4. ESTRUCTURAS DE PROGRAMACIÓN

El saber que en teoría
un problema se puede resolver
con una computadora
no basta para decirnos
si resulta práctico hacerlo o no.
Baase y Van Gerder

En programación estructurada se utilizan tres tipos de estructuras: secuenciales, aquellas que se ejecutan una después de otra siguiendo el orden en que se han escrito; de decisión, que permiten omitir parte del código o seleccionar el flujo de ejecución de entre dos o más alternativas; y las iterativas, que se utilizan para repetir la ejecución de cierta parte del programa.

4.1 ESTRUCTURAS SECUENCIALES

Estas estructuras se caracterizan por ejecutarse una después de otra en el orden en que aparecen en el algoritmo o el programa, como la asignación de un dato a una variable, la lectura o la impresión de un dato.

4.1.1 Asignación

Esta operación consiste en guardar un dato en una posición determinada de la memoria reservada mediante la declaración de una variable.

La asignación es una operación relevante en el paradigma de programación imperativo, donde todos los datos, los leídos y los obtenidos como resultado de un cálculo, se guardan en memoria en espera de posteriores instrucciones.

Ejemplo:

```
entero a, b, c
cadena: operación
a = 10
```

b = 15
c = a + b
operación = “suma”

La asignación de variables se trata con mayor detalle en la sección 2.2.4.

4.1.2 Entrada de datos

Un programa actúa sobre un conjunto de datos suministrados por el usuario o tomados desde un dispositivo de almacenamiento; de igual manera, el algoritmo requiere que se le proporcione los datos a partir de los cuales obtendrá los resultados esperados.

La lectura de datos consiste en tomar datos desde un medio externo o desde un archivo y llevarlos a la memoria donde pueden ser procesados por el programa (Joyanes, 1988). El dispositivo predeterminado es el teclado.

Para la lectura de datos se cuenta con la instrucción *leer* en pseudocódigo y sus equivalentes en las diferentes notaciones.

En pseudocódigo, la instrucción *leer* tiene la siguiente sintaxis

Leer <lista de identificadores de variables>

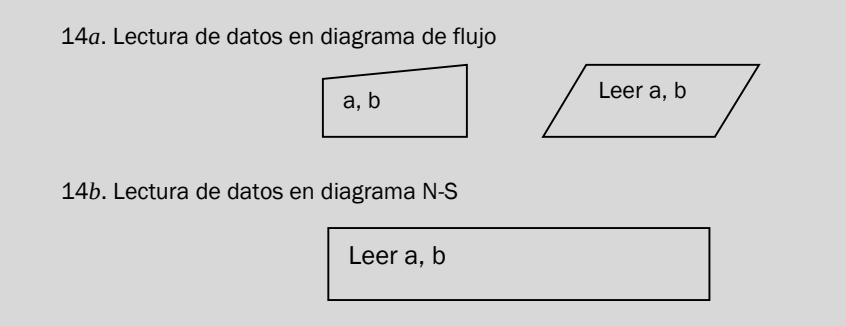
Ejemplo:

Leer a, b

Donde *a* y *b* son las variables que recibirán los valores y, por tanto, deben estar declaradas previamente.

En diagrama de flujo, la instrucción *leer* puede representarse de dos maneras: utilizando el símbolo de lectura por teclado o mediante un proceso de entrada y salida. En diagrama N-S, todas las estructuras secuenciales se escriben dentro de una caja, de manera que la lectura de datos se representa mediante la palabra *leer* dentro de una caja, como se muestran en la figura 14.

Figura 14. Símbolos de entrada de datos



Cualquiera de los dos símbolos de la figura 14*a* es válido para designar una lectura de datos en un diagrama de flujo. La figura 14b es exclusiva para diagramas N-S.

4.1.3 Salida de datos

Todo programa desarrolla una o más tareas y genera uno o más datos como resultado. Para presentar los resultados al usuario se hace uso de las instrucciones y los dispositivos de salida de datos, como la pantalla o la impresora.

Para enviar la información desde la memoria del computador hasta un dispositivo de salida se utiliza la instrucción definida en pseudocódigo como: *escribir* y sus equivalentes en las demás notaciones.

En pseudocódigo la lectura de datos se escribe de la forma:

Escribir <lista de constantes y variables>

Ejemplo:

Escribir a, b

Donde *a y b* son variables previamente definidas

Escribir "Este es un mensaje"

Cuando se escriben más de una variable es necesario separarlas con comas (,) y los mensajes se escriben entre comillas dobles " ". Si una variable es escrita entre comillas se mostrará el identificador y no el contenido.

Ejemplo:

Cadena: nombre
Entero: edad
Nombre = "Juan"

Edad = 25

La forma correcta de mostrar los datos es:

Escribir "Nombre: ", nombre
Escribir "Edad: ", edad

Para que la salida será:

Nombre: Juan
Edad: 25

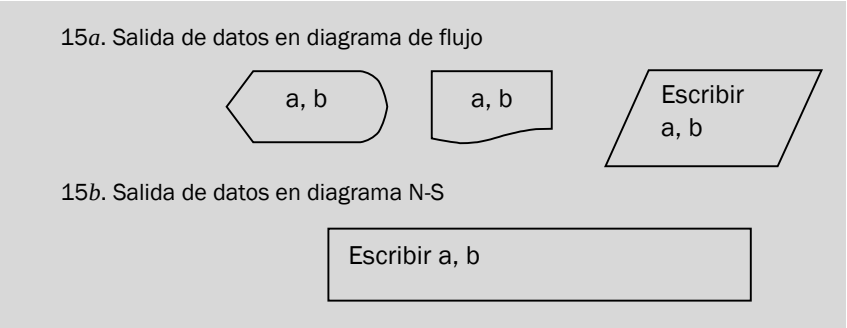
Escribir de la forma:

Escribir "nombre"
Escribir "edad"

Solo mostrará las etiquetas: "nombre" y "edad"

En diagrama de flujo la salida de datos puede representarse mediante tres símbolos: salida por pantalla, salida por impresora y proceso de entrada y salida (figura 15*a*), mientras que en diagrama N-S la instrucción *escribir* se coloca dentro de una caja, como se muestra en la figura 15*b*.

Figura 15. Símbolos de salida de datos



4.1.4 Ejemplos con estructuras secuenciales

Ejemplo 3. Sumar dos números

Este algoritmo lee dos números, los almacena en variables, luego los suma y muestra el resultado. Se trata de representar algorítmicamente el desarrollo de la operación sumar utilizando dos valores. Por ejemplo:

3 + 2 = 5

Si esta operación se la convierte en una expresión aritmética con variables se tiene:

$r = x + y$

Si se asigna o lee valores para las variables *x* e *y* se puede calcular el resultado. Ejemplo:

$x = 6$
$y = 7$

$r = x + y$
$r = 6 + 7$
$r = 13$

De este ejemplo se deduce que el algoritmo requiere que se le suministren dos datos, estos son los datos de entrada, que se almacenarán en las variables *x* e *y*. Con éstos se desarrolla la expresión (proceso) y el resultado es el dato que interesa al usuario, por tanto éste es el dato de salida.

Entrada: x, y
Proceso: r = x + y
Salida: r

Ahora que se comprende la solución general del problema, ésta se puede expresar de forma algorítmica. En el cuadro 14 se presenta el algoritmo en notación de pseudocódigo, en las figuras 16 y 17 los diagramas de flujo y N-S, respectivamente.

Cuadro 14. Pseudocódigo del algoritmo sumar dos números

```
1.  Inicio
2.      Entero: x, y, r
3.      Leer x, y
4.      r = x + y
5.  Fin algoritmo
```

Figura 16. Diagrama de flujo para sumar dos números

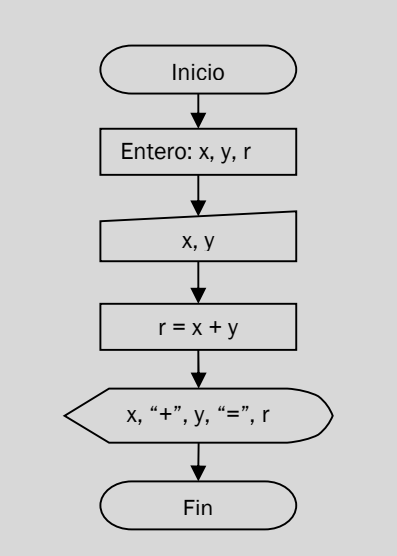


Figura 17. Diagrama N-S para sumar dos números

Inicio
Entero: x, y, r
Leer x, y
r = x + y
Escribir x, “ + “, y, “ = “, r
Fin algoritmo

En este algoritmo se declaran tres variables: las dos primeras (x, y) corresponden a los dos datos de entrada, la tercera (r) guarda el resultado del proceso. La salida para este ejercicio es, sin duda, el resultado de la operación; no obstante, en muchos programas es necesario mostrar también algunos de los datos entrados para que la salida sea más entendible, en este ejemplo se muestra los operandos y el resultado de la operación.

Para verificar si este algoritmo es correcto se crea una tabla como la que se presenta en el cuadro 15, se ejecuta el algoritmo línea por línea, proporcionando los datos de entrada y realizando las operaciones. Los datos se escriben en las columnas de la tabla etiquetadas con las variables y luego se verifica si la salida es correcta. (La verificación de algoritmos se explica en la sección 3.5).

Cuadro 15. Verificación del algoritmo para sumar dos números

Ejecución	X	y	r	Salida
1	3	4	7	3 + 4 = 7
2	5	8	13	5 + 8 = 13
3	2	3	5	2 + 3 = 5

Es importante tener presente que al verificar el algoritmo se debe ejecutar estrictamente los pasos de éste y no proceder de memoria, ya que muchas veces lo que se tiene en mente y lo que se escribe no es lo mismo y lo que se quiere verificar es lo que está escrito.

Ejemplo 4. El cuadrado de un número.

El cuadrado de un número es igual a multiplicar el número por sí mismo, de la forma:

$3^2 = 3 * 3 = 9$
$5^2 = 5 * 5 = 25$

Ahora, para generalizar la operación se utiliza una variable para el número que se desea elevar al cuadrado y se tiene una expresión de la forma:

$num^2 = num * num = ?$

Si llevamos este proceso a un algoritmo y luego a un programa, el computador podrá proporcionar el resultado para cualquier número. Ahora bien, antes de proceder a diseñar el algoritmo se organiza la información que se tiene en forma de: entrada, salida y proceso.

Entrada: número
Salida: el cuadrado del número
*Proceso: cuadrado = número * número*

Con esta información se procede a diseñar la solución del problema. En el cuadro 16 se presenta el pseudocódigo, y en las figuras 18 y 19 los diagramas de flujo y N-S.

Cuadro 16. Pseudocódigo para calcular el cuadrado de un número

```
1.  Inicio
2.      Entero: num, cuad
3.      Leer num
4.      cuad = num * num
5.      Escribir cuad
6.  Fin algoritmo
```

En este algoritmo se declaran dos variables: *num* para almacenar el número ingresado por el usuario y *cuad* para almacenar el resultado del proceso, es decir, el cuadrado del número. En el cuadro 17 se presenta tres ejecuciones de prueba para verificar la corrección del algoritmo: en la primera se ingresa el número 3 y se obtiene como resultado el 9, en la segunda se ingresa el número 2 y se obtiene el cuatro y en la tercera se ingresa el 7 y el resultado es 49, estos resultados confirman que el algoritmo es correcto.

Cuadro 17. Verificación del algoritmo el cuadrado de un número

Ejecución	num	cuad	Salida
1	3	9	9
2	2	4	4
3	7	49	49

Figura 18. Diagrama de flujo para calcular el cuadrado de un número

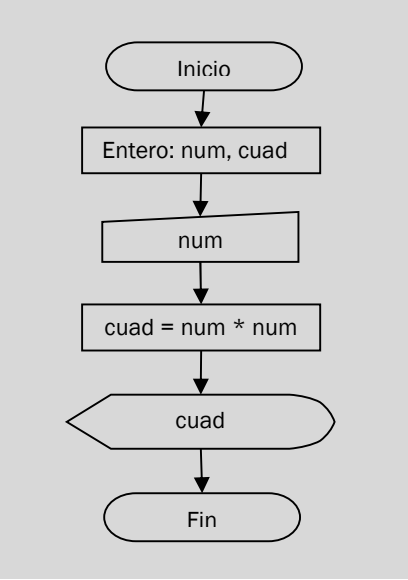


Figura 19. Diagrama N-S para calcular el cuadrado de un número

Inicio
Entero: num, cuad
Leer num
cuad = num * num
Escribir cuad
Fin algoritmo

Ejemplo 5. Precio de venta de un producto

Considérese el siguiente caso: una tienda de licores adquiere sus productos por cajas y los vende por unidades. Dado que una caja de cualquier producto tiene un costo *c* y contiene *n* unidades, se desea calcular el precio *p* para cada unidad, de manera que se obtenga el 30% de utilidad.

Lo primero que hay que hacer es comprender bien el dominio del problema, esto es, poder obtener los resultados y solucionar el problema de forma manual.

Ejemplo 1: se compra una caja de ron en $ 240.000.oo (*c*), ésta contiene 24 unidades (*n*) y se desea obtener el 30% de utilidad. ¿A qué precio (*p*) se debe vender cada unidad?

Para solucionar este problema es necesario realizar tres cálculos, primero, aplicar el porcentaje de utilidad sobre el costo de la caja; segundo, sumar el costo más la utilidad para obtener el precio total de la caja; y tercero, dividir el valor total sobre el número de unidades (*n*) para obtener el precio unitario (*p*).

Primero:

*utilidad = 240.000.oo * 30/100*
utilidad = $ 72.000.oo

Segundo:

total = $ 240.000.oo + $ 72.000.oo
total = $ 312.000.oo

Tercero:

p = $ 312.000.oo / 24
p = $ 13.000.oo

Finalmente tenemos el precio de venta de cada unidad: $ 13.000.oo

Ejemplo 2: supóngase que se compra otro producto, cuyo costo por caja es de $ 600.000.oo y contiene 12 unidades. Entonces se tiene que:
Primero:

*utilidad = $ 600.000.oo * 30/100*
utilidad = $ 180.000.oo

Segundo:

total = $ 600.000.oo + $ 180.000.oo
total = $ 780.000.oo

Tercero:

p = $ 780.000.oo / 12
p = $ 65.000.oo

El precio de venta de cada unidad es de $ 65.000.oo

Con estos dos ejemplos se logra comprender cómo se soluciona el problema, ahora es necesario generalizar el proceso de solución de manera que se pueda especificar un algoritmo; para ello hay que representar los valores mediante variables y los cálculos con fórmulas que se aplican sobre dichas variables, así:

Sea:

c = costo de la caja
n = número de unidades
p = precio de venta por unidad

De estos datos: los dos primeros son entradas, el último es la salida. Los cálculos a realizar son:

*utilidad = c * 30/100*
total = c + utilidad
p = total / n

Habiendo llegado a este punto ya se puede proceder a diseñar un algoritmo para solucionar cualquier caso de este problema. En el cuadro 18 se muestra el pseudocódigo, en la figura 20 el diagrama de flujo y en la 21 el diagrama N-S.

