

# LA RESOLUCIÓN DE PROBLEMAS CON COMPUTADORAS

---

9`dfcZYgcf`B`\_`U`g`K`|f`h`ž`|bj`Ybhc`f`XY`DUgWž`A`Cxi`U`&`mC`VYfcbž`h|hi`CEi`bc`XY`gi`g`a`zg`ZU`  
a`cgcg`"]Vfcg`5`[cf]`ha`cg`Ž`9ghfi`Wii`fUg`XY`XUhc`g`1`Dfc[fUa`Ug`gl`[b]Z|WbXcbcg`ei`Y`gCEc`gY`di`YXY`Y`!  
[UF`U`fYU]nUf`i`b`Vi`Yb`dfc[fUa`U`Wb`Y`X]gY`c`XY`i`b`U[cf]`ha`c`miY`i`gc`XY`WffY`WUg`Yghfi`Wii`fUg`  
XY`XUhc`g`9ghY`Wd`hi`c`hf`U`U`XY`cg`XUhc`g`mi`U`fYdf`YgYbhU`WCE`XY`U[cf]`ha`cg`a`YX|UbhY`YffUa`|Yb|  
hUg`XY`dfc[fUa`U`WCE`fX]U`fUa`Ug`XY`Zi`^cz`X]U`fUa`Ug`XY`B`!G`midgYi`Xc`WCE`|cL`

---

## 2.1. DATOS

Dato es la expresión general que describe los objetos con los cuales opera el algoritmo. El tipo de un dato determina su forma de almacenamiento en memoria y las operaciones que van a poder ser efectuadas con él. En principio hay que tener en cuenta que, prácticamente en cualquier lenguaje y por tanto en cualquier algoritmo, se podrán usar datos de los siguientes tipos:

- **entero.** Subconjunto finito de los números enteros, cuyo rango o tamaño dependerá del lenguaje en el que posteriormente codifiquemos el algoritmo y de la computadora utilizada.
- **real.** Subconjunto de los números reales limitado no sólo en cuanto al tamaño, sino también en cuanto a la precisión.
- **lógico.** Conjunto formado por los valores verdad y falso.
- **carácter.** Conjunto finito y ordenado de los caracteres que la computadora reconoce.
- **cadena.** Los datos (objetos) de este tipo contendrán una serie finita de caracteres, que podrán ser directamente traídos o enviados a/desde la consola.

Es decir, los tipos *entero*, *real*, *carácter*, *cadena* y *lógico* son *tipos predefinidos* en la mayoría de los lenguajes de programación. En los algoritmos para indicar que un dato es de uno de estos tipos se declarará utilizando directamente el identificador o nombre del tipo.

Además los lenguajes permiten al programador definir sus propios tipos de datos. Al definir nuevos tipos de datos hay que considerar tanto sus posibles valores como las operaciones que van a poder ser efectuadas con los datos del mismo. Lo usual es que se definan nuevos tipos de datos agrupando valores de otros tipos definidos previamente o de tipos estándar. Por este motivo se dice que están estructurados. Si todos los valores agrupados fueran del mismo tipo se consideraría a éste como el tipo base del nuevo tipo estructurado.

Los datos pueden venir expresados como constantes, variables, expresiones o funciones.

### 2.1.1. Constantes

Son datos cuyo valor no cambia durante todo el desarrollo del algoritmo. Las constantes podrán ser literales o con nombres, también denominadas simbólicas.

Las constantes simbólicas o con nombre se identifican por su nombre y el valor asignado. Una constante literal es un valor de cualquier tipo que se utiliza como tal. Tendremos pues constantes:

- **Numéricas enteras.** En el rango de los enteros. Compuestas por el signo (+,-) seguido de una serie de dígitos (0..9).
- **Numéricas reales.** Compuestas por el signo (+,-) seguido de una serie de dígitos (0..9) y un punto decimal (.) o compuestas por el signo (+,-), una serie de dígitos (0..9) y un punto decimal que constituyen la mantisa, la letra E antes del exponente, el signo (+,-) y otra serie de dígitos (0..9).
- **Lógicas.** Sólo existen dos constantes lógicas, **verdad** y **falso**.
- **Carácter.** Cualquier carácter del juego de caracteres utilizado colocado entre comillas simples o apóstrofes. Los caracteres que reconocen las computadoras son dígitos, caracteres alfabéticos, tanto mayúsculas como minúsculas, y caracteres especiales.
- **Cadena.** Serie de caracteres válidos encerrados entre comillas simples.

### 2.1.2. Variables

Una variable es un objeto cuyo valor puede cambiar durante el desarrollo del algoritmo. Se identifica por su nombre y por su tipo, que podrá ser cualquiera, y es el que determina el conjunto de valores que podrá tomar la variable. En los algoritmos se deben declarar las variables que se van a usar, especificando su tipo. Según la forma para la representación del algoritmo elegida la declaración se hará con una simple tabla de variables o de una forma algo más rígida, especificando:

```
var
    <tipo_de_dato> : <lista_identificadores_de_variable>
```

donde

<tipo\_de\_dato> representa un tipo de dato

y

<lista\_identificadores\_de\_variable> deberá ser sustituido por una lista con los nombres de las variables de dicho tipo separados por comas

Cuando se traduce el algoritmo a un lenguaje de programación y se ejecuta el programa resultante, la declaración de cada una de las variables originará que se reserve un determinado espacio en memoria etiquetado con el correspondiente identificador.

La asignación de valor a una variable se podrá hacer en modo interactivo mediante una instrucción de lectura o bien de forma interna a través del operador de asignación.

### 2.1.3. Expresiones

Una **expresión** es una combinación de operadores y operandos. Los operandos podrán ser constantes, variables u otras expresiones y los operadores de cadena, aritméticos, relacionales o lógicos. Las expresiones se clasifican, según el resultado que producen, en:

- **Núméricas.** Los operandos que intervienen en ellas son numéricos, el resultado es también de tipo numérico y se construyen mediante los operadores aritméticos. Se pueden considerar análogas a las fórmulas matemáticas.

Debido a que son los que se encuentran en la mayor parte de los lenguajes de programación, los algoritmos utilizarán los siguientes operadores aritméticos: menos unario (–), multiplicación (\*), división real (/), exponenciación (\*\*), adición (+), resta (–), módulo de la división entera (**mod**) y cociente de la división entera (**div**). Tenga en cuenta que la división real siempre dará un resultado real y que los operadores **mod** y **div** sólo operan con enteros y el resultado es entero.

