

2.6. ESCRITURA DE ALGORITMOS

Como ya se ha comentado anteriormente, el sistema para describir (“escribir”) un algoritmo consiste en realizar una descripción paso a paso con un lenguaje natural del citado algoritmo. Recordemos que un algoritmo es un método o conjunto de reglas para solucionar un problema. En cálculos elementales estas reglas tienen las siguientes propiedades:

- deben ir seguidas de alguna secuencia definida de pasos hasta que se obtenga un resultado coherente,
- sólo puede ejecutarse una operación a la vez.

El flujo de control usual de un algoritmo es secuencial; consideremos el algoritmo que responde a la pregunta:

¿Qué hacer para ver la película de Harry Potter?

La respuesta es muy sencilla y puede ser descrita en forma de algoritmo general de modo similar a:

```
ir al cine
comprar una entrada (billete o ticket)
ver la película
regresar a casa
```

El algoritmo consta de cuatro acciones básicas, cada una de las cuales debe ser ejecutada antes de realizar la siguiente. En términos de computadora, cada acción se codificará en una o varias sentencias que ejecutan una tarea particular.

El algoritmo descrito es muy sencillo; sin embargo, como ya se ha indicado en párrafos anteriores, el algoritmo general se descompondrá en pasos más simples en un procedimiento denominado *refinamiento sucesivo*, ya que cada acción puede descomponerse a su vez en otras acciones simples. Así, por ejemplo, un primer refinamiento del algoritmo ir al cine se puede describir de la forma siguiente:

```
1. inicio
2. ver la cartelera de cines en el periódico
3. si no proyectan "Harry Potter" entonces
   3.1. decidir otra actividad
   3.2. bifurcar al paso 7
   si_no
   3.3. ir al cine
   fin_si
4. si hay cola entonces
   4.1. ponerse en ella
   4.2. mientras haya personas delante hacer
     4.2.1. avanzar en la cola
   fin_mientras
   fin_si
5. si hay localidades entonces
   5.1. comprar una entrada
   5.2. pasar a la sala
   5.3. localizar la(s) butaca(s)
   5.4. mientras proyectan la película hacer
     5.4.1. ver la película
   fin_mientras
   5.5. abandonar el cine
   si_no
   5.6. refunfuñar
   fin_si
6. volver a casa
7. fin
```

En el algoritmo anterior existen diferentes aspectos a considerar. En primer lugar, ciertas palabras reservadas se han escrito deliberadamente en negrita (**mientras**, **si_no**; etc.). Estas palabras describen las estructuras de control fundamentales y procesos de toma de decisión en el algoritmo. Éstas incluyen los conceptos importantes de *selección* (expresadas por **si-entonces-si_no**, **if-then-else**) y de *repetición* (expresadas con **mientras-hacer** o a veces **repetir-hasta** e **iterar-fin_iterar**, en inglés, **while-do** y **repeat-until**) que se encuentran en casi todos los algoritmos, especialmente en los de proceso de datos. La capacidad de decisión permite seleccionar alternativas de acciones a seguir o bien la repetición una y otra vez de operaciones básicas.

```

si proyectan la película seleccionada ir al cine
  si_no ver la televisión, ir al fútbol o leer el periódico
mientras haya personas en la cola, ir avanzando repetidamente
  hasta llegar a la taquilla

```

Otro aspecto a considerar es el método elegido para describir los algoritmos: empleo de *indentación* (sangrado o justificación) en escritura de algoritmos. En la actualidad es tan importante la escritura de programa como su posterior lectura. Ello se facilita con la *indentación* de las acciones interiores a las estructuras fundamentales citadas: selectivas y repetitivas. A lo largo de todo el libro la indentación o sangrado de los algoritmos será norma constante.

Para terminar estas consideraciones iniciales sobre algoritmos, describiremos las acciones necesarias para refinar el algoritmo objeto de nuestro estudio; para ello analicemos la acción:

Localizar la(s) butaca(s) .

Si los números de los asientos están impresos en la entrada, la acción compuesta se resuelve con el siguiente algoritmo:

1. **inicio** //algoritmo para encontrar la butaca del espectador
2. caminar hasta llegar a la primera fila de butacas
3. **repetir**
 - compara número de fila con número impreso en billete
 - si** son iguales **entonces** pasar a la siguiente fila **fin_si**
 - hasta_que** se localice la fila correcta
4. **mientras** número de butaca no coincide con número de billete
 - hacer** avanzar a través de la fila a la siguiente butaca
 - fin_mientras**
5. sentarse en la butaca
6. **fin**

En este algoritmo la repetición se ha mostrado de dos modos, utilizando ambas notaciones, **repetir... hasta_que** y **mientras... fin_mientras**. Se ha considerado también, como ocurre normalmente, que el número del asiento y fila coincide con el número y fila rotulado en el billete.

2.7. REPRESENTACIÓN GRÁFICA DE LOS ALGORITMOS

Para representar un algoritmo se debe utilizar algún método que permita independizar dicho algoritmo del lenguaje de programación elegido. Ello permitirá que un algoritmo pueda ser codificado indistintamente en cualquier lenguaje. Para conseguir este objetivo se precisa que el algoritmo sea representado gráfica o numéricamente, de modo que las sucesivas acciones no dependan de la sintaxis de ningún lenguaje de programación, sino que la descripción pueda servir fácilmente para su transformación en un programa, es decir, *su codificación*.

Los métodos usuales para representar un algoritmo son:

1. *diagrama de flujo*,
2. *diagrama N-S* (Nassi-Schneiderman),
3. *lenguaje de especificación de algoritmos: pseudocódigo*,
4. *lenguaje español, inglés...*
5. *fórmulas*.