Cuadro 18. Pseudocódigo para calcular el precio unitario de un producto

```
1.  Inicio
2.      Entero: n
        Real: c, utilidad, total, p
3.      Leer c, n
4.      utilidad = c * 30/100
5.      total = c + utilidad
6.      p = total / n
7.      Escribir “Precio = ”, p
8.  Fin algoritmo
```

Figura 20. Diagrama de flujo para calcular el precio unitario de un producto

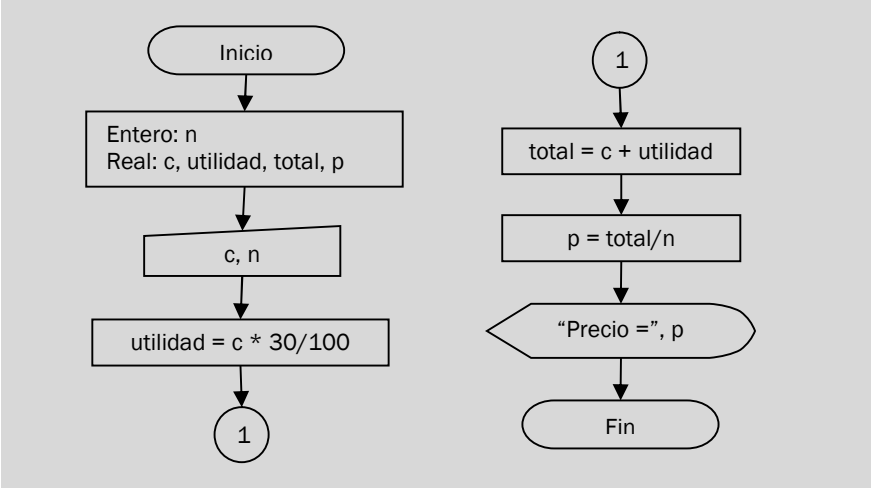


Figura 21. Diagrama N-S para calcular el precio unitario de un producto

Inicio
Entero: n Real: c, utilidad, total, p
Leer c, n
utilidad = c * 30/100
total = c + utilidad
p = total / n
Escribir "Precio = ", p
Fin algoritmo

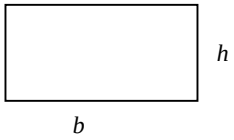
En el cuadro 19 se presentan los datos de tres pruebas efectuadas para verificar la corrección de este algoritmo, las dos primeras columnas corresponden a los datos de entrada y la última a la salida que genera el algoritmo.

Cuadro 19. Verificación del algoritmo para calcular el precio unitario de un producto

C	n	Utilidad	total	P	Salida
240.000	24	72.000	312.000	13.000	Precio = 13.000
600.000	12	180.000	780.000	65.000	Precio = 65.000
300.000	10	90.000	390.000	39.000	Precio = 39.000

Ejemplo 6. Área y perímetro de un rectángulo

Se requiere un algoritmo para calcular el área y el perímetro de un rectángulo de cualquier dimensión.



Para solucionar este problema es necesario conocer las fórmulas para obtener tanto el área como el perímetro de un rectángulo.

Siendo:

b = base
h = altura

Las fórmulas a utilizar son:

*Área = b * h*
*Perímetro = 2 * (b + h)*

Si el rectángulo tiene una base de 10 cm y una altura de 5 cm, se obtiene:

*Área = 10 * 5*
Área = 50

*Perímetro = 2 * (10 + 5)*
Perímetro = 30

Cómo se aprecia en el ejemplo, para hacer los cálculos es necesario contar con la base y la altura, de manera que éstos corresponden a los datos que el usuario debe suministrar y que el algoritmo ha de leer, mientras que el área y el perímetro son los datos que el algoritmo debe calcular y presentar al usuario; por lo tanto:

Datos de entrada: base (b) y altura (h)
Datos de salidas: área (a) y perímetro (p)
Procesos:
*a = b * h*
*p = 2 * (b + h)*

El diseño de la solución en notación de pseudocódigo se presenta en el cuadro 20.

Cuadro 20. Pseudocódigo para calcular el área y perímetro de un rectángulo

```
1.  Inicio
2.      Entero: b, h, a, p
3.      Leer b, h
4.      a = b * h
5.      p = 2 (b + h)
6.      Escribir "área:", a
7.      Escribir "perímetro:", p
8.  Fin algoritmo
```

Para verificar que el algoritmo es correcto se realiza la prueba de escritorio, paso a paso, así:

Paso 2. Se declaran variables: b, h, a, p correspondientes a: base, altura, área y perímetro respectivamente.

Paso 3. Se lee desde teclado base y altura y se almacena en las variables b y h, supóngase los valores 5 y 8

Paso 4. Se calcula el área, 5 * 8 = 40 (a = b * h) y se almacena en la variable a

Paso 5. Se calcula el perímetro, 2 * (5 + 8) = 26 y se guarda en la variable p

Pasos 6 y 7. Se muestra el contenido de las variables a y p con su respectivo mensaje.

En el cuadro 21 se presentan los resultados de la verificación con tres conjuntos de datos.

Cuadro 21. Verificación del algoritmo área y perímetro de un rectángulo

Ejecución	**b**	**h**	**a**	**p**	**Salida**
1	5	8	40	26	área: 40 perímetro: 26
2	3	4	12	14	área: 12 perímetro: 14
3	8	4	32	24	área: 32 perímetro: 24

Ejemplo 7. Tiempo dedicado a una asignatura

Se sabe que un profesor universitario contratado como hora cátedra dedica una hora a preparar una clase de dos horas y por cada cuatro horas de clase desarrolladas hace una evaluación, la cual le toma dos horas calificar. Si al docente se le asigna un curso de *n* horas al semestre, ¿cuántas horas trabajará en total?

La mejor forma de comprender un problema es tomar un caso particular y desarrollarlo. Supóngase que el profesor ha sido contratado para el curso de matemáticas cuyo número de horas (nh) es de 64 al semestre.

Si para cada par de horas de clase dedica una hora a su preparación se tiene que el tiempo de preparación (tp) es:

tp = 64 / 2
tp = 32 horas

En general: tp = nh/2

Se sabe también que hace una evaluación por cada cuatro horas de clase, de manera que se puede calcular el número de evaluaciones (ne) que realizará durante el semestre:

ne = 64 / 4
ne = 16

Para cualquier caso: ne = nh / 4

Ahora, si para calificar cada evaluación requiere dos horas, el tiempo dedicado en el semestre a calificar (tc) es:

tc = 16 * 2
tc = 32

Utilizando variables: tc = ne * 2

Finalmente, para obtener el tiempo total (tt) que el docente dedica al curso de matemáticas sólo hay que sumar el tiempo de preparación, el tiempo de desarrollo de las clases y el tiempo de calificación de exámenes, así:

tt = 32 + 64 + 32
tt = 128

Esto es: tt = tp + nh + tc

De esta manera se obtiene que para un curso de 64 horas de clase, el docente dedica 128 horas de trabajo. Como se quiere calcular este valor para cualquier curso es necesario diseñar un algoritmo, al cual se le proporciona el número de horas de clase semestral y el devolverá el total de horas dedicadas.

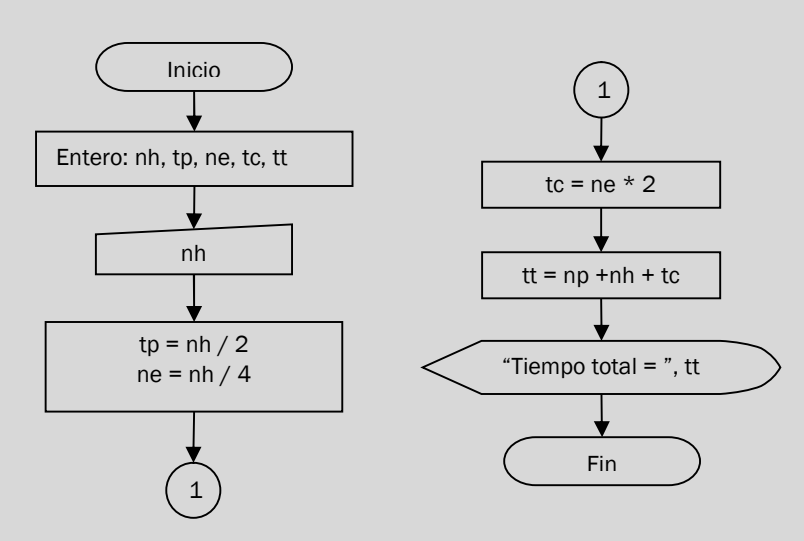
De lo anterior se deduce que:

Entrada: número de horas de clase (nh)
Salida: tiempo total dedicado (tt)
Cálculos a realizar:
tp = nh/2
ne = nh / 4
tc = ne * 2
tt = tp + nh + tc

El algoritmo, en notación de diagrama de flujo se presenta en la figura 22.

Para tener la seguridad de que el algoritmo realiza los cálculos correctamente se realiza la prueba con tres casos diferentes, los resultados se presentan en el cuadro 22.

Figura 22. Diagrama de flujo para calcular el tiempo dedicada a una asignatura



Cuadro 22. Verificación del algoritmo tiempo dedicado a una asignatura

Ejec.	nh	tp	ne	tc	tt	Salida
1	64	32	16	32	128	Tiempo total = 128
2	80	40	20	40	160	Tiempo total = 160
3	96	48	24	48	192	Tiempo total = 192

4.1.5 Ejercicios propuestos

Diseñar los algoritmos para solucionar los siguientes problemas y representarlos mediante pseudocódigo, diagrama de flujo o diagrama N-S.

1. Leer tres notas y calcular el promedio

2. Calcular el área de un triángulo

3. Calcular el área y el perímetro de un círculo

4. Calcular el volumen y la superficie de un cilindro

5. Dado el ancho, largo y alto de una caja, calcular el volumen y la cantidad de papel (en cm^2) necesario para cubrirla.

6. Leer un número entero y separar sus dígitos en: miles, centenas, decenas y unidades.

7. Calcular el interés y el valor futuro de una inversión con interés simple.

8. Calcular el interés y valor futuro de una inversión con interés compuesto.

9. Si la universidad financia las matrículas de los estudiantes a cuatro cuotas mensuales iguales con un interés del 2% sobre el saldo. Dado el valor de matrícula y el número de cuotas, un estudiante desea saber ¿Cuál será el valor de cada cuota? ¿Cuánto pagará en total?

10. Un vendedor recibe un sueldo base más el 10% de comisión sobre sus ventas. Si en un mes cualquiera hace tres ventas por valores: v1, v2 y v3, ¿cuánto recibirá por comisión? y ¿cuánto en total?

11. Un cliente de un supermercado adquiere n productos a precio unitario p. Si por temporada el producto tiene un descuento del 10% que se hace efectivo en caja ¿cuál es el valor del descuento? ¿cuánto deberá pagar?

12. Un estudiante desea saber cuál será su calificación final en Programación. Dicha calificación se compone del promedio de tres notas parciales. Cada nota parcial se obtiene a partir de un taller, una evaluación teórica y una evaluación práctica. Los talleres equivalen al 25% de la nota del parcial, las evaluaciones teóricas al 35% y las evaluaciones prácticas al 40%.

13. Un viajero desea conocer cuántos dólares obtendrá por su capital en pesos.

14. Facturar el servicio de electricidad. El consumo mensual se determina por diferencia de lecturas.

15. Las utilidades de una empresa se distribuyen entre tres socios así: socio A = 40%, socio B = 25% y socio C = 35%. Dada una cantidad de dinero ¿cuánto corresponderá a cada uno?

16. El dueño de una tienda compra un artículo por x pesos y desea obtener el 30% de utilidad. ¿Cuál es el precio de venta del artículo?

17. Un almacén tiene como política obtener el 30% de utilidad sobre cada artículo y por temporada ofrece un descuento del 15% sobre el precio de venta en todos sus productos. Dado el costo de un producto calcular el precio de venta, de manera que pueda efectuar el descuento sobre el precio de venta y obtener el 30% de utilidad sobre el costo.

18. Tres personas deciden invertir su dinero para fundar una empresa. Cada una de ellas invierte una cantidad distinta. Obtener el porcentaje que cada cual invierte con respecto a la cantidad total invertida.

19. Calcular el sueldo de un empleado que ha trabajado n horas extras diurnas y m horas extras nocturnas, siendo que cada hora extra diurna tiene un incremento del 25% sobre el valor de una hora corriente y cada hora extra nocturna un incremento del 35%.

20. Un estudiante desea saber la nota mínima que deberá obtener en la evaluación final de cálculo después de conocer las notas de los dos parciales, sabiendo que la materia se aprueba con 3.0 y la nota definitiva se obtiene de la siguiente manera: 30% para cada uno de los parciales y 40% para el final.

21. Dado el valor del capital de un préstamo, el tiempo y el valor pagado por concepto de interés, calcular la tasa de interés que se aplicó.

22. Conociendo el tiempo que un atleta tarda en dar una vuelta al estadio (400 m) se requiere estimar el tiempo que tardará en recorrer los 12 km. establecido para una competencia.

23. Resolver una ecuación cuadrática ($aX^2 + bX + c = 0$) teniendo en cuenta que a, b y c son valores enteros y pueden ser positivos o negativos.

24. Dado el valor que un cliente paga por un producto, calcular qué valor corresponde al costo del producto y cuánto al IVA. Considerando que el porcentaje del IVA puede variar en el tiempo y de un producto a otro, este dato se lee por teclado.

25. Un profesor diseña un cuestionario con n preguntas, estima que para calificar cada pregunta requiere m minutos. Si el cuestionario se aplica a x estudiantes, cuánto tiempo (horas y minutos) necesita para calificar todos los exámenes.

4.2 ESTRUCTURAS DE DECISIÓN

En la programación, como en la vida real, es necesario evaluar las circunstancias y actuar en consecuencia, pues no siempre se puede establecer por anticipado el curso de las acciones. En la ejecución de un programa, muchas expresiones estarán supedita-das al cumplimiento de determinadas condiciones; por ello, el programador ha de identificar estas situaciones y especificar, en el programa, qué hacer en cada caso (Timarán *et al*, 2009).

Todo problema, tanto en programación como en otros ámbitos, incluye un conjunto de variables que pueden tomar diferentes valores. La presencia o ausencia de determinadas variables, así como su comportamiento, requieren acciones diferentes en la solución. Esto hace que los programas no sean secuencias simples de instrucciones, sino estructuras complejas que implementan saltos y bifurcaciones según el valor de las variables y las condiciones definidas sobre éstas

Por ejemplo, para realizar una división se requieren dos variables: dividendo y divisor, que pueden contener cualquier número entero o real. No obstante, la división sobre cero no está determinada y por tanto se tiene como condición que el divisor sea diferente de cero. Al implementar esta operación en un programa se debe verificar el cumplimiento de esta restricción y decidir si se realiza el cálculo o se muestra un mensaje de error. Si no se implementa esta decisión y al ejecutarse el programa la variable toma el valor cero se presentará un error de cálculo y la ejecución se interrumpirá.

Implementar una decisión implica evaluar una o más variables para determinar si cumple una o más condiciones y establecer un flujo de acciones para cada resultado posible. Las acciones pueden consistir en ejecutar una o más expresiones o en omitirlas, o en seleccionar una secuencia de instrucciones de entre dos o más alternativas disponibles.

4.2.1 Condición

En esta sección se menciona repetidas veces el término condición, de manera que parece oportuno hacer mayor claridad sobre el mismo. Una condición es una expresión relacional o lógica que puede o no cumplirse dependiendo de los valores de las variables involucradas en la expresión. Cuando se utiliza una expresión relacional, la condición es una comparación entre dos variables del mismo tipo o una variable y una constante, mientras que cuando se utiliza una expresión lógica se tiene una o más expresiones relacionales combinadas con operadores lógicos: *y, o, no*; cuyo resultado depende de la tabla de verdad del correspondiente operador.