- **Alfanuméricas.** Los operandos son de tipo alfanumérico y producen resultados también de dicho tipo. Se construyen mediante el operador de concatenación, representado por el operador *ampersand* (&) o con el mismo símbolo utilizado en las expresiones aritméticas para la suma.
- **Booleanas.** Su resultado podrá ser **verdad** o **falso**. Se construyen mediante los operadores relacionales y lógicos. Los operadores de relación son: igual (=), distinto (<>), menor (<), mayor (>), mayor o igual (>=), menor o igual (<=). Actúan sobre operandos del mismo tipo y siempre devuelven un resultado de tipo lógico. Los operadores lógicos básicos son: negación lógica (**no**), multiplicación lógica (**y**), suma lógica (**o**). Actúan sobre operandos de tipo lógico y devuelven resultados del mismo tipo, determinados por las tablas de verdad correspondientes a cada uno de ellos.

a	b	no a	a y b	a o b
verdad	verdad	falso	verdad	verdad
verdad	falso	falso	falso	verdad
falso	verdad	verdad	falso	verdad
falso	falso	verdad	falso	falso

El orden de prioridad general adoptado, no común a todos los lenguajes, es el siguiente:

**	Exponenciación
no, –	Operadores unarios
*, /, <b>div</b> , <b>mod</b> , <b>y</b>	Operadores multiplicativos
+, –, <b>o</b>	Operadores aditivos
=, <>, >, <, >=, <=	Operadores de relación

La evaluación de operadores con la misma prioridad se realizará siempre de izquierda a derecha. Si una expresión contiene subexpresiones encerradas entre paréntesis, dichas subexpresiones se evaluarán primero.

## 2.1.4. Funciones

En los lenguajes de programación existen ciertas funciones predefinidas o internas que aceptan unos argumentos y producen un valor denominado resultado. Como funciones numéricas, se usarán:

Función	Descripción	Tipo de argumento	Resultado
abs (x)	valor absoluto de x	entero o real	igual que el argumento
arctan (x)	arcotangente de x	entero o real	real
cos (x)	coseno de x	entero o real	real
cuadrado (x)	cuadrado de x	entero o real	igual que el argumento
ent (x)	entero de x	real	entero
exp (x)	e elevado a x	entero o real	real

(continúa)

(continuación)

Función	Descripción	Tipo de argumento	Resultado
$\ln(x)$	logaritmo neperiano de $x$	entero o real	real
$\log_{10}(x)$	logaritmo base 10 de $x$	entero o real	real
$\text{raíz2}(x)$	raíz cuadrada de $x$	entero de $x$	real
$\text{redondeo}(x)$	redondea $x$ al entero más proximo	real	entero
$\text{sen}(x)$	seno de $x$	entero o real	real
$\text{trunc}(x)$	parte entera de $x$	real	entero

Las funciones se utilizarán escribiendo su nombre, seguido de los argumentos adecuados encerrados entre paréntesis, en una expresión.

2.2. REPRESENTACIÓN DE ALGORITMOS

Un algoritmo puede ser escrito en castellano narrativo, pero esta descripción suele ser demasiado prolija y, además, ambigua. Para representar un algoritmo se debe utilizar algún método que permita independizar dicho algoritmo de los lenguajes de programación y, al mismo tiempo, conseguir que sea fácilmente codificable.

Los métodos más usuales para la representación de algoritmos son:

- A. Diagrama de flujo.
- B. Diagrama N-S (Nassi-Schneiderman).
- C. *Pseudocódigo*.


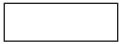
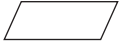
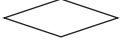
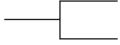
2.3. DIAGRAMA DE FLUJO

Los diagramas de flujo se utilizan tanto para la representación gráfica de las operaciones ejecutadas sobre los datos a través de todas las partes de un sistema de procesamiento de información, diagrama de flujo del sistema, como para la representación de la secuencia de pasos necesarios para describir un procedimiento particular, diagrama de flujo de detalle.

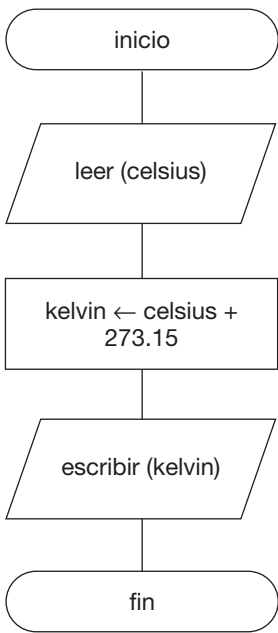
En la actualidad se siguen usando los diagramas de flujo del sistema, pero ha decaído el uso de los diagramas de flujo de detalle al aparecer otros métodos de diseño estructurado más eficaces para la representación y actualización de los algoritmos.

Los diagramas de flujo de detalle son, no obstante, uno de nuestros objetivos prioritarios y, a partir de ahora, los denominaremos simplemente diagramas de flujo.

El diagrama de flujo utiliza unos símbolos normalizados, con los pasos del algoritmo escritos en el símbolo adecuado y los símbolos unidos por flechas, denominadas líneas de flujo, que indican el orden en que los pasos deben ser ejecutados. Los símbolos principales son:

Símbolo	Función
	Inicio y fin del algoritmo
	Proceso
	Entrada/Salida
	Decisión
	Comentario

Resulta necesario indicar dentro de los símbolos la operación específica concebida por el programador. Como ejemplo veamos un diagrama de flujo básico, que representa la secuencia de pasos necesaria para que un programa lea una temperatura en grados Centígrados y calcule y escriba su valor en grados Kelvin.



2.4. DIAGRAMA NASSI-SCHNEIDERMAN

Los diagramas Nassi-Schneiderman, denominados así por sus inventores, (o también N-S, o de Chapin) son una herramienta de programación que favorece la programación estructurada y reúne características gráficas propias de diagramas de flujo y lingüísticas propias de los pseudocódigos. Constan de una serie de cajas contiguas que se leerán siempre de arriba-abajo y se documentarán de la forma adecuada. En los diagramas N-S las tres estructuras básicas de la programación estructurada, secuenciales, selectivas y repetitivas, encuentran su representación propia. La programación estructurada será tratada en capítulos posteriores.

Símbolo	Tipo de estructura
<div>Sentencia 1</div> <div>Sentencia 2</div> <div>Sentencia n</div>	Secuencial
<div><div></div><div></div></div>	Repetitiva de 0 a n veces
<div><div></div><div></div></div>	Repetitiva de 1 a n veces
<div><div></div><div></div></div>	Repetitiva n veces
<div><div>condición</div><div>si</div><div>no</div></div>	Selectiva

Copyright © 2003. McGraw-Hill España. All rights reserved.

El algoritmo que lee una temperatura en grados Celsius y calcula y escribe su valor en grados Kelvin se puede representar mediante el siguiente diagrama N-S:

inicio
leer (celsius)
$\text{kelvin} \leftarrow \text{celsius} + 273.15$
escribir (kelvin)
fin

## 2.5. PSEUDOCÓDIGO

El pseudocódigo es un lenguaje de especificación de algoritmos que utiliza palabras reservadas y exige la indentación, o sea sangría en el margen izquierdo de algunas líneas. El pseudocódigo se concibió para superar las dos principales desventajas del diagrama de flujo: lento de crear y difícil de modificar sin un nuevo redibujo. Es una herramienta muy buena para el seguimiento de la lógica de un algoritmo y para transformar con facilidad los algoritmos a programas, escritos en un lenguaje de programación específico. En este libro se escribirán casi todos los algoritmos en pseudocódigo.