Los métodos 4 y 5 no suelen ser fáciles de transformar en programas. Una descripción en *español narrativo* no es satisfactoria, ya que es demasiado prolífica y generalmente ambigua. Una *fórmula*, sin embargo, es un buen sistema de representación. Por ejemplo, las fórmulas para la solución de una ecuación cuadrática (de segundo grado) son un medio sucinto de expresar el procedimiento algorítmico que se debe ejecutar para obtener las raíces de dicha ecuación.

$$x_1 = \frac{(-b + \sqrt{b^2 - 4ac})}{2a} \quad x_2 = \frac{(-b - \sqrt{b^2 - 4ac})}{2a}$$

y significa lo siguiente:

1. Elevar al cuadrado b .
2. Tomar a ; multiplicar por c ; multiplicar por 4.
3. Restar el resultado obtenido de 2 del resultado de 1, etc.

Sin embargo, no es frecuente que un algoritmo pueda ser expresado por medio de una simple fórmula.

2.7.1. Pseudocódigo

El pseudocódigo es *un lenguaje de especificación (descripción) de algoritmos*. El uso de tal lenguaje hace el paso de codificación final (esto es, la traducción a un lenguaje de programación) relativamente fácil. Los lenguajes APL, Pascal y Ada se utilizan a veces como lenguajes de especificación de algoritmos.

El pseudocódigo nació como un lenguaje similar al inglés y era un medio de representar básicamente las estructuras de control de programación estructurada que se verán en capítulos posteriores. Se considera un *primer borrador*, dado que el pseudocódigo tiene que traducirse posteriormente a un lenguaje de programación. El pseudocódigo no puede ser ejecutado por una computadora. La *ventaja del pseudocódigo* es que en su uso, en la planificación de un programa, el programador se puede concentrar en la lógica y en las estructuras de control y no preocuparse de las reglas de un lenguaje específico. Es también fácil modificar el pseudocódigo si se descubren errores o anomalías en la lógica del programa, mientras que en muchas ocasiones suele ser difícil el cambio en la lógica, una vez que está codificado en un lenguaje de programación. Otra ventaja del pseudocódigo es que puede ser traducido fácilmente a lenguajes estructurados como Pascal, C, FORTRAN 77/90, C++, Java, C#, etc.

El pseudocódigo original utiliza para representar las acciones sucesivas palabras reservadas en inglés —similares a sus homónimas en los lenguajes de programación—, tales como **start**, **end**, **stop**, **if-then-else**, **while-end**, **repeat-until**, etc. La escritura de pseudocódigo exige normalmente la *indentación* (sangría en el margen izquierdo) de diferentes líneas.

Una representación en pseudocódigo —en inglés— de un problema de cálculo del salario neto de un trabajador es la siguiente:

```
start
    //cálculo de impuesto y salarios
    read nombre, horas, precio
    salario ← horas * precio
    tasas ← 0,25 * salario
    salario_neto ← salario - tasas
    write nombre, salario, tasas, salario
end
```

El algoritmo comienza con la palabra **start** y finaliza con la palabra **end**, en inglés (en español, **inicio**, **fin**). Entre estas palabras, sólo se escribe una instrucción o acción por línea.

La línea precedida por // se denomina *comentario*. Es una información al lector del programa y no realiza ninguna instrucción ejecutable, sólo tiene efecto de documentación interna del programa. Algunos autores suelen utilizar corchetes o llaves.

No es recomendable el uso de apóstrofos o simples comillas como representan en algunos lenguajes primitivos los comentarios, ya que este carácter es representativo de apertura o cierre de cadenas de caracteres en lenguajes como Pascal o FORTRAN, y daría lugar a confusión.

Otro ejemplo aclaratorio en el uso del pseudocódigo podría ser un sencillo algoritmo del arranque matinal de un coche.

```

inicio
  //arranque matinal de un coche
  introducir la llave de contacto
  girar la llave de contacto
  pisar el acelerador
  oír el ruido del motor
  pisar de nuevo el acelerador
  esperar unos instantes a que se caliente el motor
fin

```

Por fortuna, aunque el pseudocódigo nació como un sustituto del lenguaje de programación y, por consiguiente, sus palabras reservadas se conservaron o fueron muy similares a las del idioma inglés, el uso del pseudocódigo se ha extendido en la comunidad hispana con términos en español como **inicio**, **fin**, **parada**, **leer**, **escribir**, **si-en-tonces-si_no**, **mientras**, **fin_mientras**, **repetir**, **hasta_que**, etc. Sin duda, el uso de la terminología del pseudocódigo en español ha facilitado y facilitará considerablemente el aprendizaje y uso diario de la programación. En esta obra, al igual que en otras nuestras, utilizaremos el pseudocódigo en español y daremos en su momento las estructuras equivalentes en inglés, al objeto de facilitar la traducción del pseudocódigo al lenguaje de programación seleccionado.

Así pues, en los pseudocódigos citados anteriormente deberían ser sustituidas las palabras **start**, **end**, **read**, **write**, por **inicio**, **fin**, **leer**, **escribir**, respectivamente.

inicio	start	leer	read
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
fin	end	escribir	write

2.7.2. Diagramas de flujo

Un **diagrama de flujo** (*flowchart*) es una de las técnicas de representación de algoritmos más antigua y a la vez más utilizada, aunque su empleo ha disminuido considerablemente, sobre todo, desde la aparición de lenguajes de programación estructurados. Un diagrama de flujo es un diagrama que utiliza los símbolos (cajas) estándar mostrados en la Tabla 2.1 y que tiene los pasos de algoritmo escritos en esas cajas unidas por flechas, denominadas *líneas de flujo*, que indican la secuencia en que se debe ejecutar.

La Figura 2.17 es un diagrama de flujo básico. Este diagrama representa la resolución de un programa que deduce el salario neto de un trabajador a partir de la lectura del nombre, horas trabajadas, precio de la hora, y sabiendo que los impuestos aplicados son el 25 por 100 sobre el salario bruto.