Los siguientes son ejemplos de condiciones que contienen expresiones relacionales:

a. *x = 5*
b. *y <= 10*
c. *x > y*

Al escribir el programa se especifica la condición mediante expresiones como estas, pero al ejecutarlo se evalúan y el resultado puede ser verdadero o falso dependiendo del valor de las variables. Supóngase que:

x = 8
y = 9

En tal caso se tendría que la condición *a* no se cumple por ende genera un resultado *falso*, la *b* si se cumple generando así un resultado *verdadero* y la *c* también devuelve *falso*.

Obsérvese algunos ejemplos de condiciones formadas por expresiones lógicas:

a. *x > 0* ***Y*** *x < 10*
b. *x == 5* ***O*** *y == 9*
c. ***NO****(x>y)*

La condición *a* devuelve *verdadero* si las dos expresiones relacionales son verdaderas; la condición *b* devuelve *verdadero* si al menos una de las expresiones relacionales que la conforman es verdadera y la *c* niega el resultado de la expresión relacional que está entre paréntesis. Si las variables tiene los valores indicados anteriormente (*x = 8, y = 9*) la condición *a* es *verdadera*, la *b* es *verdadera* y la *c* es *verdadera*.

4.2.2 Tipos de decisiones

Las decisiones pueden ser de tres clases: simples, dobles y múltiples.

Decisión simple**:** consiste en decidir si ejecutar u omitir una instrucción o un conjunto de instrucciones. En este caso se determina qué debe hacer el programa si la condición es verdadera, pero si no lo es, simplemente se pasa el control a la instrucción que está después de la estructura de decisión.

Un ejemplo de decisión simple se presenta cuando se calcula el valor absoluto de un número. El valor absoluto de un número es el mismo número, pero si éste es negativo es necesario multiplicarlo por -1 para cambiar de signo. Nótese que esta operación sólo es necesaria en el caso de los números negativos. Para incluir decisiones simples en un algoritmo se utiliza la sentencia *Si*.

Decisión doble**:** se presenta cuando se tienen dos alternativas de ejecución y dependiendo del resultado de la evaluación de la condición se ejecuta la una o la otra. Si la condición es

verdadera se ejecuta la primera instrucción o bloque de instrucciones y si es falsa, la segunda.

Como ejemplo de este tipo de decisión se puede tomar el caso en que se evalúa una nota para decidir si un estudiante aprueba o reprueba una asignatura, dependiendo si la nota es mayor o igual a 3.0.

Decisión múltiple**:** consiste en evaluar el contenido de una variable y dependiendo del valor de ésta ejecutar una secuencia de acciones. En este caso, el conjunto de valores para la variable debe ser mayor a dos y solo se evalúa la condición de igualdad. Cada posible valor está asociado a una secuencia de ejecución y éstas son mutuamente excluyentes. Este tipo de decisiones se implementan haciendo uso de la sentencia *Según sea*.

Como ejemplo de decisión múltiple considérese el caso de un algoritmo que permite ejecutar cualquiera de las operaciones aritméticas básicas: suma, resta, multiplicación y división, sobre un conjunto de números. Al ejecutarse, el usuario escoge la operación y el algoritmo realiza la operación correspondiente.

4.2.3 Estructura SI

Esta instrucción evalúa un valor lógico, una expresión relacional o una expresión lógica y retorna un valor lógico, con base en éste se toma una decisión simple o doble.

Decisión simple

Como decisión simple su sintaxis en pseudocódigo es:

SI <condición> ENTONCES
    Instrucciones
FIN SI

En las figuras 23 y 24 se presenta su representación en diagrama de flujo y N-S, respectivamente.

Figura 23. Decisión simple en notación diagrama de flujo

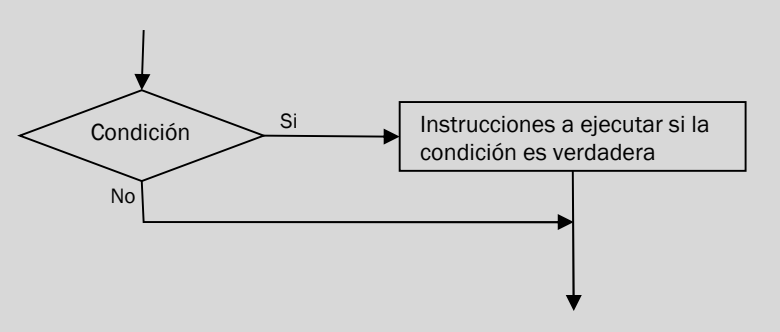
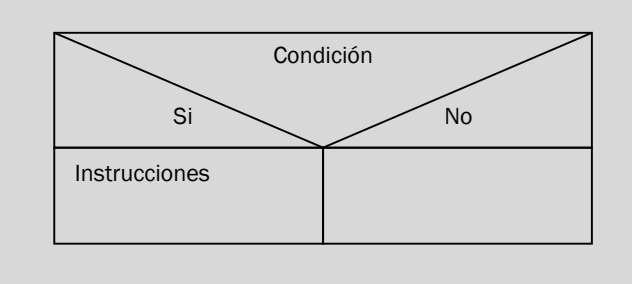


Figura 24. Decisión simple en notación diagrama N-S



Ejemplo 8. Calcular el valor absoluto de un número

El valor absoluto de un número es el mismo número para el caso de los positivos y el número cambiado de signo para los negativos; es decir, el valor absoluto es la distancia que hay desde el 0 al número y como las distancias no pueden ser negativas, éste siempre será positivo.

|5| = 5
|-3| = 3

En el cuadro 23 se presenta el pseudocódigo de este ejercicio.

Cuadro 23. Pseudocódigo para calcular el valor absoluto de un número

```
1.  Inicio
2.      Entero: num
3.      Leer num
4.      Si num < 0 entonces
5.          num = num * -1
6.      Fin si
7.      Escribir “Valor absoluto = ”, num
8.  Fin algoritmo
```

La estructura de decisión se aplica en la línea 4 para determinar si el número es negativo, si la evaluación de la condición da un resultado verdadero se ejecuta la línea 5, si da un resultado falso termina la estructura *Si* y se ejecuta la instrucción que está después de *Fin si*, en la línea 7.

Si al ejecutar este algoritmo se introduce el número 5 se obtiene que, al llegar a la decisión, la condición no se cumple y por tanto se procede a escribir el mismo número. Si en

una segunda ejecución se introduce el número -3, al evaluar la condición ésta genera como resultado *verdadero*, por tanto se ejecuta la operación que está dentro de la decisión: multiplicar el número por -1 y finalmente se muestra el número 3. Estos resultados se aprecian en el cuadro 24.

Cuadro 24. Verificación del algoritmo para calcular valor absoluto

Ejecución	num	Salida
1	5	Valor absoluto = 5
2	-3	Valor absoluto = 3

En la figura 25 se presenta el diagramas de flujo y en la 26 el diagrama N-S.

Figura 25. Diagrama de flujo de valor absoluto de un número

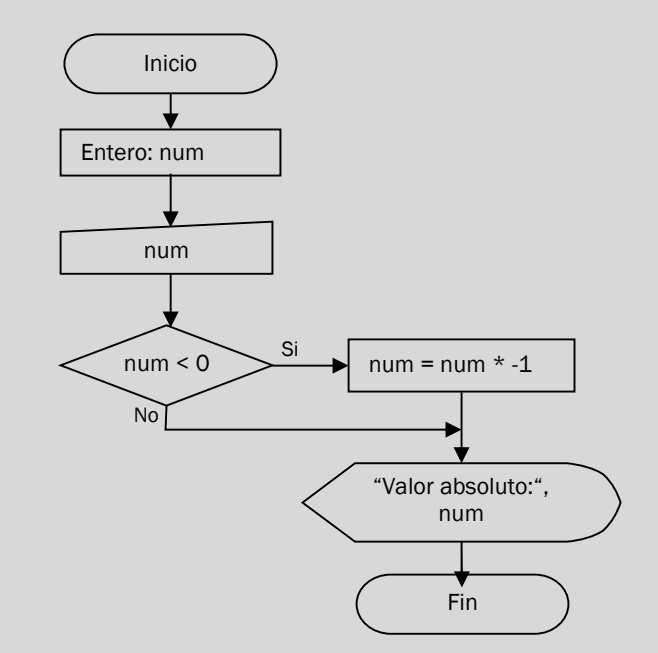
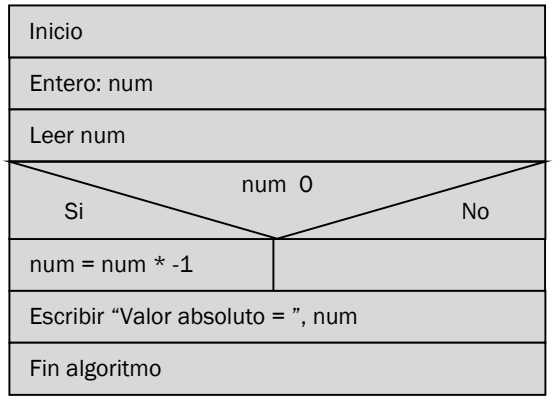


Figura 26. Diagrama N-S para calcular el valor absoluto de un número



Decisión doble

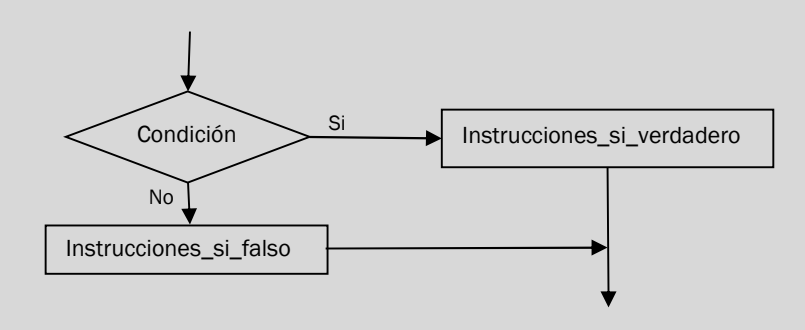
Se implementa una decisión doble cuando se cuenta con dos opciones para continuar la ejecución del algoritmo y éstas dependen de una condición; es decir, que se cuenta con una o más instrucciones para el caso que la condición sea verdadera y con otra u otro conjunto de instrucciones para el caso de que sea falsa.

Las decisiones simples se utilizan para incluir saltos en la ejecución del algoritmo o del programa, las decisiones dobles permiten hacer bifurcaciones. Su sintaxis en pseudocódigo es:

SI <condición> ENTONCES
Instrucciones_si_verdadero
SI NO
Instrucciones_si_falso
FIN SI

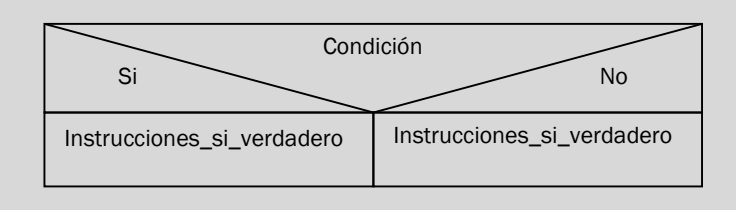
En las figuras 27 y 28 se presenta su representación en diagrama de flujo y N-S, respectivamente.

Figura 27. Decisión doble en notación diagrama de flujo



Observe que en pseudocódigo se escribe la instrucción *Fin Si* para indicar hasta donde se extiende la estructura condicional. En diagrama de flujo, el fin del condicional está determinado por la unión de los dos caminos, marcados como *Si* y *No*. En el diagrama N-S la estructura condicional tiene dos bloques, en el de la izquierda se escriben las instrucciones que deben ejecutarse cuando la condición se cumple y en el de la derecha se colocan las que deben ejecutarse cuando la condición no se cumple, el fin del condicional estará marcado por la terminación de los dos bloques y la continuación en una sola caja.

Figura 28. Decisión simple en notación diagrama N-S



Ejemplo 9. División

Como se mencionó, antes de ejecutar una división es necesario verificar que el divisor sea diferente de cero, ya que en caso contrario se generará un error de cómputo y el programa se cancelará. En consecuencia, es necesario utilizar la sentencia *Si* para decidir si se ejecuta la división o se muestra un mensaje de error.

El algoritmo para realizar una división, representado mediante pseudocódigo, se presenta en el cuadro 25. En la línea 4 se especifica la condición que debe cumplirse para proceder a efectuar la división, en caso de que la condición no se cumpla la ejecución salta a la cláusula *Si no*, en la línea 7 y se ejecuta la línea 8, mostrándose un mensaje de error.

Cuadro 25. Pseudocódigo para realizar una división

```
1.  Inicio
2.      Entero: dividendo, divisor, cociente
3.      Leer dividendo, divisor
4.      Si divisor != 0 entonces
5.          cociente = dividendo / divisor
6.          Escribir cociente
7.      Si no
8.          Escribir “Error, divisor = 0”
9.      Fin si
10. Fin algoritmo
```

Para probar si el algoritmo funciona correctamente se ejecuta paso a paso, se introduce un par de números, se verifica el cumplimiento de la condición y la ruta que toma dependiendo de ésta. En el cuadro 26 se presentan tres pruebas.

Cuadro 26. Verificación del algoritmo para hacer una división

Ejec.	Dividendo	divisor	cociente	Salida
1	15	3	5	5
2	8	0		Error, divisor = 0
3	12	3	4	4

La solución de este ejercicio en notación de diagrama de flujo se presenta en la figura 29 y en diagrama N-S en la figura 30.

Ejemplo 10. Número mayor y número menor

Este ejercicio consiste en leer dos números enteros diferentes y decidir cuál de ellos es el mayor y cuál es el menor.

Para solucionar este ejercicio es necesario declarar dos variables (*x, y*), introducir los números y guardarlos en ellas, luego decidir si *x* es mayor que *y* o lo contrario.

Figura 29. Diagrama de flujo para hacer una división

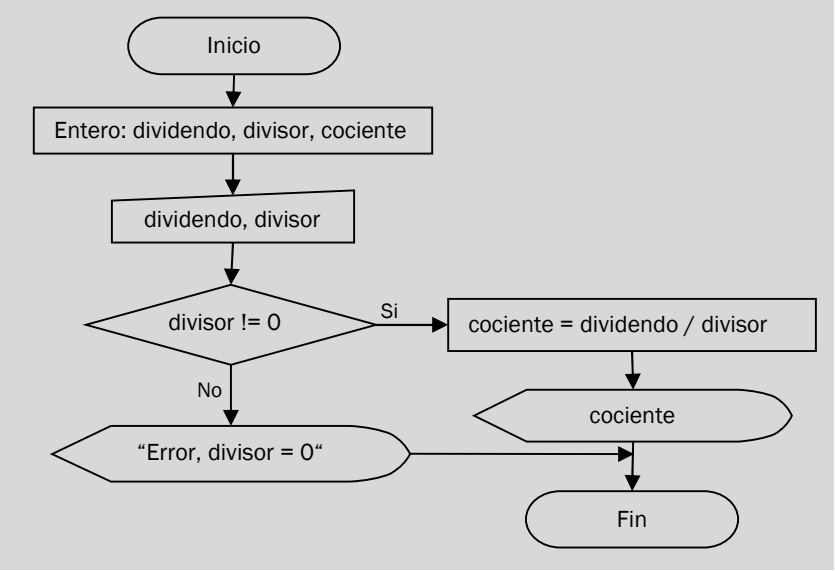
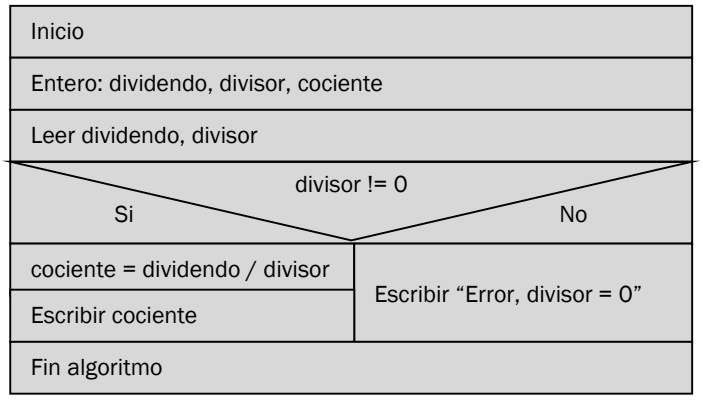


Figura 30. Diagrama N-S para hacer una división



El algoritmo del ejemplo 10, en notación de pseudocódigo, se presenta en el cuadro 27.