En nuestro pseudocódigo utilizaremos palabras reservadas en español. Así, nuestros algoritmos comenzarán con la palabra reservada **inicio** y terminarán con **fin**, constando de múltiples líneas que se sangran o indentan para mejorar la legibilidad. La estructura básica de un algoritmo escrito en pseudocódigo es:

```

algoritmo <identificador_algoritmo>
    // declaraciones, sentencias no ejecutables
inicio
    // acciones, sentencias ejecutables tanto simples como estructuradas
fin

```

Los espacios en blanco entre los elementos no resultan significativos, y las partes importantes se suelen separar unas de otras por líneas en blanco. Para introducir un valor o serie de valores desde el dispositivo estándar y almacenarlos en una o varias variables utilizaremos **leer**(<lista\_de\_variables>)<sup>1</sup>. Con **nombre\_de\_variable**  $\leftarrow$  <expresión> almacenaremos en una variable el resultado de evaluar una expresión. Hay que tener en cuenta que una única constante, variable o función, constituyen una expresión. Para imprimir en el dispositivo estándar de salida una o varias expresiones emplearemos **escribir**(<lista\_de\_expresiones>)<sup>2</sup>.

Los elementos léxicos de nuestro pseudocódigo son: *Comentarios, Palabras reservadas, Identificadores, Operadores y signos de puntuación y Literales*.

### 2.5.1 Comentarios

Los *comentarios* sirven para documentar el algoritmo y en ellos se escriben anotaciones generalmente sobre su funcionamiento. Cuando se coloque un comentario de una sola línea se escribirá precedido de //. Si el comentario es multilinea, lo pondremos entre {}.

```

// Comentario de una línea
{ Comentario que ocupa más
  de una línea }

```

<sup>1</sup> La expresión <lista\_de\_variables> deberá ser sustituida por una lista de variables separadas por comas.

<sup>2</sup> La expresión <lista\_de\_expresiones> deberá ser sustituida por una lista de expresiones separadas por comas.

## 2.5.2. Palabras reservadas

Las *palabras reservadas* o *palabras clave* (*Keywords*) son palabras que tienen un significado especial, como: **inicio** y **fin**, que marcan el principio y fin del algoritmo, y las palabras que aparecen en negrita en las estructuras especificadas a continuación. En lugar de las palabras reservadas no deberán utilizarse otras similares, aunque no se distinga entre mayúsculas y minúsculas.

*Decisión simple:*      **si** <condición> **entonces**  
    <acciones1>  
    **fin\_si**

*Decisión doble:*      **si** <condición> **entonces**  
    <acciones1>  
    **si\_no**  
    <acciones2>  
    **fin\_si**

*Decisión múltiple:*    **según\_sea** <expresión\_ordinal> **hacer**  
    <lista\_de\_valores\_ordinales>: <acciones1>  
    .....  
    [**si\_no**            // El corchete indica opcionalidad  
    <accionesN>]  
    **fin\_según**

*Repetitivas:*            **mientras** <condición> **hacer**  
    <acciones>  
    **fin\_mientras**  
  
    **repetir**  
    <acciones>  
    **hasta\_que** <condición>  
  
    **desde** <variable> ← <v\_inicial> **hasta** <v\_final>  
               [**incremento** | **decremento** <incremento>] **hacer**  
               <acciones>  
    **fin\_desde**

El ejemplo ya citado que transforma grados Celsius en Kelvin, escrito en pseudocódigo quedaría de la siguiente forma:

```
inicio
    leer(celsius)
    kelvin ← celsius + 273.15
    escribir(Kelvin)
fin
```

## 2.5.3. Identificadores

**Identificadores** son los nombres que se dan a las constantes simbólicas, variables, funciones, procedimientos, u otros objetos que manipula el algoritmo. La regla para construir un identificador establece que:

- Debe resultar significativo, sugiriendo lo que representa.
- No podrá coincidir con palabras reservadas, propias del lenguaje algorítmico. Como se verá más adelante, la representación de algoritmos mediante pseudocódigo va a requerir la utilización de palabras reservadas.
- Se recomienda un máximo de 50 caracteres.
- Comenzará siempre por un carácter alfabético y los siguientes podrán ser letras, dígitos o el símbolo de subrayado.
- Podrá ser utilizado indistintamente escrito en mayúsculas o en minúsculas.

## 2.5.4. Operadores y signos de puntuación

Los operadores se utilizan en las expresiones e indican las operaciones a efectuar con los operandos, mientras que los signos de puntuación se emplean con el objetivo de agrupar o separar, por ejemplo . ; o [ ] .

## 2.5.5. Literales

Los literales son los valores que aparecen directamente escritos en el programa y pueden ser literales: lógicos, enteros, reales, de tipo carácter, de tipo cadena y el literal `nilo`.

## 2.6. EJERCICIOS RESUELTOS

**2.1.** ¿Cuál de los siguientes datos son válidos para procesar por una computadora?

- |             |             |            |
|-------------|-------------|------------|
| a) 3.14159  | e) 2.234E2  | i) 12.5E.3 |
| b) 0.0014   | f) 12E+6    | j) .123E4  |
| c) 12345.0  | g) 1.1E-3   | k) 5A4.14  |
| d) 15.0E-04 | h) -15E-0.4 | l) A1.E04  |

Serían válidos los datos a, b, c, d, e, f, g y j. Los datos h e i no serían válidos pues el exponente no puede tener forma decimal. El k, no sería correcto pues mezcla caracteres alfabéticos y dígitos, y no se puede considerar una identificador de una variable ya que empieza por un dígito. El dato l aunque mezcla dígitos y caracteres alfabéticos, podría ser un identificador, si admitiéramos el carácter punto como carácter válido para una variable (PASCAL o BASIC lo consideran).

**2.2.** ¿Cuál de los siguientes identificadores son válidos?

- |              |                 |                     |
|--------------|-----------------|---------------------|
| a) Renta     | e) Dos Pulgadas | i) 4A2D2            |
| b) Alquiler  | f) C3PO         | j) 13Nombre         |
| c) Constante | g) Bienvenido#5 | k) Nombre_Apellidos |
| d) Tom's     | h) Elemento     | l) NombreApellidos  |

Se consideran correctos los identificadores a, b, c, f, h, k y l. El d no se considera correcto pues incluye el apóstrofo que no es un carácter válido en un identificador. Lo mismo ocurre con el e y el espacio en blanco, y el g con el carácter #. El i y el j no serían válidos al no comenzar por un carácter alfabético.