Los símbolos estándar normalizados por **ANSI** (abreviatura de *American National Standards Institute*) son muy variados. En la Figura 2.18 se representa una plantilla de dibujo típica donde se contemplan la mayoría de los símbolos utilizados en el diagrama; sin embargo, los símbolos más utilizados representan:

- **proceso**
- **decisión**
- **conectores**
- **fin**
- **entrada/salida**
- **dirección del flujo**

El diagrama de flujo de la Figura 2.17 resume sus características:

- existe una caja etiquetada “**inicio**”, que es de tipo elíptico,
- existe una caja etiquetada “**fin**” de igual forma que la anterior,
- si existen otras cajas, normalmente son rectangulares, tipo rombo o paralelogramo (el resto de las figuras se utilizan sólo en diagramas de flujo generales o de detalle y no siempre son imprescindibles).

Tabla 2.1. Símbolos de diagrama de flujo

Símbolos principales	Función
	Terminal (representa el comienzo, “inicio”, y el final, “fin” de un programa. Puede representar también una parada o interrupción programada que sea necesario realizar en un programa).
	Entrada/Salida (cualquier tipo de introducción de datos en la memoria desde los periféricos, “entrada”, o registro de la información procesada en un periférico, “salida”).
	Proceso (cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas, de transferencia, etc.).
	Decisión (indica operaciones lógicas o de comparación entre datos —normalmente dos— y en función del resultado de la misma determina cuál de los distintos caminos alternativos del programa se debe seguir; normalmente tiene dos salidas —respuestas SÍ o NO— pero puede tener tres o más, según los casos).
	Decisión múltiple (en función del resultado de la comparación se seguirá uno de los diferentes caminos de acuerdo con dicho resultado).
	Conejero (sirve para enlazar dos partes cualesquiera de un organigrama a través de un conejero en la salida y otro conejero en la entrada. Se refiere a la conexión en la misma página del diagrama).
	Indicador de dirección o línea de flujo (indica el sentido de ejecución de las operaciones).
	Línea conectora (sirve de unión entre dos símbolos).
	Conejero (conexión entre dos puntos del organigrama situado en páginas diferentes).
	Llamada a subrutina o a un proceso predeterminado (una subrutina es un módulo independientemente del programa principal, que recibe una entrada procedente de dicho programa, realiza una tarea determinada y regresa, al terminar, al programa principal).
	Pantalla (se utiliza en ocasiones en lugar del símbolo de E/S).
	Impresora (se utiliza en ocasiones en lugar del símbolo de E/S).
	Teclado (se utiliza en ocasiones en lugar del símbolo de E/S).
	Comentarios (se utiliza para añadir comentarios clasificadores a otros símbolos del diagrama de flujo. Se pueden dibujar a cualquier lado del símbolo).

Problema:

Calcular el salario bruto y el salario neto de un trabajador “por horas” conociendo el nombre, número de horas trabajadas, impuestos a pagar y salario neto.

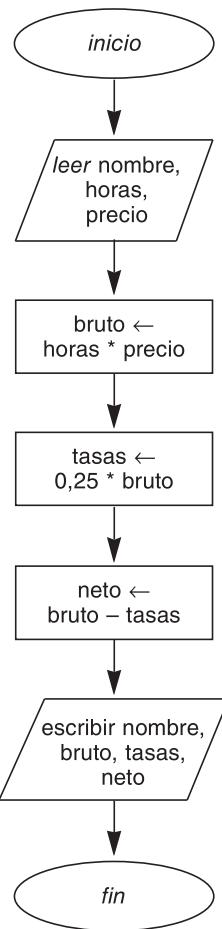


Figura 2.17. Diagrama de flujo.

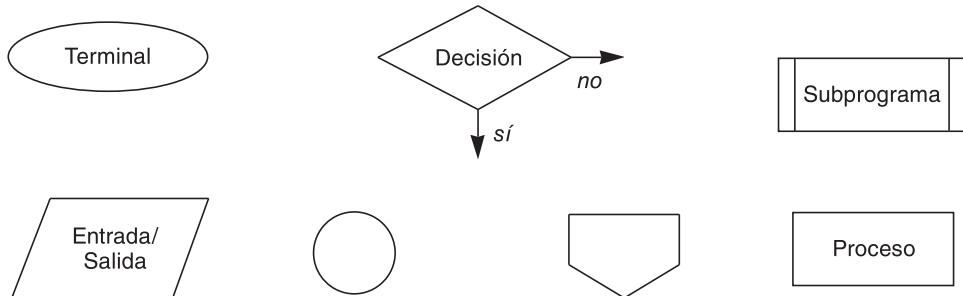


Figura 2.18. Plantilla típica para diagramas de flujo.

Se puede escribir más de un paso del algoritmo en una sola caja rectangular. El uso de flechas significa que la caja no necesita ser escrita debajo de su predecesora. Sin embargo, abusar demasiado de esta flexibilidad conduce a diagramas de flujo complicados e ininteligibles.