Cuadro 27. Pseudocódigo para identificar número mayor y menor

```
1.  Inicio
2.      Entero: x, y
3.      Leer x, y
4.      Si x > y entonces
5.          Escribir "Número mayor: ", x
6.          Escribir "Número menor", y
7.      Si no
8.          Escribir "Número mayor: ", y
9.          Escribir "Número menor", x
10.     Fin si
11. Fin algoritmo
```

En el cuadro 28 se muestra el comportamiento del algoritmo al ser probado con dos conjuntos de datos.

Cuadro 28. Verificación del algoritmo número mayor y menor

x	y	x > y	Salida
23	12	Verdadero	Número mayor: 23 Número menor: 12
5	11	Falso	Número mayor: 11 Número menor: 5

Encontrará más ejemplos desarrollados en la sección 4.2.6.

4.2.4 Estructura SEGÚN SEA

Muchas decisiones deben tomarse no solo entre dos alternativas sino de un conjunto mayor. Estos casos bien pueden solucionarse utilizando decisiones dobles anidadas; sin embargo, en favor de la claridad del algoritmo y la facilidad para el programador, es mejor utilizar una estructura de decisión múltiple, la cual es fácil de pasar a un lenguaje de programación, ya que éstos incluyen alguna instrucción con este fin.

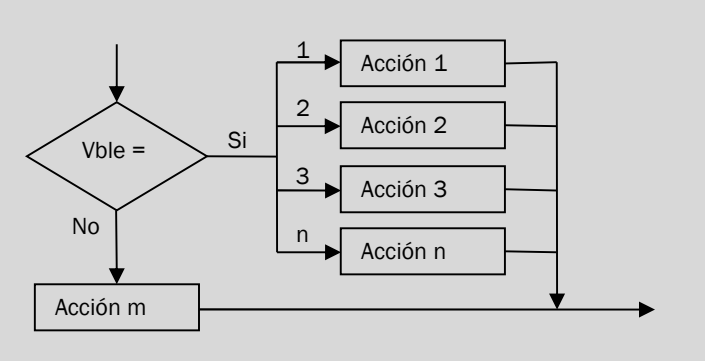
La instrucción *Según sea* determina el valor de una variable y dependiendo de éste sigue un curso de acción. Es importante tener en cuenta que solo se verifica la condición de igualdad entre la variable y la constante. Con base en Joyanes (1996) se propone la siguiente sintaxis:

Según sea *<variable >* ***hacer***
Valor_1:
Acción 1
Valor_2:
Acción 2
Valor_3:
Acción 3
Valor_n
Acción n
Si no
Acción m
Fin según sea

Dado que se evalúa la igualdad entre la variable y la constante, todas las alternativas son mutuamente excluyentes; eso significa que sólo se podrá ejecutar un conjunto de acciones. Si ninguna de las alternativas se cumple, por no presentarse ninguna igualdad, se ejecutará el grupo de acciones asociadas con la cláusula *si no***.** Esta última es opcional, en caso de no aparecer y no cumplirse ninguno de los casos, simplemente la ejecución del algoritmo continuará en la línea siguiente a *fin según sea*.

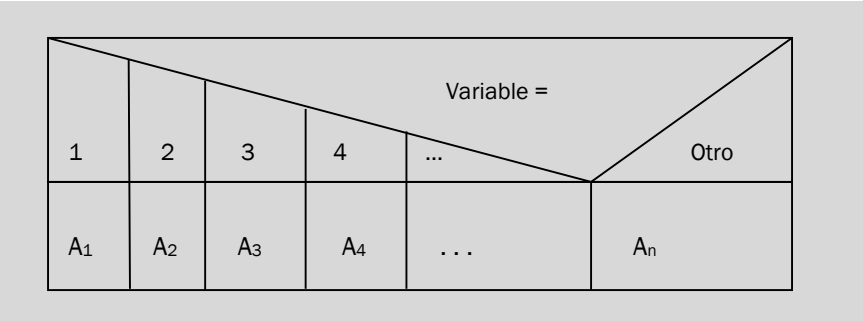
En diagrama de flujo la decisión múltiple se representa mediante un rombo en el que se coloca la variable y dos salidas, una de ellas se marca con la palabra *Si* y se subdivide según los posibles valores de la variable, los mismos que se colocan junto al flujo que les corresponde; en la otra salida se escribe la palabra *No* y las acciones a realizar en caso de que el valor de la variable no coincida con ninguno de los valores establecidos, como se muestra en la figura 31.

Figura 31. Decisión múltiple en notación diagrama de flujo



En diagrama N-S se representa mediante una caja con un triángulo invertido en el que se coloca la variable y el signo igual, y se divide la caja en tantas sub-cajas como alternativas tenga la decisión, incluyendo una caja para el caso en que la variable tome un valor diferente a los contemplados en las opciones, esta se coloca bajo el rótulo *Otro*, como se muestra en la figura 32.

Figura 32. Decisión múltiple en notación N-S



Ejemplo 11. Número romano

Este algoritmo lee un número arábigo y muestra su equivalente en romano, pero sólo contempla los 10 primeros números, de manera que si el número digitado es superior a 10 se mostrará el mensaje: número no válido.

La solución de ejercicios de este tipo se facilita utilizando la estructura de decisión múltiple, ya que para cada valor hay un equivalente.

El pseudocódigo se muestra en el cuadro 29, en el que se observa la estructura de decisión múltiple implementada entre las líneas 4 y 17.

Cuadro 29. Pseudocódigo para números romanos

```
1.  Inicio
2.      Entero: num
3.      Leer num
4.      Según sea num hacer
5.          1: Escribir “I”
6.          2: Escribir “II”
7.          3: Escribir “III”
8.          4: Escribir “IV”
9.          5: Escribir “V”
10.         6: Escribir “VI”
11.         7: Escribir “VII”
12.         8: Escribir “VIII”
13.         9: Escribir “IX”
14.         10: Escribir “X”
15.     Si no
16.         Escribir “Número no valido”
17.     Fin según sea
18. Fin algoritmo
```

En el pseudocódigo de este ejercicio se programa 10 posibles valores para la variable *num*, éstos se escriben después de la instrucción *según sea*, seguidos de dos puntos, y para cada uno de ellos se especifica las instrucciones que se deben realizar si el contenido de la variable coincide con dicho valor. Finalmente se tiene la cláusula *si no* para cualquier otro valor mayor a 10 o menor a 1.

Los diagrama de flujo y N-S se presentan en la figuras 33 y 34 respectivamente. En los diagramas se aprecia las diferentes rutas de ejecución que se generan a partir de la estructura de decisión múltiple.

Figura 33. Diagrama de flujo para número romano

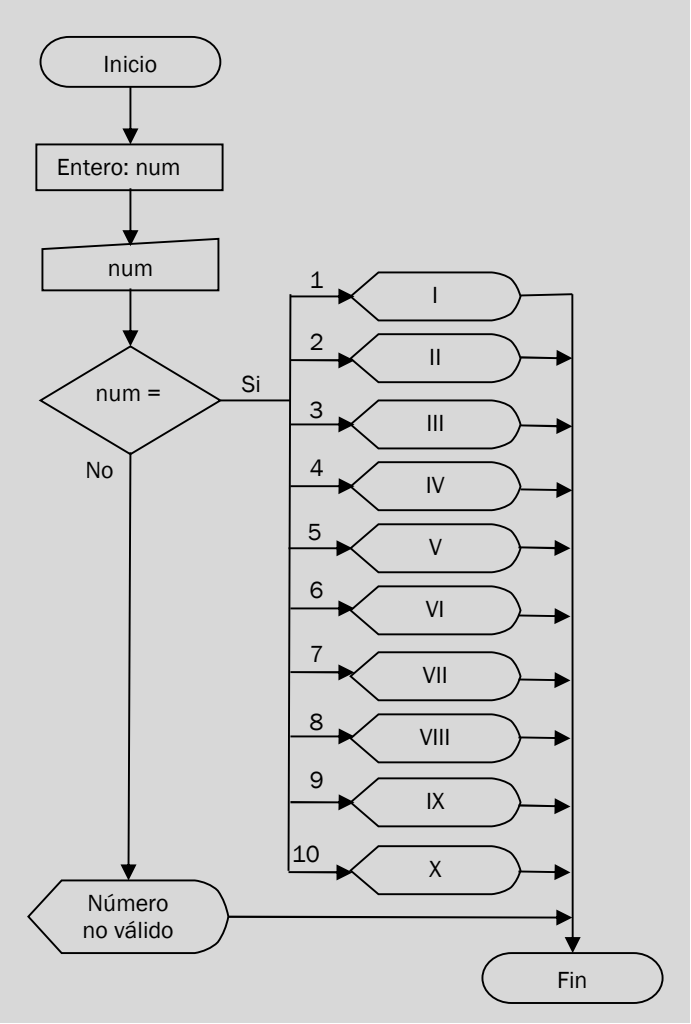
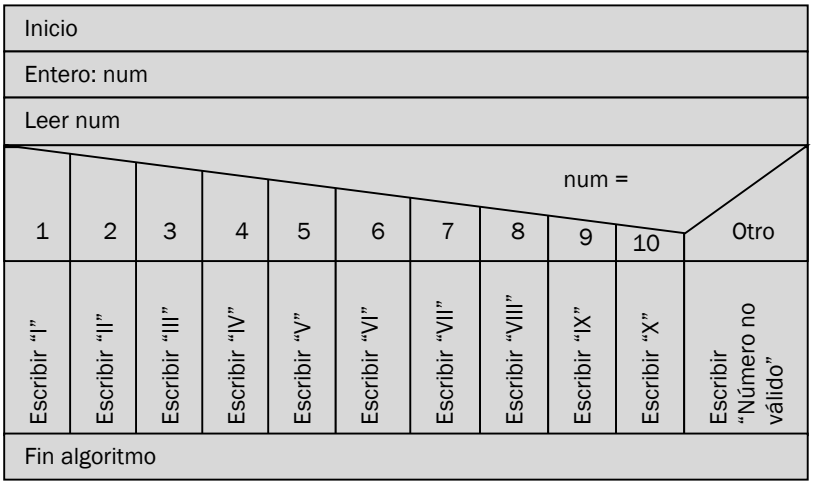


Figura 34. Diagrama N-S para número romano



En el cuadro 30 se muestra los resultados de tres ejecuciones con diferentes números.

Cuadro 30. Verificación del algoritmo número romano

Ejecución	num	Salida
1	3	III
2	9	IX
3	12	Número no valido

Ejemplo 12. Nombre del día de la semana

Este algoritmo lee un número entre uno y siete correspondiente al día de la semana y muestra el nombre del día, haciendo corresponder el uno (1) con lunes, dos con martes y así sucesivamente, para lo cual utiliza la sentencia *según sea*. Si el número digitado es menor a uno o mayor a siete, muestra un mensaje de error. El pseudocódigo se muestra en el cuadro 31, el diagrama de flujo en la figura 35 y el diagrama N-S en la 36.

En este ejercicio no se escribe el nombre del día en cada caso de la decisión, se declara una variable y se le asigna el nombre que corresponda (líneas 5 a 11). Al final, después de haber cerrado la estructura de decisión, se muestra el nombre del día (línea 15).

Cuadro 31. Pseudocódigo del algoritmo día de la semana

```
1.  Inicio
2.      Entero: día
        Cadena: nomdía
3.      Leer día
4.      Según sea día hacer
5.          1: nomdía = “Lunes”
6.          2: nomdía = “Martes”
7.          3: nomdía = “Miércoles”
8.          4: nomdía = “Jueves”
9.          5: nomdía = “Viernes”
10.         6: nomdía = “Sábado”
11.         7: nomdía = “Domingo
12.     Si no
13.         nomdía = “Número no válido”
14.     Fin según sea
15.     Escribir nomdía
16. Fin algoritmo
```

Figura 35. Diagrama de flujo del algoritmo nombre del día de la semana

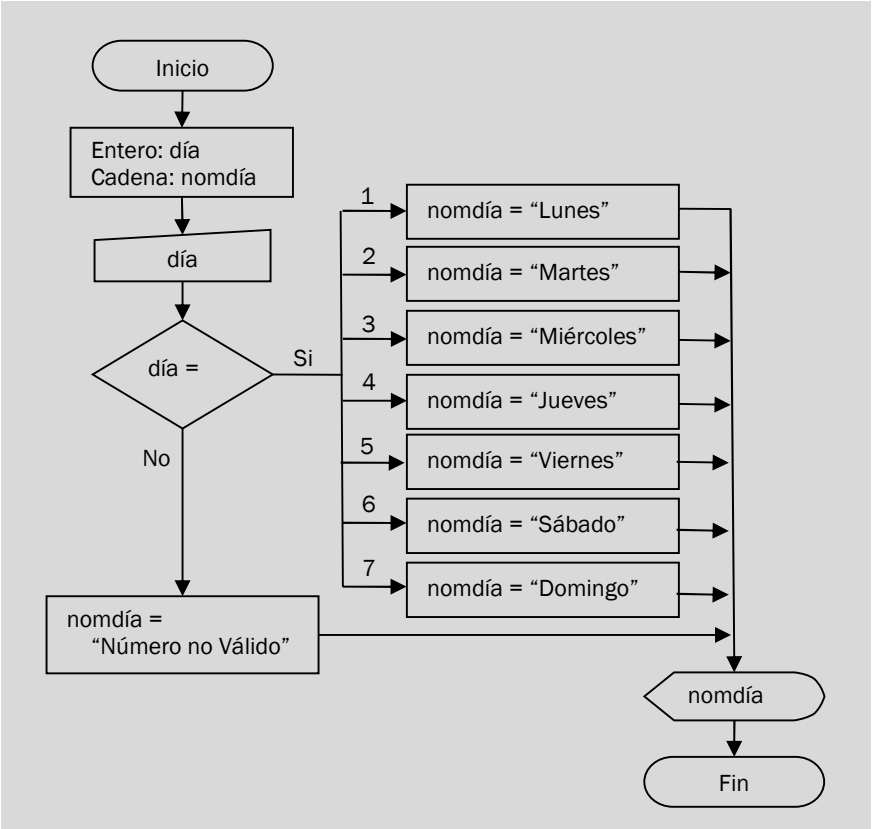
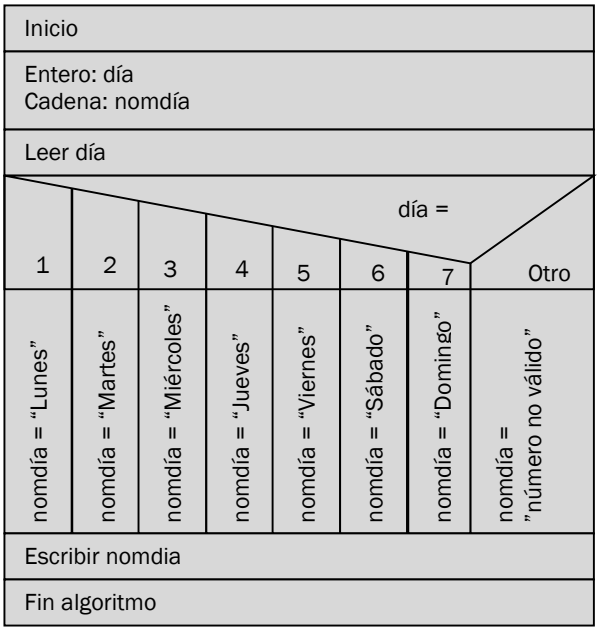


Figura 36. Diagrama N-S del algoritmo día de la semana



En el cuadro 32 se muestran tres casos utilizados para verificar el funcionamiento del algoritmo.