**2.3.** Escribir un algoritmo que lea un valor entero, lo doble, se multiplique por 25 y visualice el resultado.

### Análisis de problema

DATOS DE SALIDA: Resultado (es el resultado de realizar las operaciones)  
 DATOS DE ENTRADA: Número (el número que leemos por teclado)



Leemos el número por teclado y lo multiplicamos por 2, metiendo el contenido en la propia variable de entrada. A continuación lo multiplicamos por 25 y asignamos el resultado a la variable de salida resultado. También podríamos asignar a la variable de salida directamente la expresión  $\text{número} * 2 * 25$ .

### Diseño del algoritmo

```
algoritmo ejercicio_2_3
var
    entero : número, resultado
inicio
    leer (número)
    número ← número * 2
    resultado ← número * 25
    escribir (resultado)
fin
```

- 2.4.** Diseñar un algoritmo que lea cuatro variables y calcule e imprima su producto, su suma y su media aritmética.

### Análisis del problema

DATOS DE SALIDA: Producto, suma y media

DATOS DE ENTRADA: A, b, c, d

Después de leer los cuatro datos, asignamos a la variable producto la multiplicación de las cuatro variables de entrada. A la variable suma le asignamos su suma y a la variable media le asignamos el resultado de sumar las cuatro variables y dividirlos entre cuatro. Como el operador suma tiene menos prioridad que el operador división, será necesario encerrar la suma entre paréntesis. También podríamos haber dividido directamente la variable suma entre cuatro.

La variables a, b, c, d, producto y suma podrán ser enteras, pero no así la variable media, ya que la división produce siempre resultados de tipo real.

### Diseño del algoritmo

```
algoritmo ejercicio_2__4
var
    entero: a,b,c,d,producto,suma
    real: media
inicio
    leer (a,b,c,d)
    producto ← a * b * c * d
    suma ← a + b + c + d
    media ← (a + b + c + d) / 4
    escribir (producto,suma,media)
fin
```

- 2.5.** Diseñar un programa que lea el peso de un hombre en libras y nos devuelva su peso en kilogramos y gramos (Nota: una libra equivale a 0.453592 kilogramos).

### Análisis del problema

DATOS DE SALIDA: Kg, gr

DATOS DE ENTRADA: Peso

DATOS AUXILIARES: Libra (los kilogramos que equivalen a una libra)

El dato auxiliar libra lo vamos a considerar como una constante, pues no variará a lo largo del programa. Primero leemos el peso. Para hallar su equivalencia en kilogramos multiplicamos éste por la constante libra. Para hallar el peso en gramos multiplicamos los kilogramos entre mil. Como es posible que el dato de entrada no sea exacto, consideraremos todas las variables como reales.

### Diseño del algoritmo

```
algoritmo ejercicio_2_5
const
    libra = 0.453592
var
    real: peso, kg, gr
inicio
    leer (peso)
    kg ← peso * libra
    gr ← kg * 1000
    escribir ('Peso en kilogramos: ', kg)
    escribir ('Peso en gramos: ', gr)
fin
```

2.6. Si  $A = 6$ ,  $B = 2$  y  $C = 3$ , encontrar los valores de las siguientes expresiones:

- |                                 |  |
|---------------------------------|--|
| a) $A - B + C$                  | d) $A * B \bmod C$                               |
| b) $A * B \operatorname{div} C$ | e) $A + B \bmod C$                               |
| c) $A \operatorname{div} B + C$ | f) $A \operatorname{div} B \operatorname{div} C$ |

Los resultados serían:

- a)  $(6 - 2) + 3 = 7$   
 b)  $(6 * 2) \operatorname{div} 3 = 4$   
 c)  $(6 \operatorname{div} 2) + 3 = 6$   
 d)  $(6 * 2) \bmod 3 = 0$   
 e)  $6 + (2 \bmod 3) = 8$   
 f)  $(6 \operatorname{div} 2) \operatorname{div} 3 = 1$

2.7. ¿Qué se obtiene en las variables A y B después de la ejecución de las siguientes instrucciones?

```
A ← 5
B ← A + 6
A ← A + 1
B ← A - 5
```

Las variables irían tomando los siguientes valores:

```
A ← 5      ,   A = 5      ,   B indeterminado
B ← A + 6   ,   A = 5      ,   B = 5 + 6 = 11
A ← A + 1   ,   A = 5 + 1 = 6 ,   B = 11
B ← A - 5   ,   A = 6      ,   B = 6 - 5 = 1
```

Los valores finales serían:  $A = 6$  y  $B = 1$ .

2.8. ¿Qué se obtiene en las variables A, B y C después de ejecutar las siguientes instrucciones?

```
A ← 3
B ← 20
C ← A + B
B ← A + B
A ← B
```

Las variables tomarían sucesivamente los valores:

```
A ← 3      , A = 3      , B y C indeterminados
B ← 20     , A = 3      , B = 20      , C indeterminado
C ← A + B  , A = 3      , B = 20      , C = 3 + 20 = 23
B ← A + B  , A = 3      , B = 3 + 20 = 23 , C = 23
A ← B      , A = 23     , B = 23      , C = 23
```

Los valores de las variables serían: A = 23, B = 23 y C = 23.

**2.9.** ¿Qué valor toman las variables A y B tras la ejecución de las siguientes asignaciones?

```
A ← 10
B ← 5
A ← B
B ← A
```

El resultado sería el siguiente:

```
A ← 10 , A = 10 , B indeterminada
B ← 5  , A = 10 , B = 5
A ← B  , A = 5  , B = 5
B ← A  , A = 5  , B = 5
```

con lo que A y B tomarían el valor 5.

**2.10.** Escribir las siguientes expresiones en forma de expresiones algorítmicas.

$$\begin{array}{lll} a) \frac{M}{N} + 4 & b) M + \frac{N}{P - Q} & c) \frac{\sin x \cos y}{\tan x} \\ d) \frac{M + N}{P - Q} & e) \frac{P + \frac{N}{P}}{Q - \frac{r}{5}} & f) \frac{-b + \sqrt{b^2 - 4ac}}{2a} \end{array}$$

```
a) M / N + 4
b) M + N / (P - Q)
c) (sen(X) + cos(X)) / tan(X)
d) (M + N) / (P - Q)
e) (M + N / P) / (Q - R / 5)
f) ((-B) + raíz2(B**2-4*A*C)) / (2*A)
```

**2.11.** Se tienen tres variables A, B y C. Escribir las instrucciones necesarias para intercambiar entre sí sus valores del modo siguiente:

```
B toma el valor de A
C toma el valor de B
A toma el valor de C
```

*Nota: sólo se debe utilizar una variable auxiliar que llamaremos AUX.*

```
AUX ← A
A ← C
C ← B
B ← AUX
```

2.12. *Deducir los resultados que se obtienen del siguiente algoritmo:*

```
algoritmo ejercicio_2_12
var
    entero: X, Y, Z
inicio
    X ← 15
    Y ← 30
    Z ← Y - X
    escribir (X,Y)
    escribir (Z)
fin
```

Para analizar los resultados de un algoritmo, lo mejor es utilizar una tabla de seguimiento. En la tabla de seguimiento aparecen varias columnas, cada una con el nombre de una de las variables que aparecen en el algoritmo, pudiéndose incluir además una con la salida del algoritmo. Más abajo se va realizando el seguimiento del algoritmo rellenando la columna de cada variable con el valor que ésta va tomando.