EJEMPLO 2.7

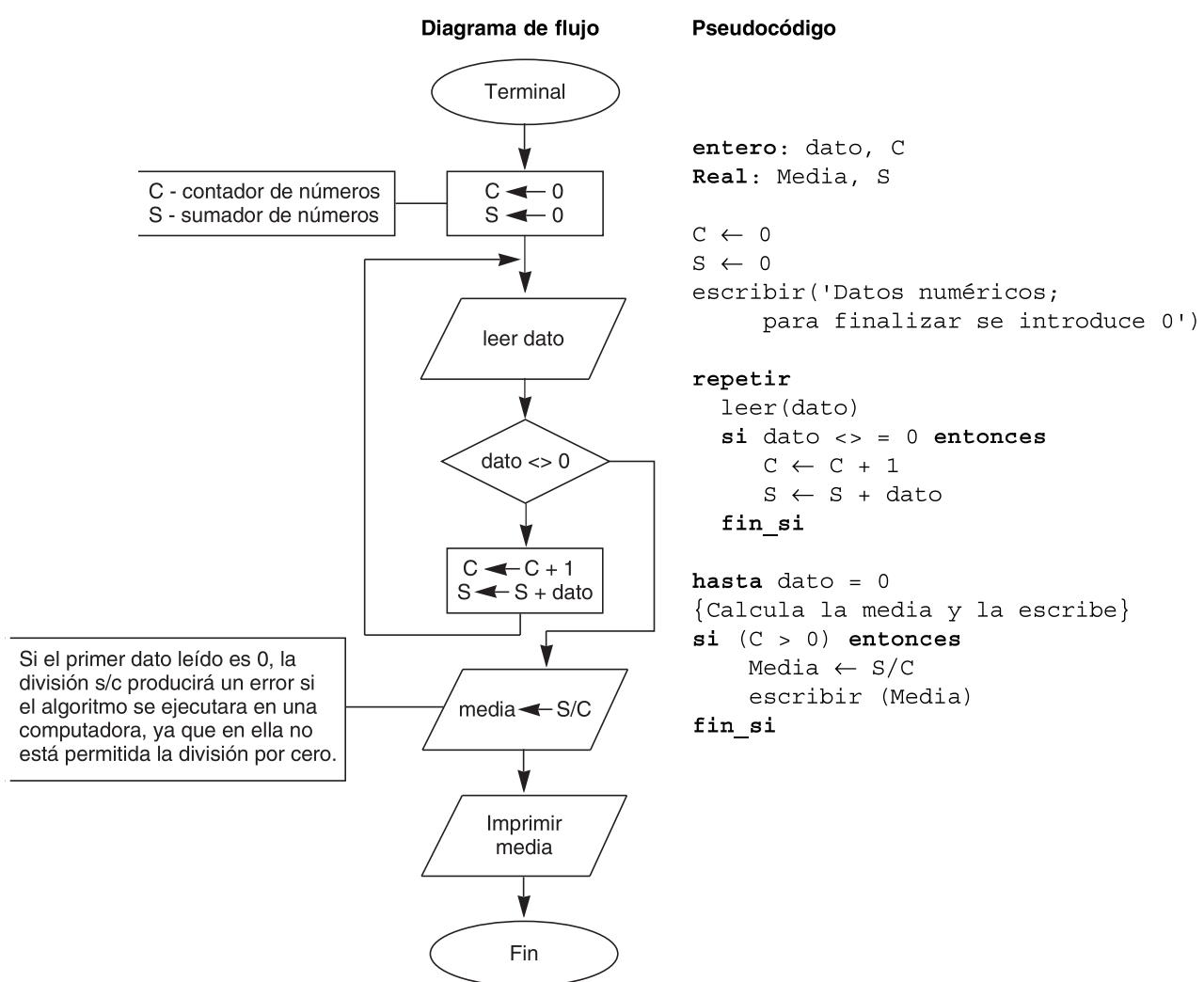
Calcular la media de una serie de números positivos, suponiendo que los datos se leen desde un terminal. Un valor de cero —como entrada— indicará que se ha alcanzado el final de la serie de números positivos.

El primer paso a dar en el desarrollo del algoritmo es descomponer el problema en una serie de pasos secuenciales. Para calcular una media se necesita sumar y contar los valores. Por consiguiente, nuestro algoritmo en forma descriptiva sería:

1. Inicializar contador de números C y variable suma S.
2. Leer un número.
3. Si el número leído es cero:
 - calcular la media;
 - imprimir la media;
 - fin del proceso.
 Si el número leído no es cero:
 - calcular la suma;
 - incrementar en uno el contador de números;
 - ir al paso 2.
4. Fin.

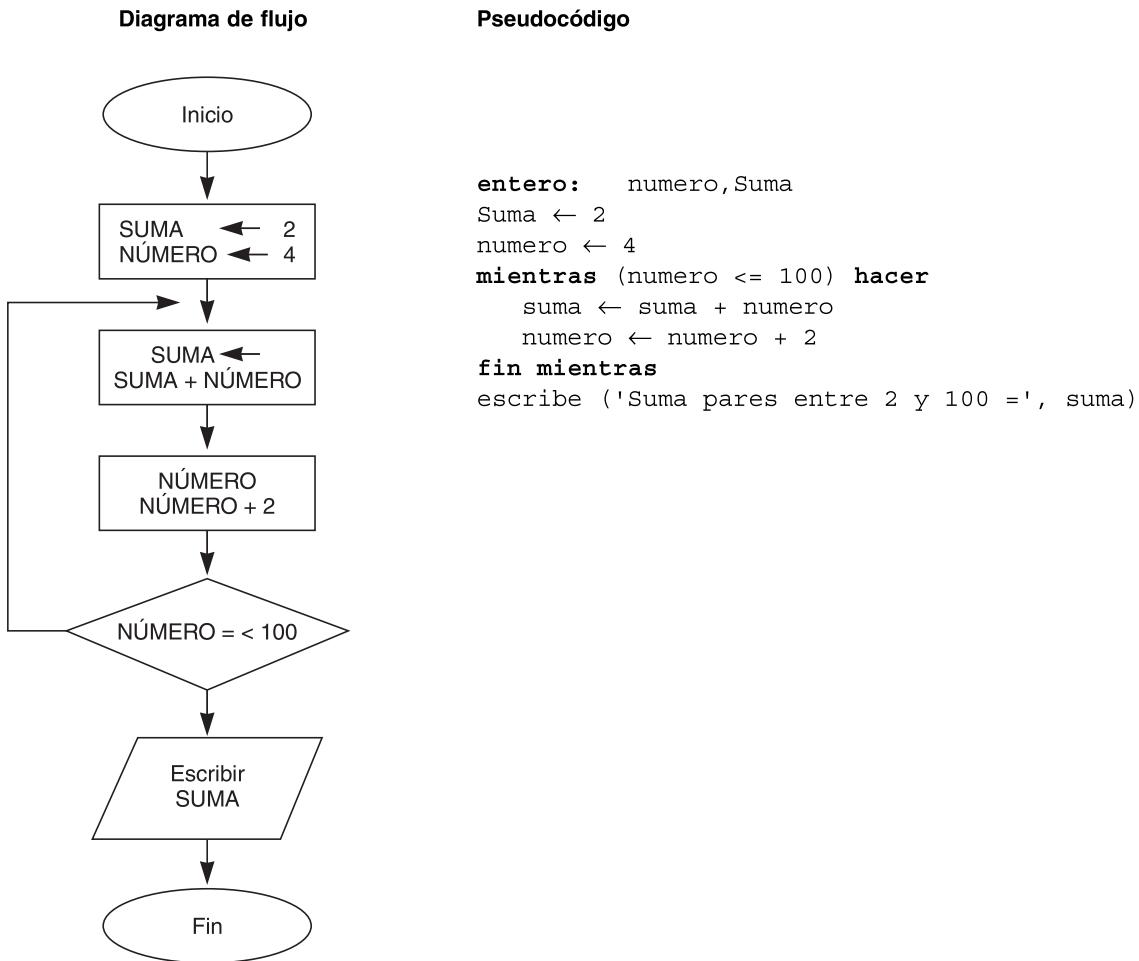
El refinamiento del algoritmo conduce a los pasos sucesivos necesarios para realizar las operaciones de lectura, verificación del último dato, suma y media de los datos.