Cuadro 32. Verificación del algoritmo día de la semana

Ejecución	Día	nomdía	Salida
1	6	Sábado	Sábado
2	4	Jueves	Jueves
3	9	Número no válido	Número no válido

4.2.5 Decisiones anidadas

Se denominan anidadas a las decisiones que se escriben una dentro de otra; lo que significa que después de haber tomado una decisión es necesario tomar otra. En pseudocódigo el anidamiento tiene la forma de la estructura que se muestra en el cuadro 33.

Si condición-1 es verdadera se evalúa condición-2 y si esta también es verdadera se ejecuta instrucciones-1. De forma que instrucciones-1 se ejecuta únicamente si condición-1 y condición-2 son verdaderas; instrucciones-2, si condición-1 es verdadera y condición-2 es falsa.

Cuadro 33. Decisiones anidadas

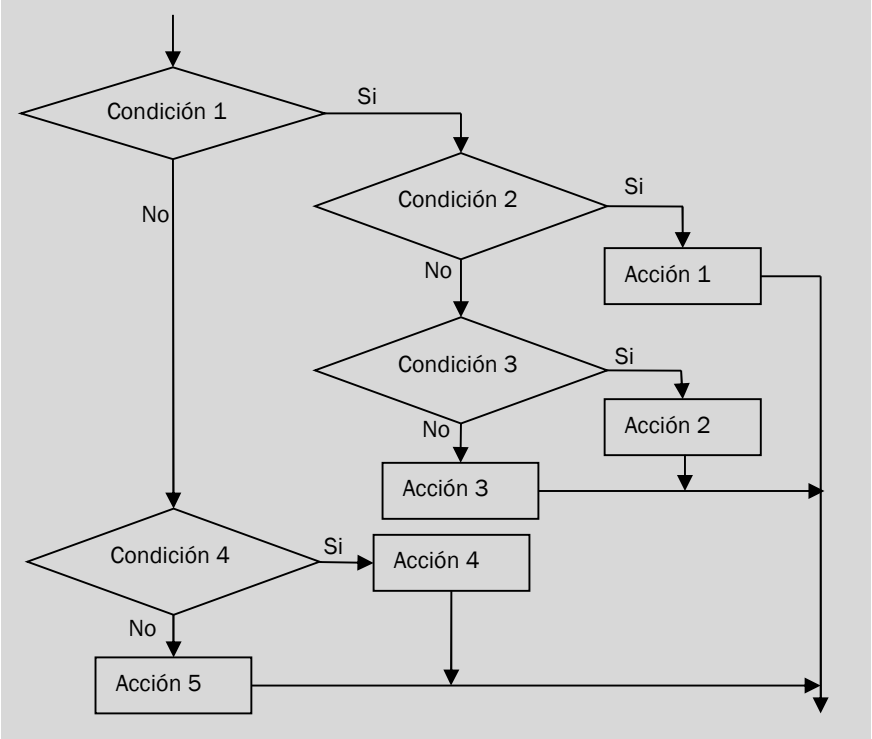
```
Si <condición 1> entonces
        Si <condición 2> entonces
            Instrucciones 1
        Si no
                Si <condición 3> entonces
                        Instrucciones 2
                Si no
                        Instrucciones 3
                Fin si
        Fin si
Si no
        Si <condición 4> entonces
            Instrucciones 4
        Si no
            Instrucciones 5
        Fin si
Fin si
```

Cuando la segunda decisión se escribe para el caso de que la primera sea verdadera se dice que se anida por verdadero, como ocurre con la condición-2. Si la segunda decisión se evalúa cuando la primera no se cumple, se dice que se anida por falso. Tanto por verdadero como por falso se pueden anidar todas las decisiones que sean necesarias.

El anidamiento se puede hacer con la estructura *Si* de igual manera con *Según sea* y se pueden combinar las dos: un *Si* dentro de un *Según sea* o lo contrario, tantos niveles como la solución del problema lo requiera.

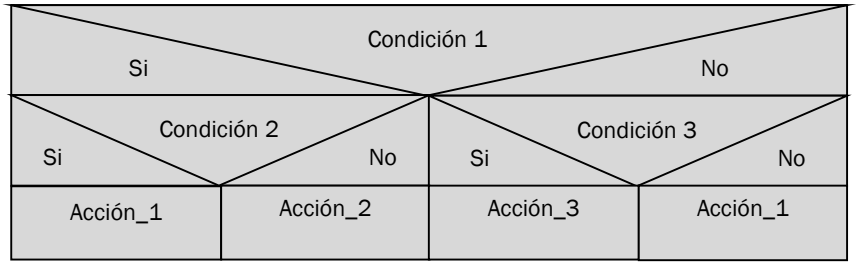
La representación en diagrama de flujo se muestra en la figura 37. En este tipo diagrama es muy fácil de apreciar el anidamiento de las decisiones y de comprender la dependencia de las operaciones con respecto a las decisiones, basta con evaluar la condición y seguir la flecha que corresponda.

Figura 37. Diagrama de flujo de decisiones anidadas



El Diagrama N-S puede parecer complicado, pero no lo es, basta con leer una caja a la vez, de arriba hacia abajo y de izquierda a derecha (ver figura 38). *Condición 1* establece dos caminos, siguiendo por el de la izquierda encuentra la *Condición 2*, ésta también proporciona dos rutas. Si la *Condición 1* es falsa se toma el camino de la derecha y se encuentra con la *Condición 3*, la cual proporciona también dos caminos a seguir, dependiendo del resultado de la condición.

Figura 38. Diagrama N-S de decisiones anidadas



Ejemplo 13. Comparación de dos números

Dados dos números enteros, estos pueden ser iguales o diferentes, si son diferentes ¿cuál de ellos es mayor? y ¿cuál es menor?

En este caso se leen dos números desde el teclado y se almacenan en dos variables: n1 y n2.

El algoritmo requiere dos decisiones, la primera está dada por la condición:

n1 = n2 ?

Si esta condición es verdadera muestra un mensaje, si no lo es debe tomar otra decisión con base en la condición:

n1 > n2 ?

Si esta condición es verdadera mostrará un resultado, si es falsa otro.

En el cuadro 34 se presenta la solución en notación de pseudocódigo, los diagramas de flujo y N-S se presentan en las figuras 39 y 40.

Cuadro 34. Pseudocódigo del algoritmo comparar números

```
1.  Inicio
2.      Entero: n1, n2
3.      Leer n1, n2
4.      Si n1 = n2 entonces
5.          Escribir “números iguales”
6.      Si no
7.          Si n1 > n2 entonces
8.              Escribir n1, “Mayor”
9.              Escribir n2, “Menor”
10.         Si no
11.             Escribir n2, “Mayor”
12.             Escribir n1, “Menor”
13.         Fin si
14.     Fin si
15. Fin algoritmo
```

Para verificar la corrección del algoritmo se hacen tres pruebas. En la primera ejecución se introduce el número 3 para cada una de las variables, en consecuencia al evaluar la primera condición (línea 4) se obtiene el valor verdadero y se muestra el mensaje “números iguales”

(línea 5). En la segunda ejecución se ingresan los números 5 y 2, esta vez la primera condición no se cumple y la ejecución continúa en la línea 7 donde se encuentra con otra condición, ésta sí se cumple y por ende se ejecutan las líneas 8 y 9. En la tercera ejecución se introducen los números 6 y 8, al evaluar la primera condición ésta no se cumple, salta a la línea 7 y evalúa la segunda condición, dado que ésta también es falsa, se ejecutan las líneas 11 y 12.

Figura 39. Diagrama de flujo del algoritmo comparar números

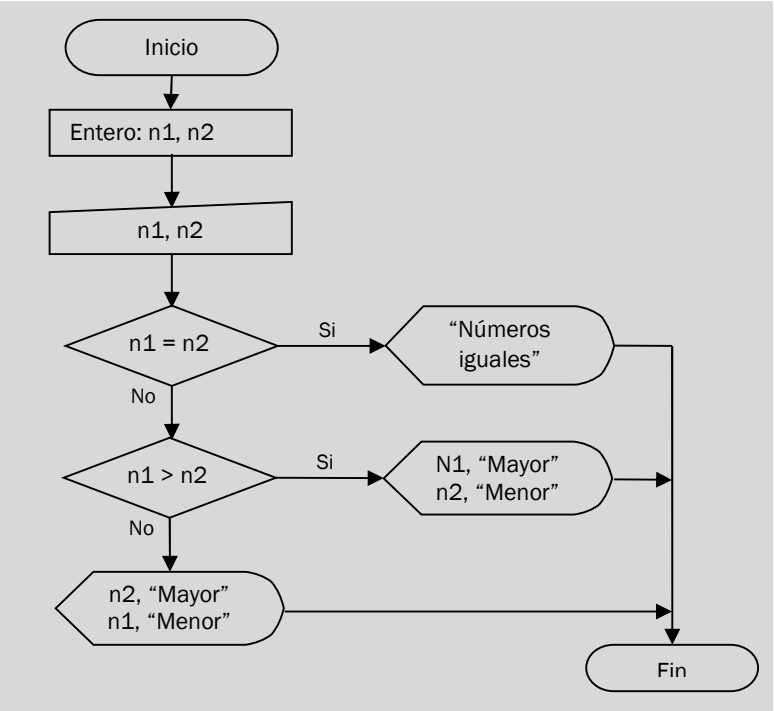
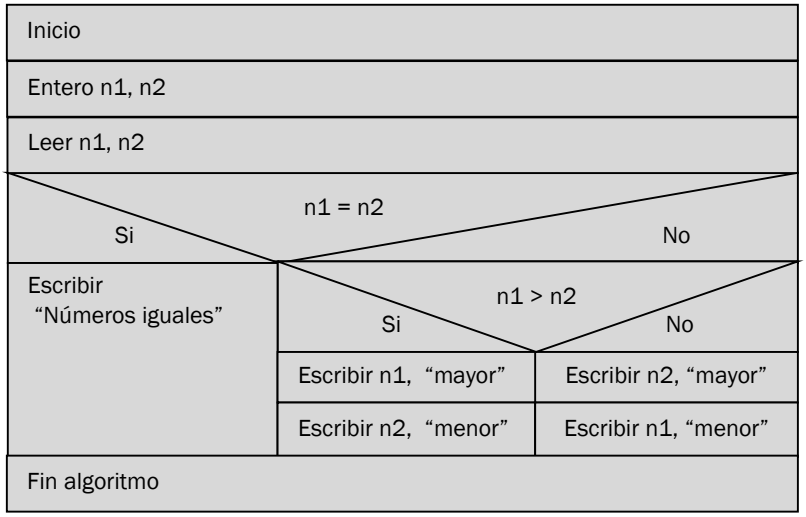


Figura 40. Diagrama N-S del algoritmo comparar números



Los resultados obtenidos en las pruebas del algoritmo se muestran en el cuadro 35.

Cuadro 35. Pseudocódigo del algoritmo comparar números

Ejecución	n1	n2	Salida
1	3	3	Números iguales
2	5	2	5 Mayor 2 Menor
3	6	8	8 Mayor 6 Menor

Ejemplo 14. Calcular aumento de sueldo

La empresa La Generosa S.A desea aumentar el sueldo a sus empleados, para ello ha establecido las siguientes condiciones: quienes ganan hasta $ 800.000 tendrán un incremento del 10%, quienes devengan más de $ 800.000 y hasta 1'200.000 recibirán un aumento del 8% y los demás del 5%. Se requiere un algoritmo que calcule el valor del aumento y el nuevo salario para cada empleado.

Para comprender el problema considérese los siguientes casos:

Un empleado devenga $ 700.000; dado que este valor es inferior a $ 800.000.oo tendrá un incremento del 10%, por tanto:

Aumento = 700.000 * 10 / 100 = 70.000
Nuevo sueldo = 700.000 + aumento
Nuevo sueldo = 770.000

Otro empleado devenga $ 1’000.000; este valor es superior a 800.000 e inferior a 1’200.000, por tanto tendrá un incremento del 8%, en consecuencia:

Aumento = 1’000.000 * 8 / 100 = 80.000
Nuevo sueldo = 1’000.000 + aumento
Nuevo sueldo = 1’080.000

Un tercer empleado tiene un sueldo de 1’500.000, como este valor es superior a 1’200.000 el porcentaje de aumento es del 5%, se tiene que:

Aumento = 1’500.000 * 5 / 100 = 75.000
Nuevo sueldo = 1’500.000 + aumento
Nuevo sueldo = 1’575.000

De esto se desprende que:

Entradas: sueldo (sue)
Salida: valor aumento (aum) y nuevo sueldo (nsue)
Cálculos: aumento = sueldo * porcentaje (por)
nuevo sueldo = sueldo + aumento

El diseño de la solución en notación pseudocódigo se presenta en el cuadro 36.

Cuadro 36. Pseudocódigo del algoritmo nuevo sueldo

```
1.  Inicio
2.      Real: sue, por, aum, nsue
3.      Leer sue
4.      Si sue <= 800000 entonces
5.          por = 10
6.      Si no
7.          Si sue <= 1200000 entonces
8.              por = 8
9.          Si no
10.             por = 5
11.         Fin si
12.     Fin si
13.     aum = sue * por /100
14.     nsue = sue + aum
15.     Escribir “Aumento:”, aum
16.     Escribir “Nuevo sueldo:”, nsue
17. Fin algoritmo
```

El diagrama de flujo se presenta en la figura 41 y en diagrama N-S en la figura 42.