X	Y	Z	Salida
15	30	15	
			15,30
			15

2.13. *Encontrar el valor de la variable VALOR después de la ejecución de las siguientes operaciones:*

```
a) VALOR ← 4.0 * 5
b) X ← 3.0
   Y ← 2.0
   VALOR ← X ** Y - Y
c) VALOR ← 5
   X ← 3
   VALOR ← VALOR * X
```

Los resultados serían:

```
a) VALOR = 20.0
b) VALOR = 7.0
c) VALOR = 15
```

Nótese que en los casos *a)* y *b)*, valor tendrá un contenido de tipo real, ya que alguno de los operandos son de tipo real.

2.14. *Determinar los valores de A, B, C y D después de la ejecución de las siguientes instrucciones:*

```
algoritmo ejercicio_2_14
var
    entero: A, B, C, D
inicio
    A ← 1
    B ← 4
    C ← A + B
```

```

D ← A - B
A ← C + 2 * B
B ← C + B
C ← A * B
D ← B + D
A ← D + C
si C > D entonces
    C ← A - D
si_no
    C ← B - D
fin_si
fin

```

Realizaremos una tabla de seguimiento:

A	B	C	D
1	4	5	-3
13	9	117	6
123		117	

con lo que  $A = 123$ ,  $B = 9$ ,  $C = 117$  y  $D = 6$ .

### 2.15. Escribir un algoritmo que calcule y escriba el cuadrado de 821.

#### Análisis del problema

DATOS DE SALIDA: Cuadr (almacenará el cuadrado de 821)

DATOS DE ENTRADA: El propio número 821

Lo único que hará este algoritmo será asignar a la variable `cuadr` el cuadrado de 821, es decir,  $821 * 821$ .

#### Diseño del algoritmo

```

algoritmo ejercicio_2_15
var
    entero: cuadr
inicio
    cuadr ← 821 * 821
    escribir (cuadr)
fin

```

### 2.16. Realizar una llamada telefónica desde un teléfono público.

#### Análisis del problema

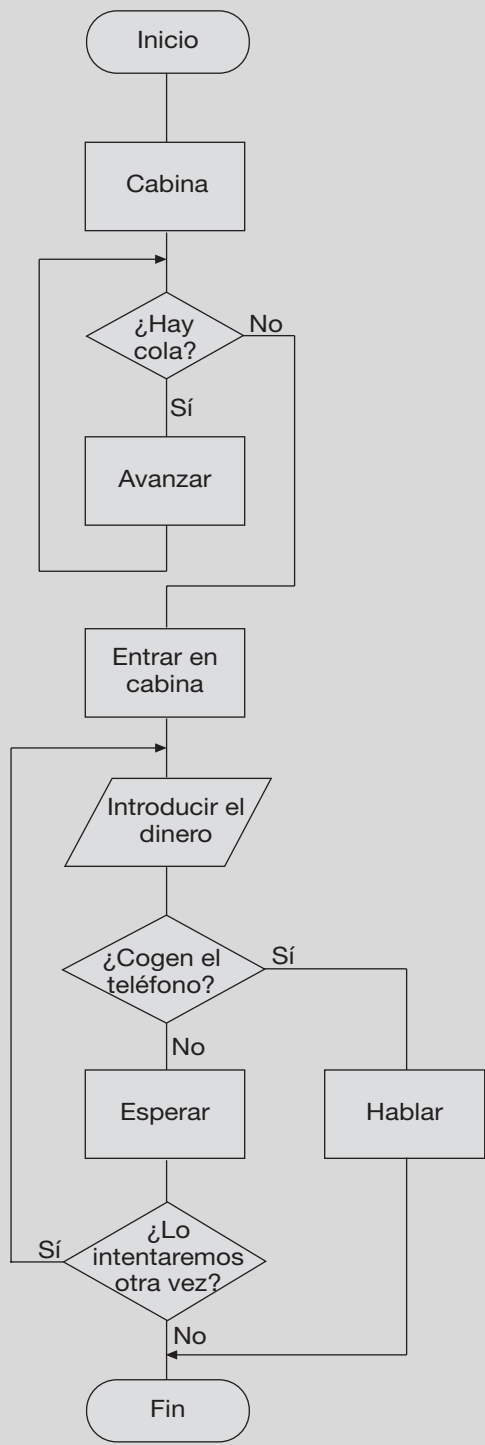
DATOS DE SALIDA: La comunicación por teléfono

DATOS DE ENTRADA: El número de teléfono, el dinero

DATOS AUXILIARES: Distintas señales de la llamada (comunicando, etc.)

Se debe ir a la cabina y esperar si hay cola. Entrar e introducir el dinero. Se marca el número y se espera la señal, si está comunicando o no contestan se repite la operación hasta que descuelgan el teléfono o decide irse.

*Diseño del algoritmo*



**2.17.** Realizar un algoritmo que calcule la suma de los enteros entre 1 y 10, es decir  $1+2+3+\dots+10$ .

#### Análisis del problema

DATOS DE SALIDA: suma (contiene la suma requerida)

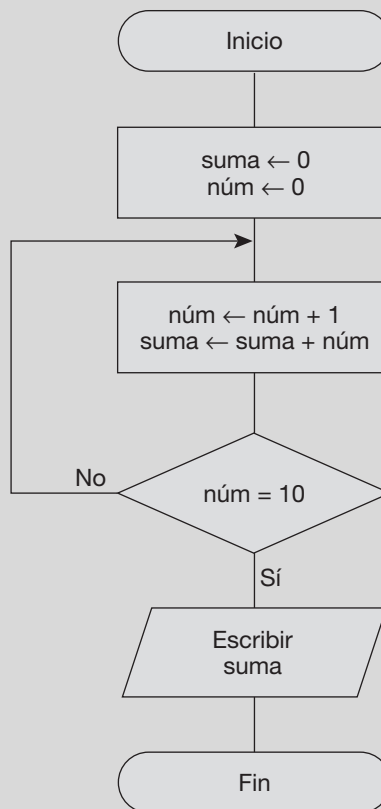
DATOS AUXILIARES: núm (será una variable que vaya tomando valores entre 1 y 10 y se acumulará en suma)

Hay que ejecutar un bucle que se realice 10 veces. En él se irá incrementando en 1 la variable núm, y se acumulará su valor en la variable suma. Una vez salgamos del bucle se visualizará el valor de la variable suma.

