero i
Para i ← N Hasta 1 Incremento
   -1 Hacer
   <instrucciones>
Fin Para
```

Foreach

```
entero i
Para Cada elemento En conjunto
   Hacer
   <instrucciones>
Fin Para Cada
```

Ejercicio 7:

- Teniendo en cuenta que la clave es **“mariamoliner”**, escribe un algoritmo que nos pida una clave. Solo tenemos 3 intentos para acertar, si fallamos los 3 intentos nos mostrará un mensaje indicándonos que hemos agotado esos 3 intentos. (Recomiendo utilizar un interruptor o flag). Si acertamos la clave, saldremos directamente del programa

Aritméticos		Relacionales (Usados para formar condiciones)		Lógicos (Usados para formar condiciones)	
+	Suma	=	Igual	and	y lógico (conjunción)
-	Resta	<	Menor	or	o lógico (disyunción)
*	Multiplicación	≤	Menor o igual	no	Negación lógica
/	División real	>	Mayor	**Especiales**	
div	División entera	≥	Mayor o igual	←	Asignación
mod o ÷	Resto o módulo	<>	Distinto	//	Comentario
^	Potencia				

SOLUCIÓN

```
Proceso ejercicio_16
    contador<-0
    //interruptor
    acierto<-Falso
    //usamos un interruptor, cuando acertemos,
    //cambiara y la condicion sera falsa
    Mientras (contador<3 Y acierto=falso) Hacer
        //ponemos aqui leer porque con las variables
        //iniciales entra en el bucle
        Escribir "introduce la clave"
        Leer clave
        si (clave="mariamoliner") Entonces
            Escribir "la clave es correcta"
            //el interruptor cambia cuando acertamos
            acierto<-Verdadero
        FinSi
        contador<-contador+1
    FinMientras
    //este mensaje solo aparecera si hemos agotado
    //todos los intentos y no hemos acertado
    si (contador=3 Y acierto=falso) Entonces
         Escribir "Ya no tienes mas intentos"
    FinSi
FinProceso
```