Figura 41. Diagrama de flujo del algoritmo nuevo sueldo

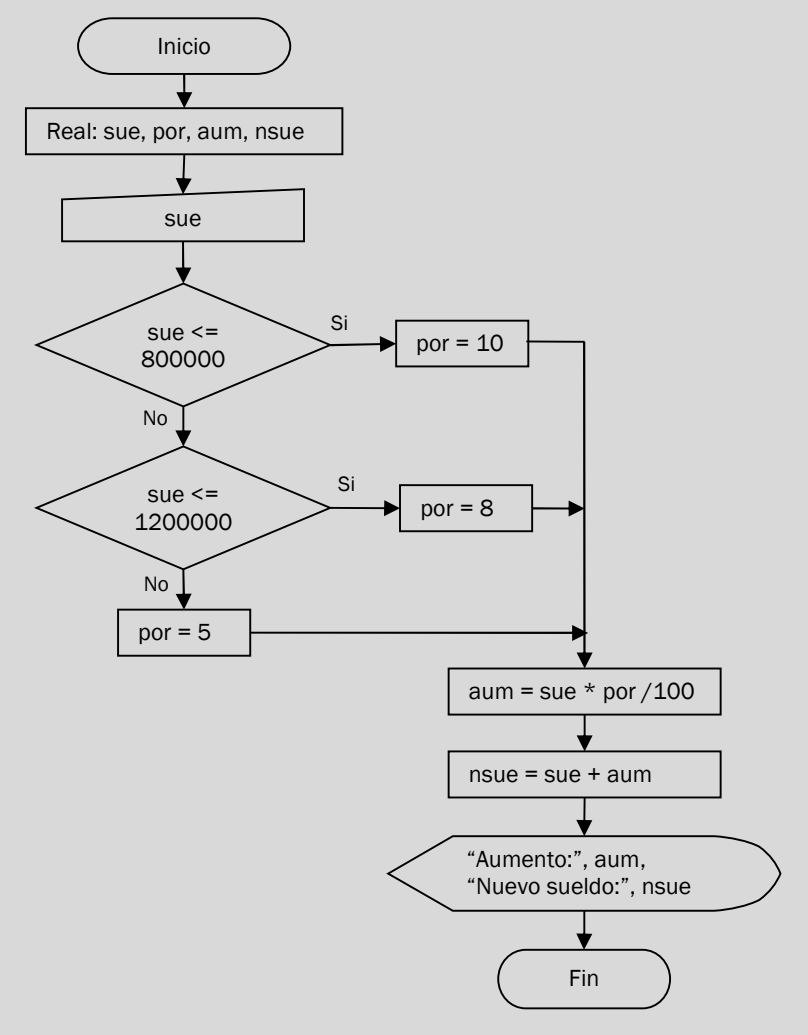
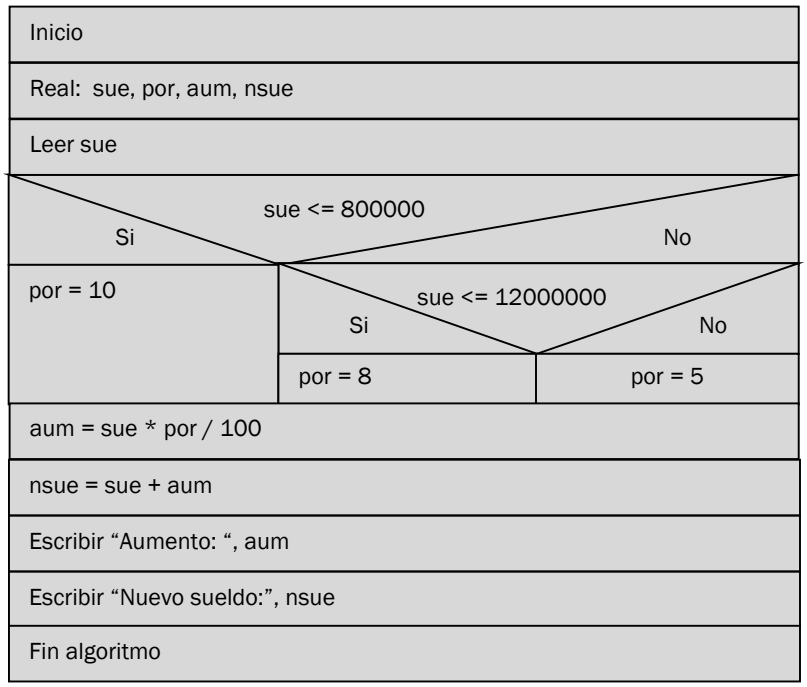


Figura 42. Diagrama N-S del algoritmo nuevo sueldo



Para verificar que el algoritmo funciona correctamente se prueba con los datos que se analizó anteriormente. Los resultados de esta prueba se muestran en el cuadro 37.

Cuadro 37. Verificación del algoritmo nuevo sueldo

Ejec.	sue	por	aum	nsue	Salida
1	700.000	10	70.000	770.000	Aumento: 70.000 Nuevo sueldo: 770.000
2	1'000.000	8	80.000	1'080.000	Aumento: 80.000 Nuevo sueldo: 1'080.000
3	1'500.000	5	75.000	1'575.000	Aumento: 75.000 Nuevo sueldo: 1'575.000

4.2.6 Más ejemplos de decisiones

Ejemplo 15. Selección de personal

Una empresa desea contratar un profesional para cubrir una vacante, los requisitos para ser considerado elegible son: ser profesional y tener entre 25 y 35 años inclusive, o contar con formación académica de especialista o superior en cuyo caso no se tiene en cuenta la

edad. Se requiere un algoritmo que evalúe los datos de cada candidato e informe si es apto o no apto.

Siguiendo la estrategia propuesta en este documento, se analizan casos particulares, se identifican los datos y las operaciones, para luego proceder a diseñar una solución genérica.

A la convocatoria se presenta Pedro, quien tiene formación de pregrado en administración de empresas y 28 años de edad. Al confrontar sus datos con las condiciones del problema se encuentra que su edad está en el rango establecido como válido y cuenta con formación de pregrado, por lo tanto es considerado apto.

Luego se presenta Lucía, quien es contador público y tienen una especialización en alta gerencia y 38 años. Al revisar las condiciones que debe cumplir se encuentra que la edad está fuera de rango, pero al contar con especialización es catalogada como apta.

Finalmente, se presenta Carlos, de 45 años y con formación de pregrado en economía. Al examinar el cumplimiento de la primera condición se encuentra que su edad está por encima del límite establecido, esta situación podría ser ignorada si contara con formación especializada, pero se verifica que su nivel de estudios es de pregrado; por tanto es declarado no apto.

Ahora, se trata de diseñar un algoritmo que tome esta decisión de forma automática para cualquier persona que desee saber si cumple los requisitos.

En este caso se requiere tres datos de entrada: el nombre, la edad y la formación académica. Para minimizar la posibilidad de error al ingresar el tercer dato, se presenta al usuario un menú del cual seleccione la opción que corresponda:

Formación académica

1. Tecnológico
2. Profesional
3. Especialista
4. Magister
5. Doctor

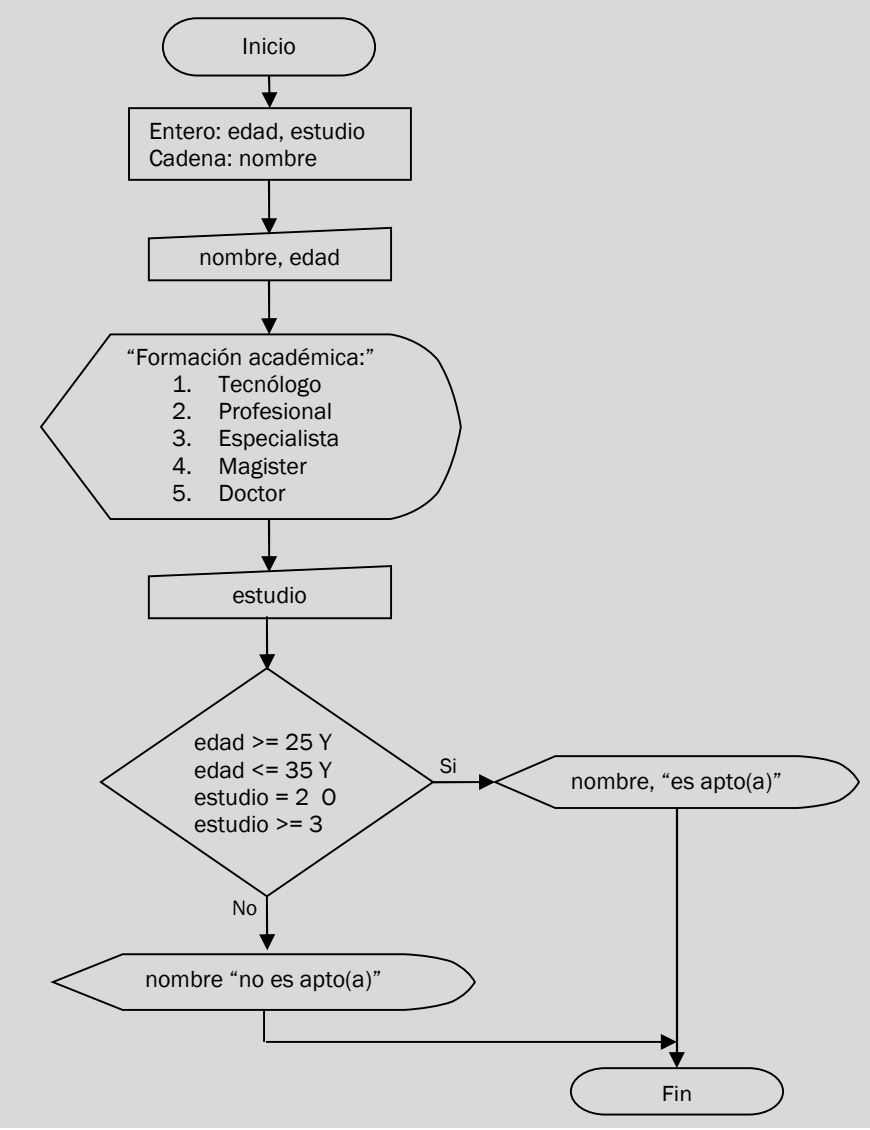
De esta manera la condición se establece sobre datos numéricos. El algoritmo para solucionar este problema se presente como diagrama de flujo en la figura 43.

Los resultados de la verificación del algoritmo se presentan en el cuadro 38.

Cuadro 38. Verificación del algoritmo selección de personal

Ejec.	Nombre	Edad	estudio	Salida
1	Pedro	28	2	Pedro es apto
2	Lucía	38	3	Lucía es apta
3	Carlos	45	2	Carlos no es apto

Figura 43. Diagrama de flujo para seleccionar un empleado



Ejemplo 16. Auxilio de transporte

Se requiere un algoritmo que decida si un empleado tiene derecho al auxilio de transporte. Se conoce que todos los empleados que devengan un salario menor o igual a dos salarios mínimos legales tienen derecho a este rubro.

Para identificar los datos del problema y la forma de solucionarlo se proponen dos ejemplos:

Supóngase un salario mínimo legal de $ 600.000.oo y José devenga un salario mensual de $750.000.oo. El planteamiento del problema estipula que tiene derecho al auxilio si su salario es menor o igual a dos veces el salario mínimo legal; entonces:

750.000.oo <= 2 * 600.000.oo
750.000.oo <= 1’200.000.oo

Esta expresión relacional es verdades; por tanto, José tiene derecho al auxilio de transporte.

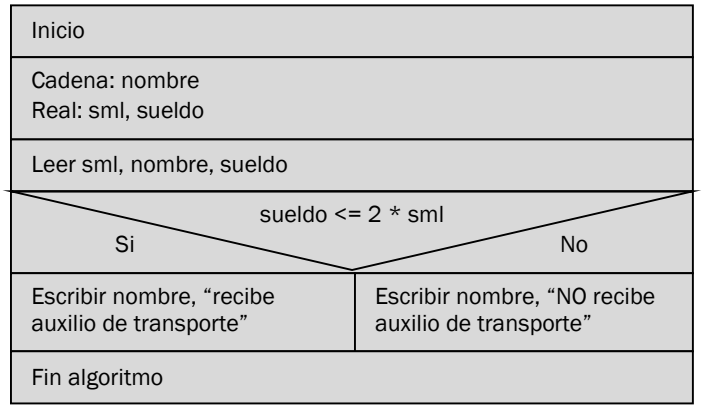
Luis devenga mensualmente $ 1’300.000.oo ¿tiene derecho al auxilio?

1’300.000.oo <= 2 * 600.000.oo
1’300.000.oo <= 1’200.000.oo

La condición no se cumple; en consecuencia, Luis no recibe auxilio de transporte.

El diseño de la solución a este problema se propone mediante el diagrama N-S de la figura 44.

Figura 44. Diagrama N-S del algoritmo auxilio de transporte



Los resultados de la verificación de este algoritmo, con los ejemplos propuestos se muestran en el cuadro 39.

Cuadro 39. Verificación del algoritmo auxilio de transporte

Ejec.	Sml	nombre	sueldo	sueldo <= 2 * sml	Salida
1	515.000	José	750.000	Verdadero	José recibe auxilio de transporte
2	515.000	Luis	1'200.000	Falso	Luis NO recibe auxilio de transporte

Ejemplo 17. Nota definitiva

En la universidad Buena Nota se requiere un algoritmo para calcular la nota definitiva y decidir si el estudiante aprueba o reprueba la asignatura. La nota final se obtiene a partir de dos notas parciales y un examen final, donde el primer parcial equivale al 30%, el segundo parcial al 30% y el examen final al 40%, y la nota mínima aprobatoria es 3.0.

Si el promedio de los dos parciales es menor a 2.0, el estudiante no puede presentar examen final y pierde la materia por bajo promedio, en este caso la nota definitiva es el promedio de los parciales, si el promedio es igual o superior a 2.0 puede presentar el examen final.

Si la nota del examen final es inferior a 2.0, se desconoce las notas parciales y la nota definitiva es la obtenida en el examen final. Si la nota es igual o superior a 2.0 se calcula la nota definitiva aplicando los porcentajes mencionados a los parciales y al final.

Si la nota definitiva es igual o superior a 3.0 el estudiante aprueba la asignatura; si es inferior a 3.0 pierde la materia; sin embargo, puede habilitarla, siempre y cuando en el examen final obtenga nota mayor o igual a 2.0, en este caso la nota definitiva será la que obtenga en la habilitación.

Para comprender mejor esta situación considérese estos casos:

Raúl obtuvo las siguientes notas:

Parcial 1 = 2.2
Parcial 2 = 1.6
Promedio = 1.9
Nota definitiva = 1.9

Siguiendo la información del problema Raúl no puede presentar examen final dado que el promedio de los dos parciales en menor a 2.0; en consecuencia la nota final es el promedio de los parciales y no puede presentar ni examen final ni habilitación. Pierde la asignatura.

Karol obtuvo las siguientes notas:

Parcial 1 = 3.4
Parcial 2 = 2.0
Promedio = 2.7

Puede presentar examen final

Examen final = 1.5

Dado que su nota de examen final es menor a 2.0, ésta pasa a ser la nota definitiva y no puede habilitar.

Las notas de Carlos fueron:

Parcial 1 = 3.5
Parcial 2 = 2.5
Promedio = 3.0

Puede presentar examen final

Examen final = 2.2

Nota definitiva = 3.5 * 30% + 2.5 * 30% + 2.2 * 40% = 2.7

Carlos no aprueba la materia, pero como tiene nota mayor a 2.0 en el examen final puede habilitar

Habilitación = 3.5
Nota definitiva = 3.5. Aprueba la materia

Ana, por su parte, obtuvo las siguientes notas:

Parcial 1 = 3.7
Parcial 2 = 4.3

Promedio = 4.0

Presenta examen final

Examen final = 3.5

Nota definitiva = 3.8

Por lo tanto, Ana aprueba la materia.

Ahora, que se comprende bien el problema ya se puede diseñar un algoritmo para solucionar cualquier caso. Este algoritmo se presenta en el cuadro 40.