#### Diseño del algoritmo

TABLA DE VARIABLES:

**entero** : suma, núm



**2.18.** Realizar un algoritmo que calcule y visualice las potencias de 2 entre 0 y 10.

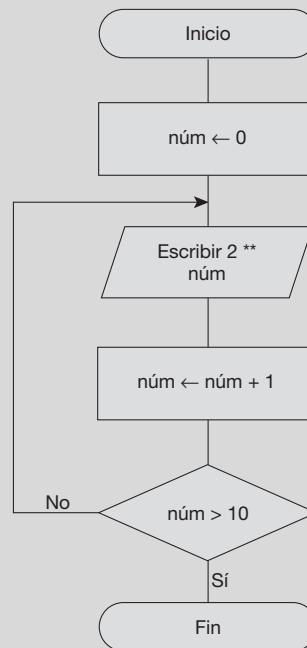
#### Análisis del problema

Hay que implementar un bucle que se ejecute once veces y dentro de él ir incrementando una variable que tome valores entre 0 y 10 y que se llamará núm. También dentro de él se visualizará el resultado de la operación  $2^{**} \text{núm}$ .

#### Diseño del algoritmo

TABLA DE VARIABLES:

**entero**: núm



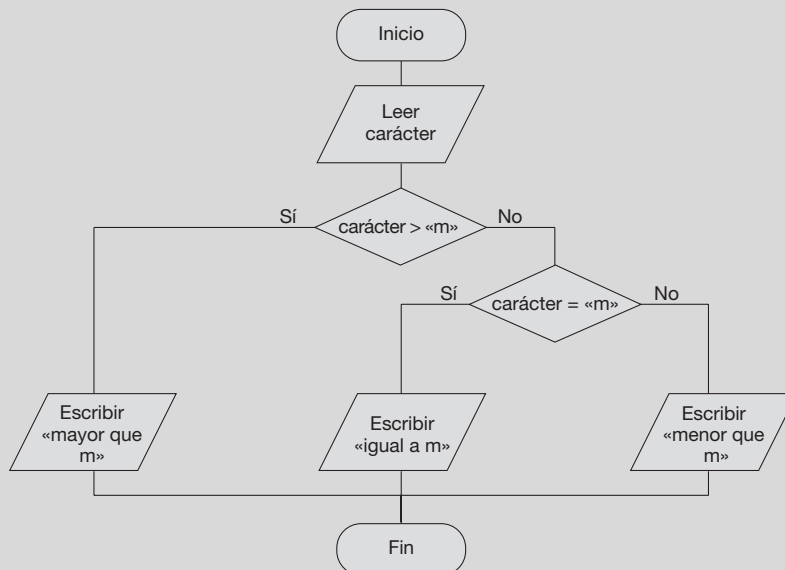
**2.19.** Leer un carácter y deducir si está situado antes o después de la «m» en orden alfabético.

#### Análisis del problema

Como dato de salida está el mensaje que nos dice la situación del carácter con respecto a la «m». Como entrada el propio carácter que introducimos por teclado. No se necesita ninguna variable auxiliar.

Se debe leer el carácter y compararlo con la «m» para ver si es mayor, menor o igual que ésta. Recuerde que para el ordenador también un carácter puede ser mayor o menor que otro, y lo hace comparando el código de los dos elementos de la comparación.

#### Diseño del algoritmo





**2.20. Leer dos caracteres y deducir si están en orden alfabético.****Análisis del problema**

DATOS DE SALIDA: El mensaje que nos indica si están en orden alfabético

DATOS DE ENTRADA: A y B (los caracteres que introducimos por teclado)

Se deben leer los dos caracteres y compararlos. Si A es mayor que B, no se habrán introducido en orden. En caso contrario estarán en orden o serán iguales.

**Diseño del algoritmo**

```
algoritmo ejercicio_2_20
var
    carácter: a, b
inicio
    leer (a,b)
    si a < b entonces
        escribir('están en orden')
    si_no
        si a = b entonces
            escribir ('son iguales')
        si_no
            escribir ('no están en orden')
        fin_si
    fin_si
fin
```

**2.21. Leer un carácter y deducir si está o no comprendido entre las letras I y M ambas inclusive.****Análisis del problema**

DATOS DE SALIDA: El mensaje

DATOS DE ENTRADA: El carácter

Se lee el carácter y se comprueba si está comprendido entre ambas letras con los operadores  $\geq$  y  $\leq$ .

**Diseño del algoritmo**

```
algoritmo Ejercicio_2_21
var
    carácter: c
inicio
    leer (c)
    si (c >= 'I') y (carácter <= 'M') entonces
        escribir ('Está entre la I y la M')
    si_no
        escribir ('no se encuentra en ese rango')
    fin_si
fin
```

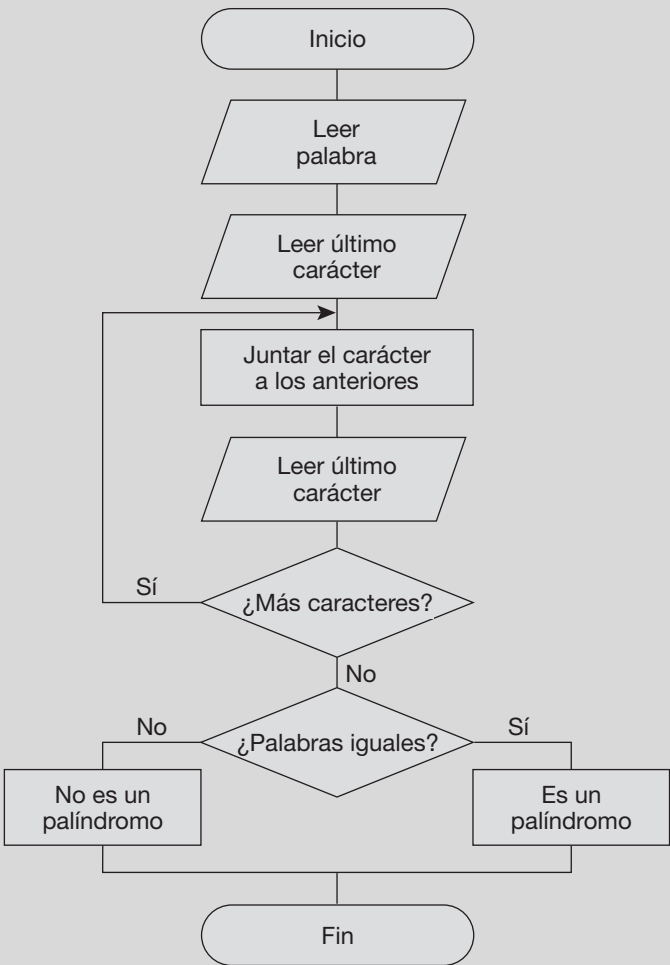
**2.22.** Averiguar si una palabra es un palíndromo. Un palíndromo es una palabra que se lee igual de izquierda a derecha que de derecha a izquierda, como por ejemplo «radar».