Si el primer dato leído es 0, la división S/C produciría un error si el algoritmo se ejecutara en una computadora, ya que en ella no está permitida la división por cero.



EJEMPLO 2.8

Suma de los números pares comprendidos entre 2 y 100.

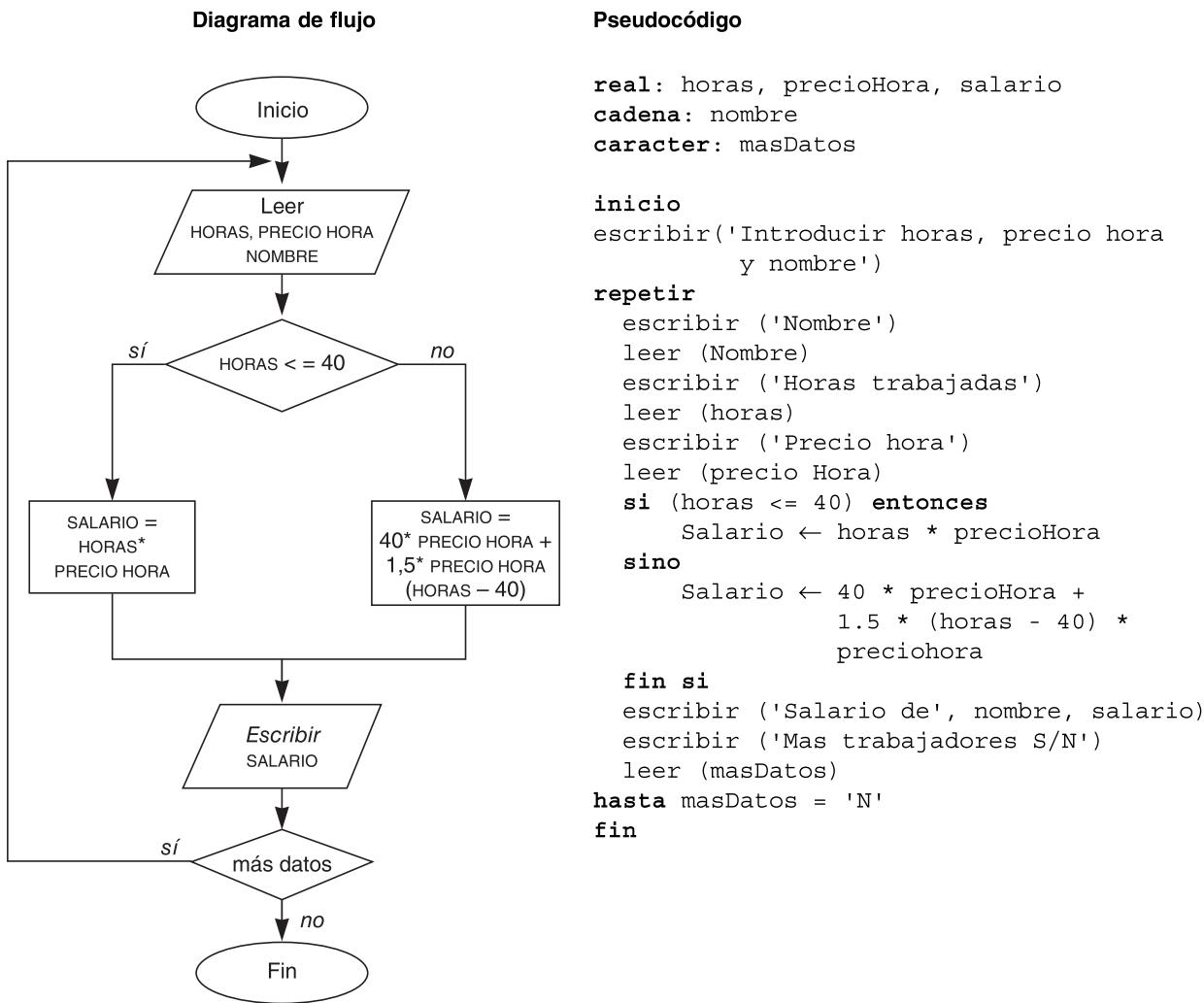
**EJEMPLO 2.9**

Se desea realizar el algoritmo que resuelva el siguiente problema: Cálculo de los salarios mensuales de los empleados de una empresa, sabiendo que éstos se calculan en base a las horas semanales trabajadas y de acuerdo a un precio especificado por horas. Si se pasan de cuarenta horas semanales, las horas extraordinarias se pagarán a razón de 1,5 veces la hora ordinaria.