Cuadro 40. Pseudocódigo algoritmo Nota Definitiva

```
1.  Inicio
2.      Real: p1, p2, ef, nd, prom, nh
3.      Leer p1, p2
4.      prom = (p1 + p2)/2
5.      Si prom < 2.0 entonces
6.          nd = prom
7.          Escribir “pierde materia por bajo promedio”
8.      si no
9.          Leer ef
10.         Si ef < 2.0 entonces
11.             nd = ef
12.             Escribir “Pierde materia y no puede habilitar”
13.         si no
14.             nd = p1 * 0.3 + p2 * 0.3 + ef * 0.4
15.             Si nd >= 3.0 entonces
16.                 Escribir “Ganó la materia”
17.             si no
18.                 Escribir “Perdió la materia pero puede habilitar”
19.                 Leer nh
20.                 nd = nh
21.                 Si nh >= 3.0 entonces
22.                     Escribir “Ganó la habilitación”
23.                 si no
24.                     Escribir “Perdió la habilitación”
25.                 Fin si
26.             Fin si
27.         Fin si
28.     Fin si
29.     Escribir “Nota definitiva:”, nd
30. Fin algoritmo
```

En este algoritmo se declaran variables para almacenar las dos notas parciales, el promedio, el examen final, la nota definitiva y la nota de habilitación. Para comenzar se leen

los dos parciales, se calcula el promedio y se evalúa (línea 5), si éste es menor a 2.0, se asigna a la nota definitiva, se muestra un mensaje, la nota y el algoritmo termina.

Si el promedio es superior a 2.0 se lee el examen final y se evalúa (línea 10) si es menor a 2.0 se asigna como definitiva, se muestra un mensaje y salta al final del algoritmo. Si el examen final es superior a 2.0 se hace el cálculo de la nota definitiva y se decide si aprueba o pierde, si pierde puede presentar habilitación, esta es la última oportunidad de aprobar la asignatura.

En el cuadro 41 se muestra tres conjuntos de datos con los que se prueba el algoritmo y los resultados obtenidos.

Cuadro 41. Verificación del algoritmo Nota Definitiva

Ejec	P1	P2	prom	ef	nd	nh	Salida
1	2.2	1.6	1.9	-	1.9		Perdió la materia por bajo promedio Nota definitiva: 1.9
2	3.4	2.0	2.7	1.5	1.5		Pierde materia Nota definitiva: 1.5
3	3.5	2.5	3.0	2.2	2.7	3.5	Perdió la materia pero puede habilitar Ganó la habilitación Nota definitiva: 3.5
4	3.7	4.3	4.0	3.5	3.8		Ganó la materia Nota definitiva: 3.8

Ejemplo 18. El hermano mayor

Este algoritmo lee los nombres y las edades de tres hermanos y decide cuál es el nombre del hermano mayor.

Supóngase los siguientes casos:

José, Luis y María son hermanos, el primero tiene 19 años, el segundo 15 y la tercera 23. ¿Cuál de ellos es el mayor? Evidentemente, María.

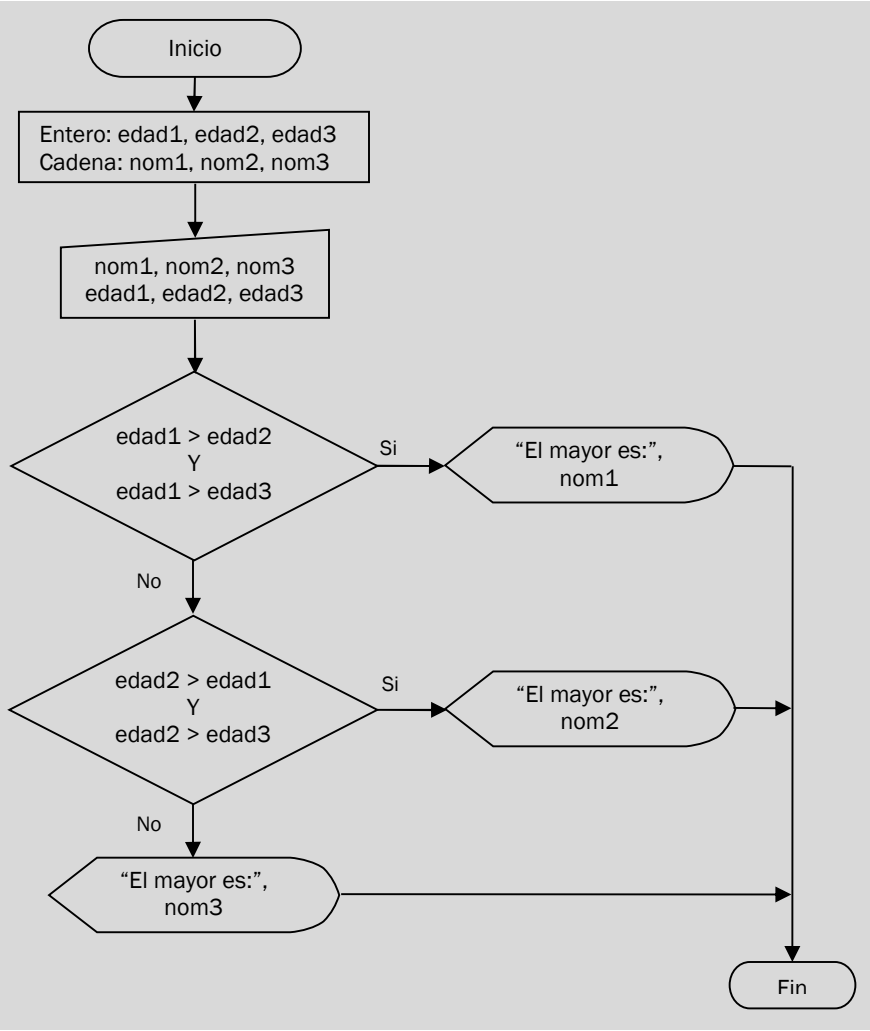
Carlos tiene 32, rocío tiene 28 y Jesús tiene 25. En este caso, el mayor es Carlos.

Martha tiene 8 años, Ana 10 y Camilo 4. De los tres, Ana es mayor.

Ahora se trata de diseñar un algoritmo para que ésta decisión la tome un programa de computador. Es necesario declarar tres variables de tipo cadena y relacionarlas con tres de tipo numérico, así se comparan las edades y se muestra el nombre como resultado. El diagrama de flujo de este algoritmo se presenta en la figura 45.

Para verificar que el algoritmo es correcto se revisa paso a paso registrando los valores correspondientes a las variables y la salida por pantalla. En el cuadro 42 se muestran los datos de prueba y los resultados obtenidos.

Figura 45. Diagrama de flujo del algoritmo hermano mayor



Cuadro 42. Verificación algoritmo del hermano mayor

Ejec.	nom1	edad1	nom2	edad2	nom3	edad3	Salida
1	José	19	Luis	15	María	23	María
2	Carlos	32	Rocío	28	Jesús	25	Carlos
3	Martha	8	Ana	10	Camilo	4	Ana

Ejemplo 19. Descuento a mayoristas

El almacén Gran Distribuidor vende camisas al por mayor y hace descuentos según la cantidad facturada: en cantidades superiores o iguales a 1000 unidades hace el 10% de descuento; entre 500 y 999, el 8%; entre 200 y 499, el 5%; y en menos de 200 no hay descuento. Dada la cantidad facturada y el precio unitario, se requiere calcular el descuento que se le hace a un cliente y el valor a pagar.

Antes de proceder con el diseño del algoritmo es conveniente comprender bien el problema a partir de un caso particular para realizar los cálculos.

Si una camisa tiene un costo de $ 80.000 y un cliente hace un pedido de 350 unidades, entonces tiene derecho a un descuento del 5%:

Unidades: 350
Valor unitario: 80.000
Valor total = 350 * 80.000 = 28.000.000
Porcentaje descuento: 5%
Valor descuento = 28.000.000 * 5 / 100 = 1.400.000
Valor a pagar = 28.000.000 – 1.400.000 = 26.600.000

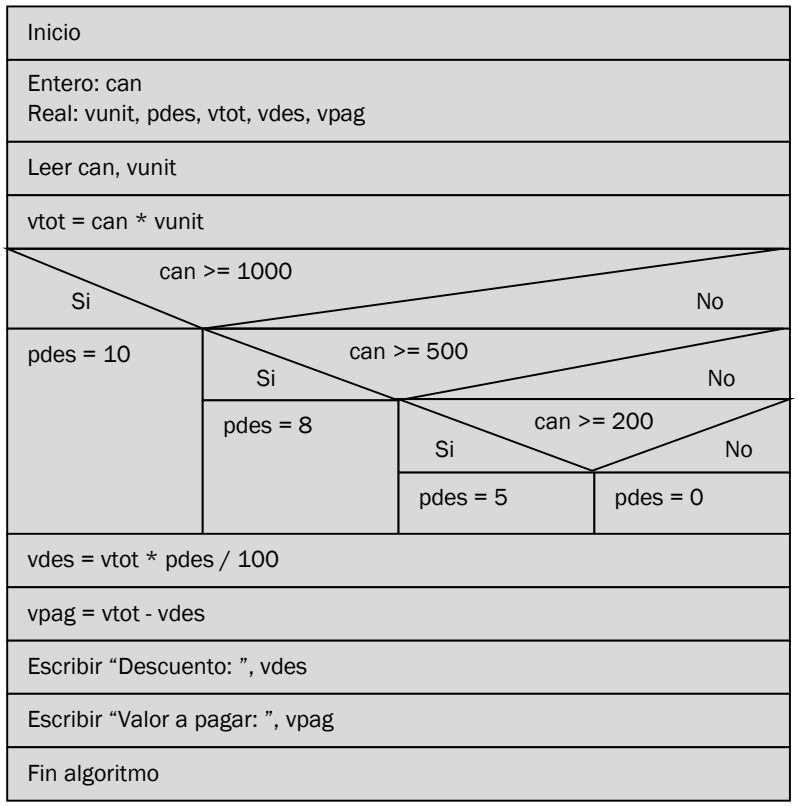
En este caso el cliente tendrá un descuento por valor de $ 1.400.000 y pagará en total $ 26.600.000.

Ahora se puede identificar los datos de entrada, de salida y los procesos a realizar:
Entrada: cantidad, valor unitario
Salida: valor descuento, valor a pagar
Procesos:

- Total = cantidad * valor unitario
- Valor descuento = total * porcentaje descuento / 100
- Valor a pagar = total - descuento

La solución se presenta mediante diagrama N-S en la figura 46.

Figura 46. Diagrama N-S algoritmo descuento a mayoristas



En el cuadro 43 se muestra los datos de prueba de este algoritmo.

Cuadro 43. Verificación algoritmo descuento a mayorista

Ejec.	can	vunit	Vtot	pdes	vdes	Vpag	Salida
1	350	80000	28000000	5	1400000	26600000	Descuento 1400000 Pagar 26600000
2	600	20000	12000000	8	960000	11040000	Descuento 960000 Pagar 11040000
3	1100	30000	33000000	10	3300000	29700000	Descuento 3300000 Pagar 29700000

Ejemplo 20. Depósito a término

El banco Buena Paga ofrece diferentes tasas de interés anual para depósitos a término dependiendo del tiempo por el que se hagan. Si el depósito es por un periodo menor o igual a seis meses la tasa es del 8% anual; entre siete y 12 meses, 10%; entre 13 y 18, 12%; entre 19 y 24, 15%; y para periodos superiores a 24 meses el 18%. Se requiere un algoritmo para determinar cuánto recibirá un cliente por un depósito, tanto por concepto de interés como en total.

Considérese algunos casos particulares:

Caso 1: Pedro Pérez hace un depósito de un millón de pesos ($ 1’000.000.oo) durante cinco meses. Según la información que se da en el planteamiento del problema, el bando le pagará una tasa de interés del 8% anual.

Dado que la tasa de interés está expresada en años y el depósito en meses, lo primero que hay que hacer es pasarla a meses, para ello se divide sobre 12.

Tasa interés mensual = 8 / 12
Tasa interés mensual = 0.667

Ahora, se tienen los siguientes datos:

Capital = 1000000.oo
Tiempo = 5
Tasa interés mensual = 0.667 %

Para conocer cuánto recibirá por concepto de interés se aplica al valor del capital el porcentaje que se acaba de obtener y se multiplica por el número de periodos, en este caso cinco meses, de la forma:

Interés = 1000000.oo * (0.667 / 100) * 5
Interés = 33350

Y para obtener el valor futuro de la inversión, se suma el interés al capital:

Valor futuro = 1000000 + 33350
Valor futuro = 1033350

Caso 2: José López desea depositar cinco millones por un periodo de un año y medio (18 meses). En este caso, el banco le ofrece un interés de 12% anual.

La tasa de interés convertida a meses es de:

Tasa interés mensual = 12 / 12 = 1

Entonces, los datos son:

Capital = 5000000
Tasa interés mensual = 1 %
Tiempo = 18

De dónde se obtiene:
Interés = 5000000 * (1 / 100) * 18
Interés = 900000

Valor futuro = capital + interés
Valor futuro = 5000000 + 900000
Valor futuro = 5900000

Con base en los dos ejemplos se concluye que:

Datos de entrada: capital, tiempo
Datos de salida: interés, valor futuro
Procesos:

- Determinar la tasa de interés anual (es una decisión)
- Interés mensual = tasa interés anual / 12
- Interés = capital * tasa interés mensual / 100 * tiempo
- Valor futuro = capital + interés

El algoritmo de solución a este ejercicio se presenta, en notación de pseudocódigo, en el cuadro 44.

En este algoritmo se utiliza las estructuras de decisión *si anidadas* para determinar el porcentaje correspondiente a la tasa de interés anual, dependiendo del tiempo. Se aplica decisiones anidadas y no la estructura *según sea*, ya que ésta sólo puede utilizarse para evaluar la condición de igualdad entre el contenido de la variable y un valor constante, pero no para relaciones de mayor o menor.

En el cuadro 45 se muestra los resultados de verificar el algoritmo con los datos utilizados en los dos ejemplos desarrollados al comienzo de este ejercicio, más un tercero por valor de 8000000 a 36 meses.

Cuadro 44. Pseudocódigo algoritmo depósito a término

```
1.  Inicio
2.      Real: capital, tasaintanual, tasaintmes, interés, valfuturo
3.      Entero: tiempo
4.      Leer capital, tiempo
5.      Si tiempo < 1 entonces
6.          Escribir “Tiempo no valido”
7.          tasaintanual = 0
8.      Si no
9.          Si tiempo <= 6 entonces
10.             tasaintanual = 8
11.         Si no
12.             Si tiempo <= 12 entonces
13.                 tasaintanual = 10
14.             Si no
15.                 Si tiempo <= 18 entonces
16.                     tasaintanual = 12
17.                 Si no
18.                     Si tiempo <= 24 entonces
19.                         tasaintanual = 15
20.                     Si no
21.                         tasaintanual = 18
22.                     Fin si
23.                 Fin si
24.             Fin si
25.         Fin si
26.     Fin si
27.     tasaintmes = tasaintanual / 12
28.     interés = capital * tasaintmes / 100 * tiempo
29.     valfuturo = capital + interés
30.     Escribir “Interes = ”, interés
31.     Escribir “Valor futuro = ”, valfuturo
32. Fin algoritmo
```

Cuadro 45. Verificación del algoritmo depósito a término

Capital	Tiempo	tasaintanual	tasaintmes	interés	Valfuturo
1000000	5	8	0.667	33350	1033350
5000000	18	12	1	900000	5900000
8000000	36	18	1.5	4320000	12320000

Ejemplo 21. Empresa editorial

Una empresa editorial tiene tres grupos de empleados: vendedores, diseñadores y administrativos. Se requiere calcular el nuevo sueldo para los empleados teniendo en cuenta que se incrementarán así: administrativos 5%, diagramadores 10% y vendedores 12%.

Como un caso particular se calcula el nuevo sueldo de la secretaria, quien devenga actualmente un sueldo de $ 600.000.oo. Su cargo es de tipo administrativo, por tanto le corresponde un incremento del 5%.