**Análisis del problema**

DATOS DE SALIDA: El mensaje que nos dice si es o no un palíndromo  
DATOS DE ENTRADA: Palabra  
DATOS AUXILIARES: Cada carácter de la palabra, palabra al revés

Para comprobar si una palabra es un palíndromo, se puede ir formando una palabra con los caracteres invertidos con respecto a la original y comprobar si la palabra al revés es igual a la original. Para obtener esa palabra al revés, se leerán en sentido inverso los caracteres de la palabra inicial y se irán juntando sucesivamente hasta llegar al primer carácter.

**Diseño del algoritmo**



- 2.23.** *Escribir un algoritmo para determinar el máximo común divisor de dos números enteros por el algoritmo de Euclides.*

### **Análisis del problema**

DATOS DE SALIDA:      Máximo común divisor (mcd)  
 DATOS DE ENTRADA:    Dos números enteros (a y b)  
 DATOS AUXILIARES:    Resto

Para hallar el máximo común divisor de dos números se debe dividir uno entre otro. Si la división es exacta, es decir si el resto es 0, el máximo común divisor es el divisor. Si no, se deben dividir otra vez los números, pero en este caso el dividendo será el antiguo divisor y el divisor el resto de la división anterior. El proceso se repetirá hasta que la división sea exacta.

Para diseñar el algoritmo se debe crear un bucle que se repita mientras que la división no sea exacta. Dentro del bucle se asignarán nuevos valores al dividendo y al divisor.

### **Diseño del algoritmo**

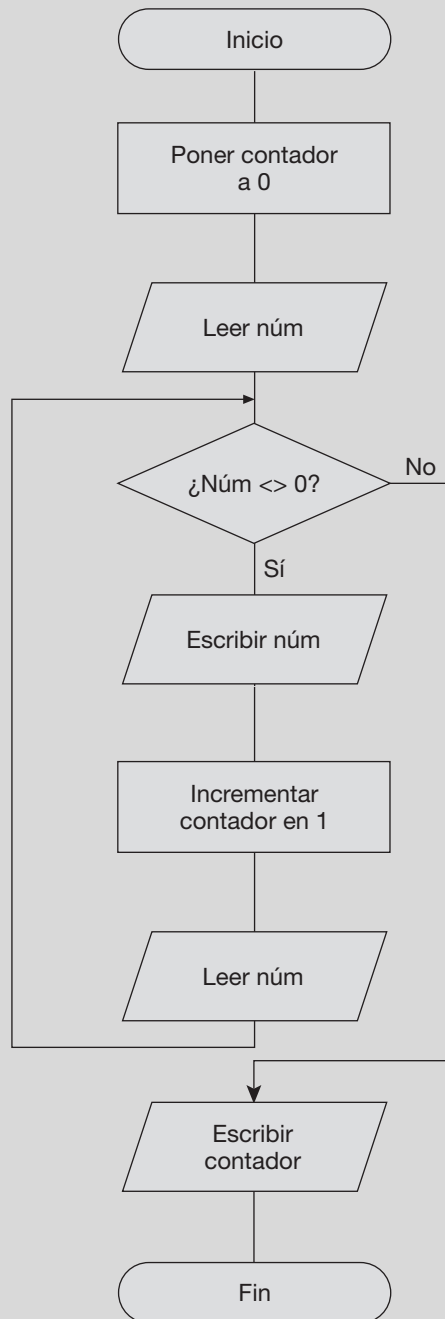
```
algoritmo Ejercicio_2_23
var
  entero: a,b,resto
inicio
  leer (a,b)
  mientras a mod b <> 0 hacer
    resto ← a mod b
    a ← b
    b ← resto
    mcd ← b
  fin_mientras
  escribir (mcd)
fin
```

- 2.24.** *Diseñar un algoritmo que lea e imprima una serie de números distintos de cero. El algoritmo debe terminar con un valor cero que no se debe imprimir. Finalmente se desea obtener la cantidad de valores leídos distintos de 0.*

### **Análisis del problema**

DATOS DE ENTRADA:    Los distintos números (núm)  
 DATOS DE SALIDA:    Los mismos números menos el 0, la cantidad de números (contador)

Se deben leer números dentro de un bucle que terminará cuando el último número leído sea cero. Cada vez que se ejecute dicho bucle y antes que se lea el siguiente número se imprime éste y se incrementa el contador en una unidad. Una vez se haya salido del bucle se debe escribir la cantidad de números leídos, es decir, el contador.

**Diseño del algoritmo**

**2.25.** Diseñar un algoritmo que imprima y sume la serie de números 3,6,9,12,...,99.

**Análisis del problema**

Se trata de idear un método con el que obtengamos dicha serie, que no es más que incrementar una variable de tres en tres. Para ello se hará un bucle que se acabe cuando el número sea mayor que 99 (o cuando se rea-

lice 33 veces). Dentro de ese bucle se incrementa la variable, se imprime y se acumula su valor en otra variable llamada suma, que será el dato de salida.

No tendremos por tanto ninguna variable de entrada, y sí dos de salida, la que nos va sacando los números de tres en tres (núm) y suma.

### Diseño del algoritmo

```

algoritmo Ejercicio_2_25
var
    entero: núm, suma
inicio
    suma  $\leftarrow$  0
    núm  $\leftarrow$  3
    mientras núm  $\leq$  99 hacer
        escribir (núm)
        suma  $\leftarrow$  suma + núm
        núm  $\leftarrow$  núm + 3
    fin_mientras
    escribir (suma)
fin
  