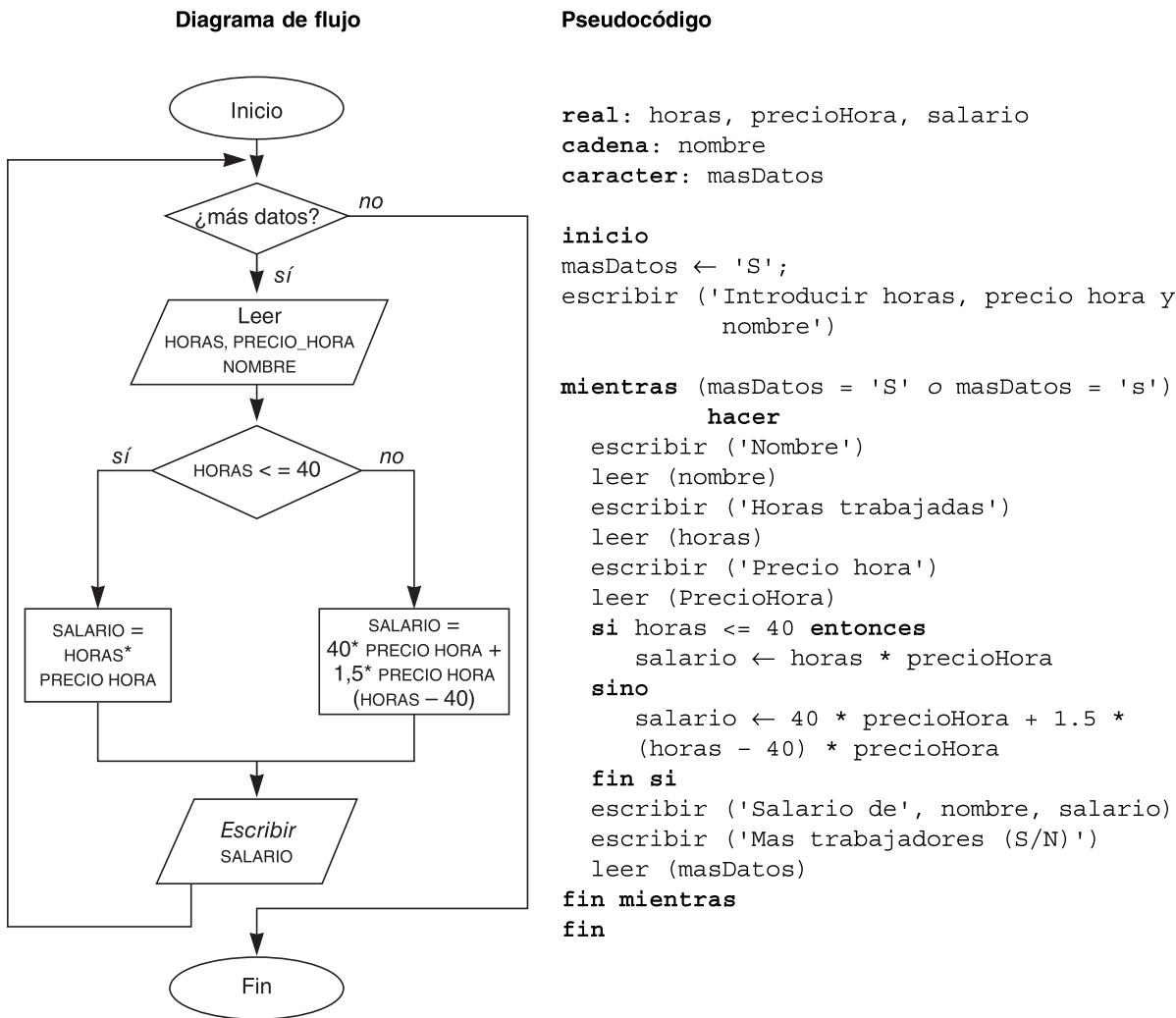
Los cálculos son:

1. Leer datos del archivo de la empresa, hasta que se encuentre la ficha final del archivo (HORAS, PRECIO_HORA, NOMBRE).
2. Si HORAS <= 40, entonces SALARIO es el producto de horas por PRECIO_HORA.
3. Si HORAS > 40, entonces SALARIO es la suma de 40 veces PRECIO_HORA más 1.5 veces PRECIO_HORA por (HORAS-40).

El diagrama de flujo completo del algoritmo y la codificación en pseudocódigo se indican a continuación:



Una variante también válida del diagrama de flujo anterior es:



EJEMPLO 2.10

La escritura de algoritmos para realizar operaciones sencillas de conteo es una de las primeras cosas que una computadora puede aprender.

Supongamos que se proporciona una secuencia de números, tales como

5 3 0 2 4 4 0 0 2 3 6 0 2

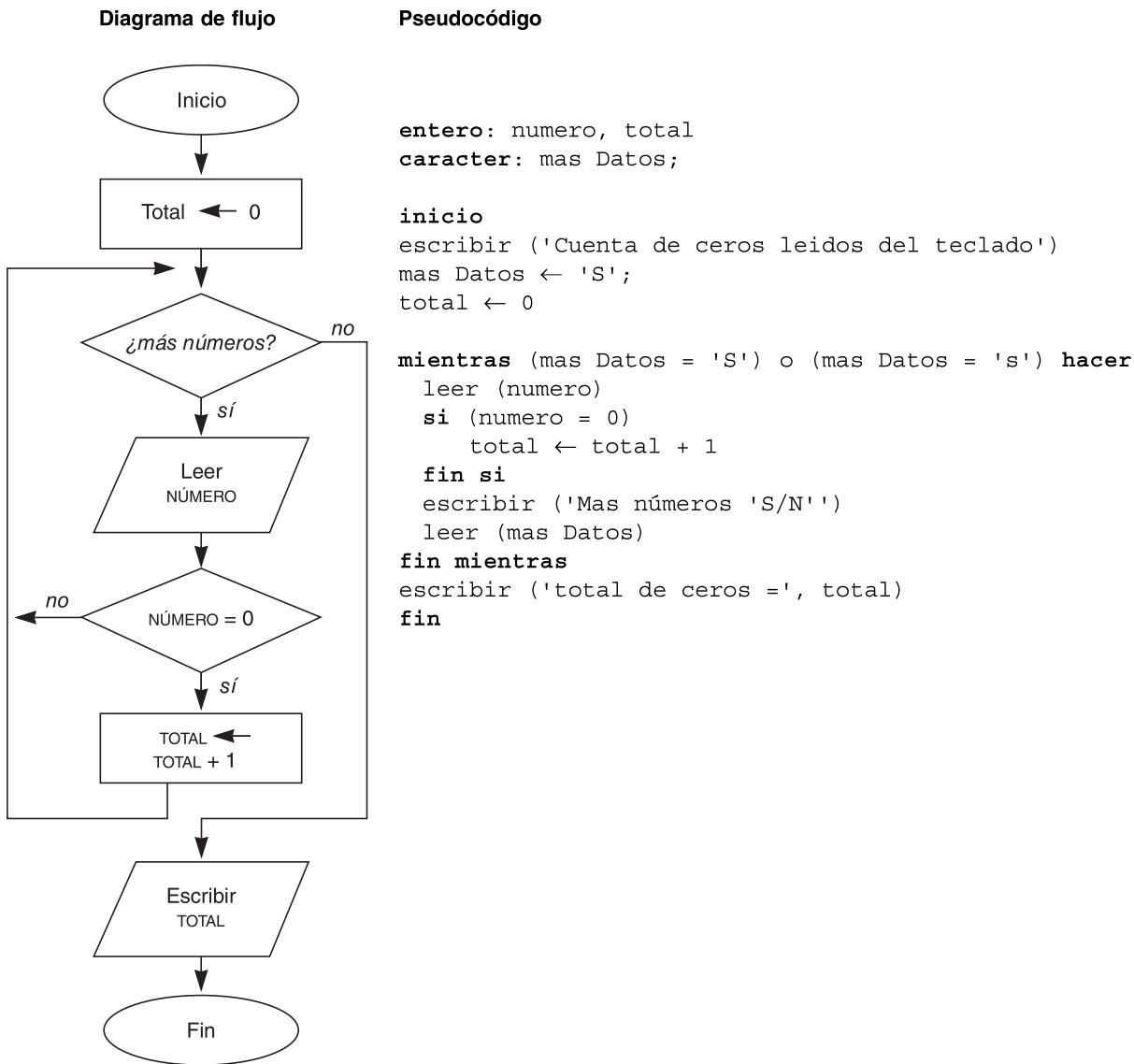
y desea contar e imprimir el número de ceros de la secuencia.

El algoritmo es muy sencillo, ya que sólo basta leer los números de izquierda a derecha, mientras se cuentan los ceros. Utiliza como variable la palabra NUMERO para los números que se examinan y TOTAL para el número de ceros encontrados. Los pasos a seguir son:

1. Establecer TOTAL a cero.
2. ¿Quedan más numeros a examinar?
3. Si no quedan numeros, imprimir el valor de TOTAL y fin.

4. Si existen mas numeros, ejecutar los pasos 5 a 8.
5. Leer el siguiente numero y dar su valor a la variable NUMERO.
6. Si NUMERO = 0, incrementar TOTAL en 1.
7. Si NUMERO <> 0, no modificar TOTAL.
8. Retornar al paso 2.

El diagrama de flujo y la codificación en pseudocódigo correspondiente es:



EJEMPLO 2.11

Dados tres números, determinar si la suma de cualquier pareja de ellos es igual al tercer número. Si se cumple esta condición, escribir "Iguales" y, en caso contrario, escribir "Distintas".

En el caso de que los números sean: 3 9 6

la respuesta es "Iguales", ya que $3 + 6 = 9$. Sin embargo, si los números fueran:

2 3 4

el resultado sería "Distintas".

Para resolver este problema, se puede comparar la suma de cada pareja con el tercer número. Con tres números solamente existen tres parejas distintas y el algoritmo de resolución del problema será fácil.

1. Leer los tres valores, A, B y C.
2. Si $A + B = C$ escribir "Iguales" y parar.
3. Si $A + C = B$ escribir "Iguales" y parar.
4. Si $B + C = A$ escribir "Iguales" y parar.
5. Escribir "Distintas" y parar.

El diagrama de flujo y la codificación en pseudocódigo correspondiente es la Figura 2.19.