```

**2.26.** Escribir un algoritmo que lea cuatro números y, a continuación, escriba el mayor de los cuatro.

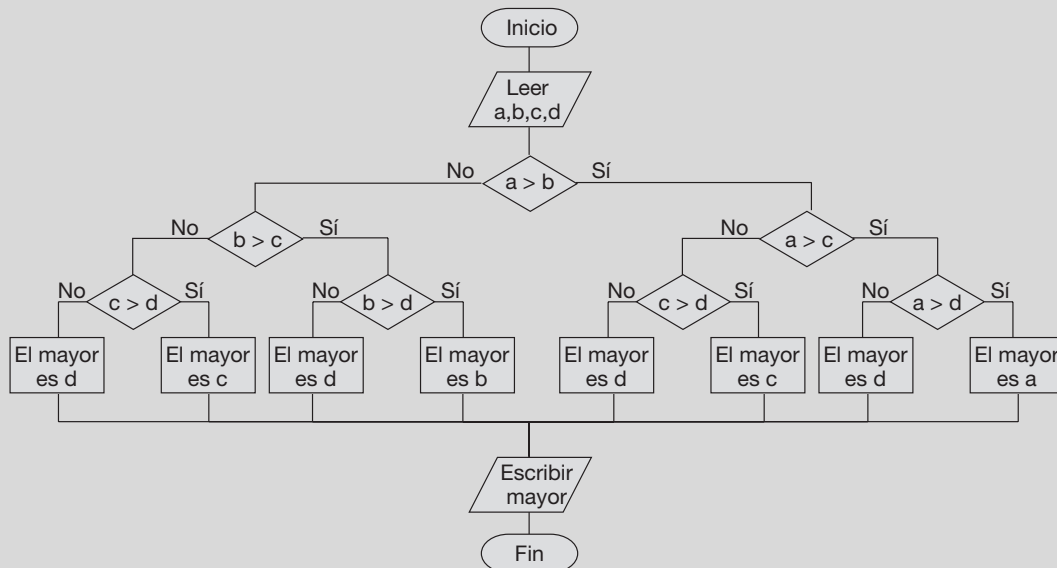
### Análisis del problema

DATOS DE SALIDA: Mayor (el mayor de los cuatro números)

DATOS DE ENTRADA: A, b, c, d (los números que leemos por teclado)

Hay que comparar los cuatro números, pero no hay necesidad de compararlos todos con todos. Por ejemplo, si a es mayor que b y a es mayor que c, es evidente que ni b ni c son los mayores, por lo que si a es mayor que d el número mayor será a y en caso contrario lo será d.

### Diseño del algoritmo



- 2.27.** Diseñar un algoritmo para calcular la velocidad (en metros/segundo) de los corredores de una carrera de 1.500 metros. La entrada serán parejas de números (minutos, segundos) que darán el tiempo de cada corredor. Por cada corredor se imprimirá el tiempo en minutos y segundos, así como la velocidad media. El bucle se ejecutará hasta que demos una entrada de 0,0 que será la marca de fin de entrada de datos.

### Análisis del problema

DATOS DE SALIDA:  $v$  (velocidad media)  
 DATOS DE ENTRADA:  $Mm, ss$  (minutos y segundos)  
 DATOS AUXILIARES: Distancia (distancia recorrida, que en el ejemplo es de 1500 metros) y tiempo (los minutos y los segundos que ha tardado en recorrerla)

Se debe efectuar un bucle que se ejecute hasta que  $mm$  sea 0 y  $ss$  sea 0. Dentro del bucle se calcula el tiempo en segundos con la fórmula  $tiempo = ss + mm * 60$ . La velocidad se hallará con la fórmula  $velocidad = distancia/tiempo$ .

### Diseño del algoritmo

```
algoritmo Ejercicio_2_27
var
  real: distancia, mm, tiempo, ss, v
inicio
  distancia ← 1500
  leer (mm,ss)
  mientras (mm <> 0.0) o (ss <> 0.0) hacer
    tiempo ← ss + mm * 60
    v ← distancia / tiempo
    escribir (mm,ss,v)
    leer (mm,ss)
  fin_mientras
fin
```

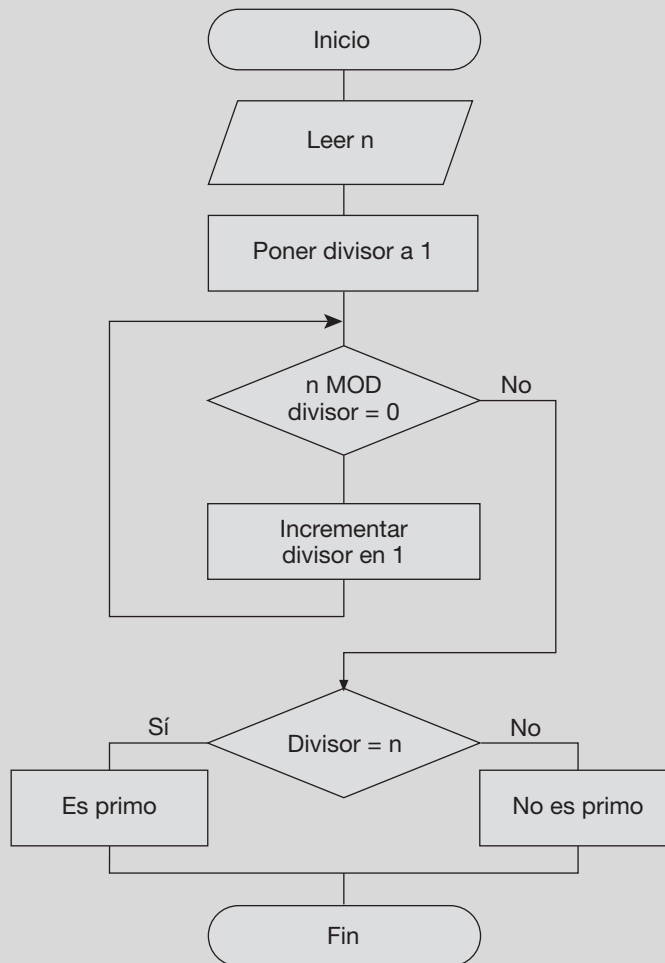
- 2.28.** Diseñar un algoritmo para determinar si un número  $n$  es primo (un número primo sólo es divisible por él mismo y por la unidad).

### Análisis del problema

DATOS DE SALIDA: El mensaje que nos indica si es o no primo  
 DATOS DE ENTRADA:  $n$   
 DATOS AUXILIARES: divisor (es el número por el que vamos a dividir  $n$  para averiguar si es primo)

Una forma de averiguar si un número es primo es por tanteo. Para ello se divide sucesivamente el número por los números comprendidos entre 2 y  $n$ . Si antes de llegar a  $n$  encuentra un divisor exacto, el número no será primo. Si el primer divisor es  $n$  el número será primo.

Por tanto se hará un bucle en el que una variable ( $divisor$ ) irá incrementándose en una unidad entre 2 y  $n$ . El bucle se ejecutará hasta que se encuentre un divisor, es decir hasta que  $n \bmod divisor = 0$ . Si al salir del bucle  $divisor = n$ , el número será primo.

**Diseño del algoritmo**

**2.29.** Escribir un algoritmo que calcule la superficie de un triángulo en función de la base y la altura.

**Análisis del problema**

DATOS DE SALIDA:  $s$  (superficie)

DATOS DE ENTRADA:  $b$  (base),  $a$  (altura)

Para calcular la superficie se aplica la fórmula  $s = \text{base} * \text{altura} / 2$ .

**Diseño del algoritmo**

```

algoritmo Ejercicio_2_29
var
  real:  $s$ ,  $a$ ,  $b$ 
inicio
  leer ( $b, a$ )
   $s \leftarrow b * a / 2$ 
  escribir ( $s$ )
fin
  
```