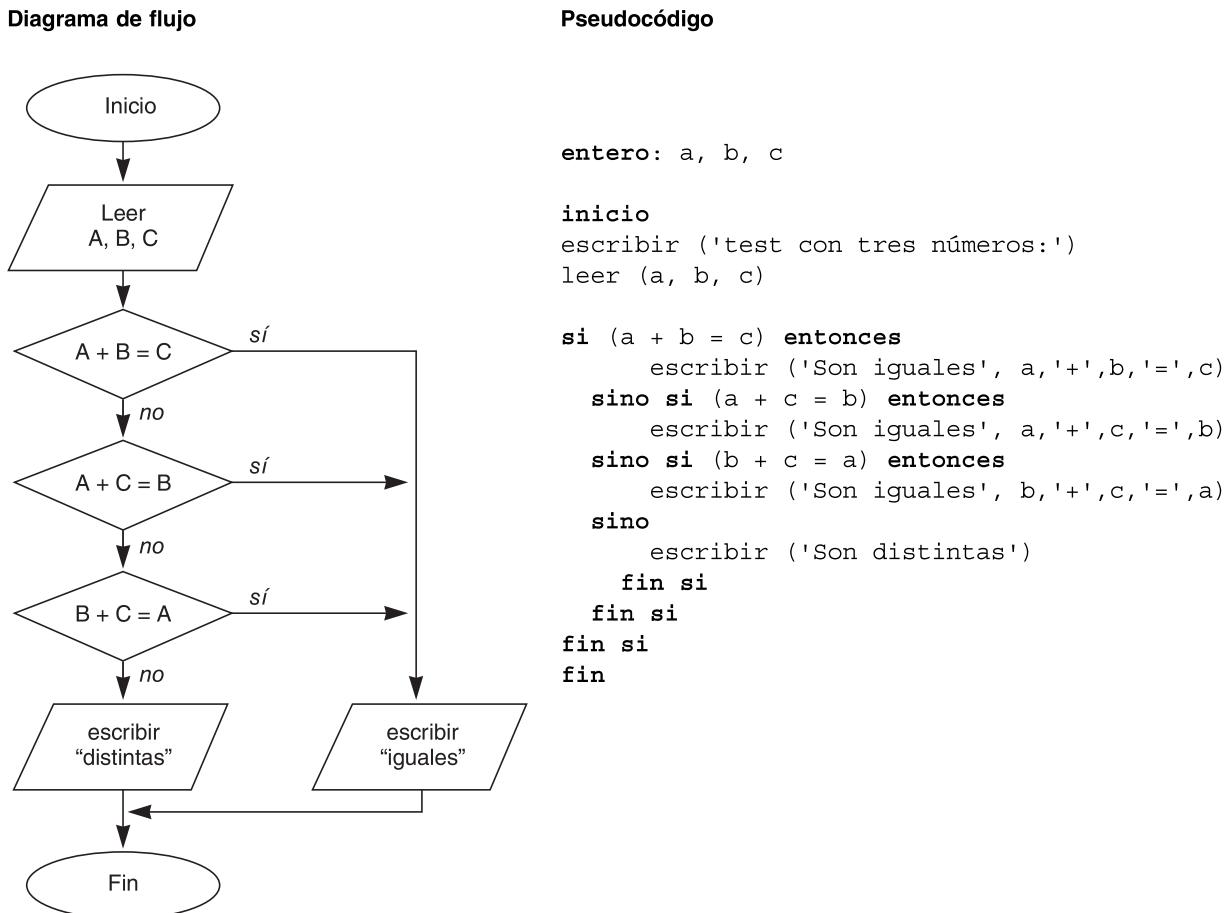


Figura 2.19. Diagrama de flujo y codificación en pseudocódigo (Ejemplo 2.11).

2.7.3. Diagramas de Nassi-Schneiderman (N-S)

El diagrama N-S de Nassi Schneiderman —también conocido como diagrama de Chapin— es como un diagrama de flujo en el que se omiten las flechas de unión y las cajas son contiguas. Las acciones sucesivas se escriben en cajas sucesivas y, como en los diagramas de flujo, se pueden escribir diferentes acciones en una caja.

Un algoritmo se representa con un rectángulo en el que cada banda es una acción a realizar.

EJEMPLO

Escribir un algoritmo que lea el nombre de un empleado, las horas trabajadas, el precio por hora y calcule los impuestos a pagar (tasa = 25%) y el salario neto.

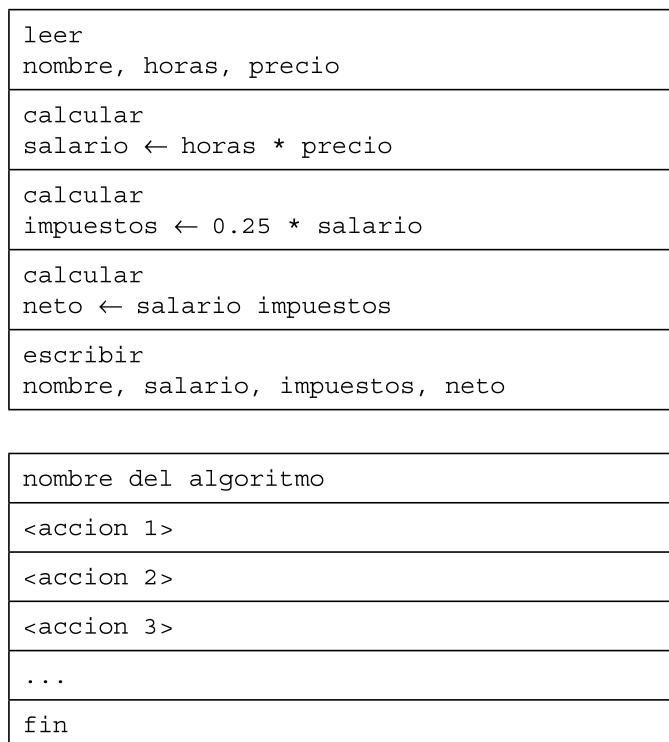


Figura 2.20. Representación gráfica N-S de un algoritmo.

Otro ejemplo es la representación de la estructura condicional (Figura 2.21).

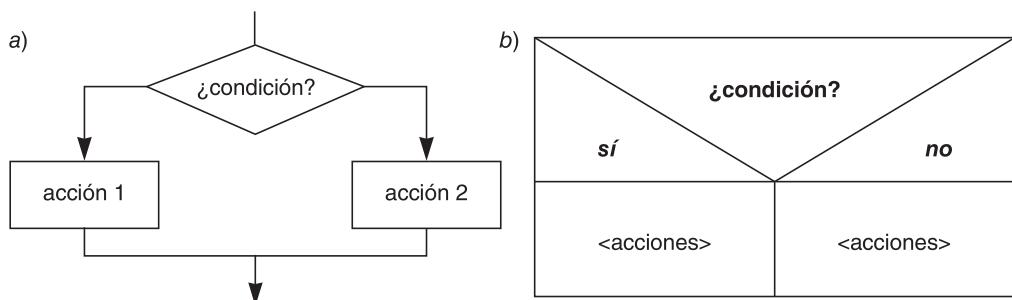


Figura 2.21. Estructura condicional o selectiva: a) diagrama de flujo; b) diagrama N-S.