Cargo: secretaria
Tipo de cargo: administrativo
Sueldo actual: 600.000.oo
Porcentaje incremento: 5%

Por lo tanto

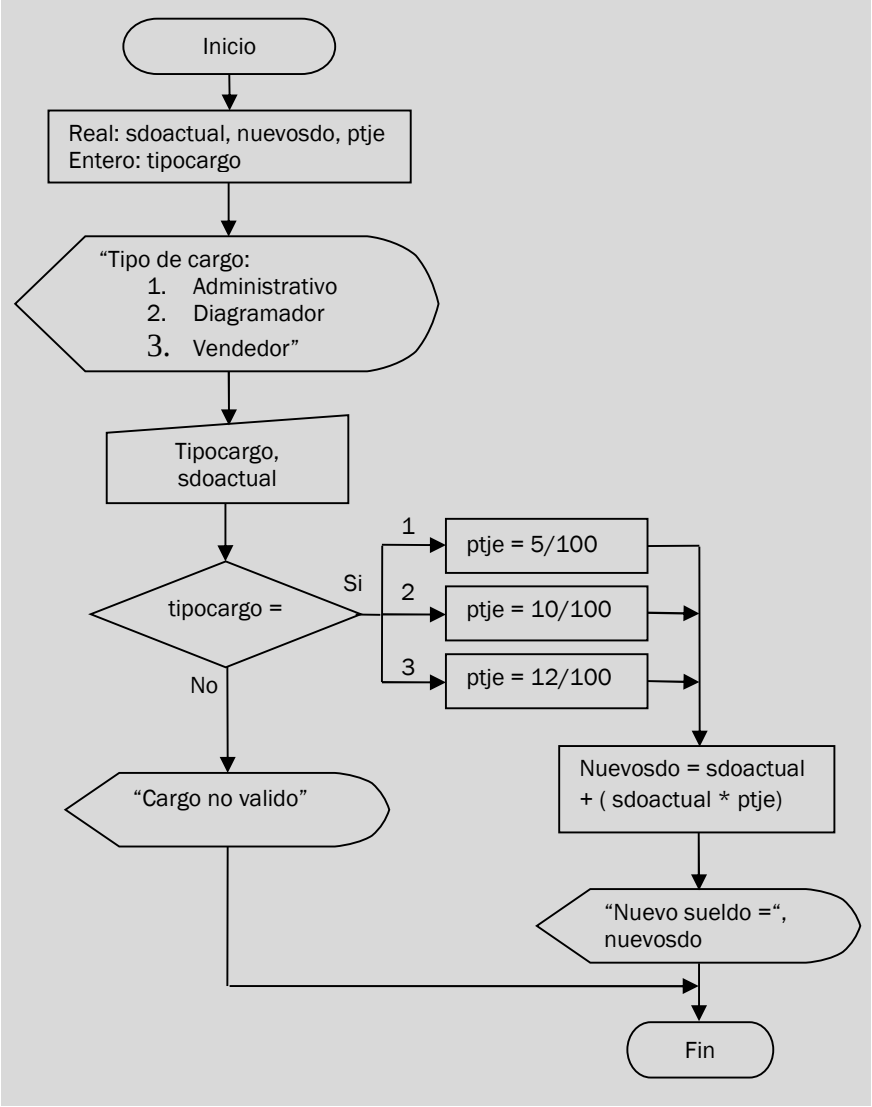
Valor incremento: 30.000.oo
Nuevo sueldo: 630.000.oo

En este ejercicio se tiene como dato de entrada el sueldo actual y el tipo de cargo que desempeña (administrativo, diseñador, vendedor), ya que el porcentaje de incremento depende del cargo y se aplica sobre el valor del sueldo vigente. El algoritmo para solucionar este problema se presenta en notación de diagrama de flujo en la figura 47.

En este algoritmo se declaran tres variables de tipo real para sueldo actual, nuevo sueldo y porcentaje de incremento, y una variable entera para capturar la opción correspondiente al tipo de cargo que tiene el empleado. Se toma una decisión (múltiple) sobre el tipo de cargo y se establece el porcentaje a aplicar, se realiza el cálculo y se muestra el nuevo sueldo. Si la opción seleccionada como cargo no está entre 1 y 3 se muestra un mensaje de error.

Se proponen tres conjuntos de datos para verificar si el algoritmo es correcto. Los resultados de las ejecuciones se presentan en el cuadro 46.

Figura 47. Diagrama de flujo del algoritmo Empresa editorial



Cuadro 46. Verificación del algoritmo Empresa editorial

Ejec.	tipocargo	sdoactual	Ptje	nuevosdo	Salida
1	1	600.000	0.05	630.000	Nuevo sueldo = 630.000
2	2	800.000	0.1	880.000	Nuevo sueldo = 880.000
3	3	700.000	0.12	784.000	Nuevo sueldo = 784.000
4	4	500.000			Cargo no válido

Ejemplo 22. Descuento en motocicleta

La distribuidora de motocicletas Rueda Floja ofrece una promoción que consiste en lo siguiente: las motos marca Honda tienen un descuento del 5%, las de marca Yamaha del 8% y las Suzuki el 10%, las de otras marcas el 2%. Se requiere calcular el valor a pagar por una motocicleta.

Considérense algunos ejemplos particulares de este problema.

Pedro desea comprar una motocicleta Honda cuyo precio es de seis millones de pesos. ¿Cuánto debe pagar en realidad por aquella moto?

En la descripción del problema se dice que este tipo de motos tienen un descuento del 5%, entonces hay que calcular el valor del descuento que se le hará a la moto que desea Pedro; se tienen los siguientes datos:

Valor moto = 6’000.000.oo
Porcentaje descuento = 5/100 (5%)
Valor descuento = 6’000.000.oo * 5 / 100 = 300.000.oo
Valor neto = 6’000.000.oo – 300.000.oo = 5’700.000.oo

Dado que la motocicleta que Pedro desea tiene un descuento del 5% el valor neto a pagar es de $ 5’700.000.oo.

Juan quiere comprar una motocicleta Yamaha que tiene un precio de lista de $ 8’500.000.oo. Dado que esta marca tiene un descuento de 8%, se tiene que:

Valor moto = 8’500.000.oo
Porcentaje descuento = 8/100 (8%)
Valor descuento = 8’500.000.oo * 8 / 100 = 680.000.oo
Valor neto = 8’500.000.oo – 680.000.oo = 7’820.000.oo

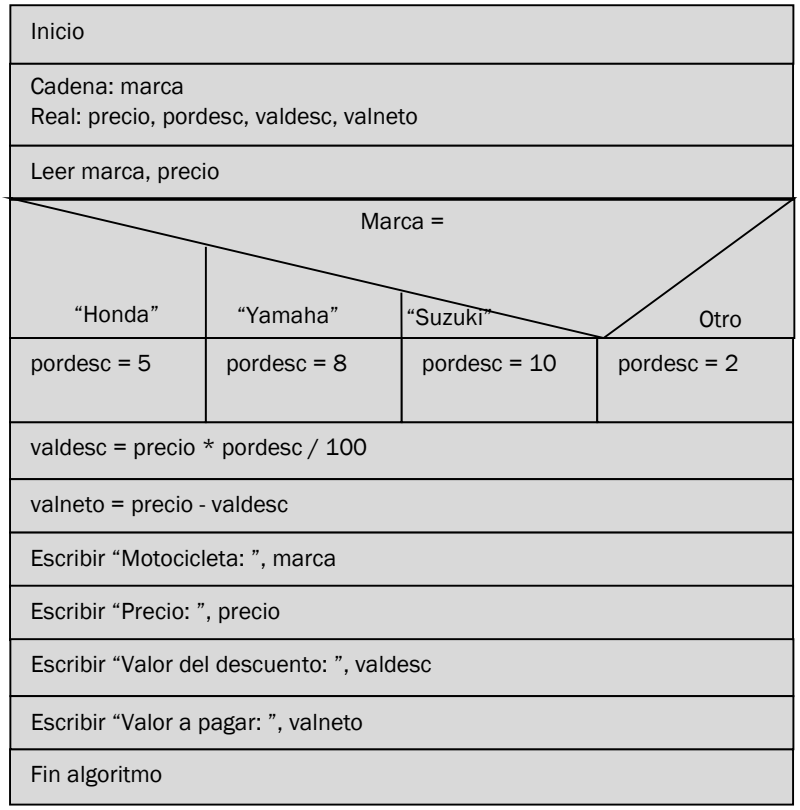
Por lo tanto, Juan debe pagar la suma de $ 7’820.000.oo por la motocicleta Yamaha.

Con base en los ejemplos anteriores se puede generalizar el procedimiento, así:

Porcentaje de descuento = depende de la marca (5%, 8%, 10% o 2%)
Valor descuento = valor moto * porcentaje descuento
Valor neto = valor moto – valor descuento

En consecuencia los datos a introducir corresponden a: precio de lista y la marca de la motocicleta. El algoritmo que soluciona este problema se presenta en notación N-S en la figura 48.

Figura 48. Descuento en motocicleta



Para verificar que el algoritmo es correcto se realiza la prueba de escritorio considerando cada una de las alternativas de decisión, como se muestra en el cuadro 47.

Ejemplo 23. Comisión de ventas

La empresa V&V paga a sus vendedores un salario básico más una comisión sobre las ventas efectuadas en el mes, siempre que éstas sean mayores a $ 1.000.000. Los porcentajes de bonificación son:

Ventas mayores a $ 1.000.000 y hasta $ 2.000.000 = 3%
Ventas mayores a $ 2.000.000 y hasta $ 5.000.000 = 5%
Ventas mayores a $ 5.000.000 = 8%

Se desea conocer cuánto corresponde al vendedor por comisión y cuánto recibirá en total en el mes.

Siguiendo el enfoque que se ha propuesto para este libro, lo primero es considerar algunos ejemplos que permitan comprender el problema y la forma de solucionarlo. En este orden de ideas se plantean cuatro ejemplos:

Cuadro 47. Verificación del algoritmo descuento en motocicleta

Ejec.	Marca	precio	pordesc	valdesc	valneto	Salida
1	Honda	6'000.000	5	300.000	5'700.000	Motocicleta: Honda Precio: 6'000.000 Valor del descuento: 300.000 Valor a pagar: 5'700.000
2	Yamaha	8'500.000	8	680.000	7'820.000	Motocicleta: Yamaha Precio: 8'500.000 Valor del descuento: 680.000 Valor a pagar: 7'820.000
3	Suzuki	3'800.000	10	380.000	3'420.000	Motocicleta: Suzuki Precio: 3'800.000 Valor del descuento: 380.000 Valor a pagar: 3'420.000
4	Kawasaki	5'000.000	2	100.000	4'900.000	Motocicleta: Kawasaki Precio: 5'000.000 Valor del descuento: 100.000 Valor a pagar: 4'900.000

Jaime fue contratado con un salario básico de $ 600.000.oo y durante el último mes vendió la suma de $ 1'200.000.oo. Dado que vendió más de un millón tiene derecho a comisión; entonces, el siguiente paso es decidir qué porcentaje le corresponde. En la descripción del ejercicio se establece que si las ventas están entre uno y dos millones el vendedor recibirá una comisión del 3% sobre el valor de las ventas. Esto es:

Vendedor(a): Jaime
Salario básico: 600.000.oo
Valor ventas: 1'200.000.oo
Porcentaje comisión: 3%
Valor comisión = 1'200.000 * 3 / 100 = 36.000
Sueldo del mes = 600.000 + 36.000 = 636.000

Jaime recibirá $ 36.000 por concepto de comisión y el sueldo total será de $ 636.000.oo

Susana tiene un salario básico de $ 700.000.oo y en el último mes sus ventas fueron por valor de $ 3'500.000.oo. Este caso, confrontando los requerimientos del problema, ella tiene derecho a una comisión del 5%. Los resultados del ejercicio para Susana son:

Vendedor(a): Susana
Salario básico: 700.000.oo
Valor ventas: 3'500.000.oo
Porcentaje comisión: 5%
Valor comisión = 3'500.000 * 5 / 100 = 175.000
Sueldo del mes = 700.000 + 175.000 = 875.000

Susana ha conseguido una comisión de ventas por valor de $ 175.000 para un sueldo total de $ 875.000.oo.

Carlos tiene un sueldo básico de $ 600.000.oo y en el último mes sus ventas fueron de $ 950.000.oo. Como no alcanzó a vender el millón no recibirá comisión.

Vendedor(a): Carlos
Salario básico: 600.000.oo
Valor ventas: 950.000.oo
Porcentaje comisión: Ninguno
Valor comisión = Ninguna
Sueldo del mes = 600.000

Carlos recibe únicamente el salario básico.

Rocío es conocida como la mejor vendedora de la empresa, tiene asignado un salario básico de $ 800.000.oo. En el último mes logró vender siete millones de pesos.

Vendedor(a): Rocío
Salario básico: 800.000.oo
Valor ventas: 7’000.000.oo
Porcentaje comisión: 8%
Valor comisión = 7’000.000 * 8 / 100 = 560.000
Sueldo del mes = 800.000 + 560.000 = 1’360.000

Rocío obtiene una comisión de $ 560.000.oo y un sueldo total de $ 1’360.000.oo

De los anteriores ejemplos se identifica los datos de entrada y los cálculos a realizar, los datos de salida los especifica el planteamiento del ejercicio: vendedor, comisión y sueldo total. La solución algorítmica se presenta en el cuadro 48 en la notación de pseudocódigo.

Los datos a ingresar son:

Nombre del vendedor
Sueldo básico
Valor de ventas
Los cálculos a realizar son:
Valor comisión = valor ventas * porcentaje comisión (3, 5, 8) / 100
Sueldo mes = sueldo básico + valor comisión

Para verificar si el algoritmo funciona correctamente se ejecuta el pseudocódigo, paso a paso, con los datos de los ejemplos explicados previamente para los cuales ya se hicieron los cálculos y se conoce los resultados. Los resultados se presentan en el cuadro 49.

Cuadro 48. Algoritmo comisión de ventas

```
1.  Inicio
2.      Cadena: nom
3.      Real: sbasico, venta, pcomi, vcomi, smes
4.      Leer nom,sbasico,venta
5.      Si venta > 1000000 entonces
6.          Si venta <= 2000000 entonces
7.              pcomi = 3
8.          Si no
9.              Si venta <= 5000000 entonces
10                 pcomi = 5
11             Si no
12                 pcomi = 8
13             Fin si
14         Fin si
15     Si no
16         pcomi = 0
17     Fin si
18     vcomi = venta * pcomi / 100
19     smes = sbasico + vcomi
20     Escribir "Vendedor(a):", nom
21     Escribir "Comisión:", vcomi
22     Escribir "Sueldo total:", smes
23  Fin algoritmo
```

Cuadro 49. Verificación del algoritmo comisión de ventas

Nom	sbasico	Venta	pcomi	vcomi	smes	Salida
Jaime	600000	1200000	3	36000	636000	Vendedor(a): Jaime Comisión: 36000 Sueldo total: 636000
Susana	700000	3500000	5	175000	875000	Vendedor(a): Susana Comisión: 175000 Sueldo total: 875000
Carlos	600000	950000	0	0	600000	Vendedor(a): Carlos Comisión: 0 Sueldo total: 600000
Rocío	800000	7000000	8	560000	1360000	Vendedor(a): Rocío Comisión: 560000 Sueldo total: 1360000

Ejemplo 24. Factura del servicio de energía

Se requiere un algoritmo para facturar el servicio de energía eléctrica. El consumo del mes se determina por diferencia de lecturas. El valor del kilovatio (kW[**]) es el mismo para todos los usuarios, pero se hace un descuento para algunos estratos, así:

Estrato 1: 10%
Estrato 2: 6%
Estrato 3: 5%

Para todos los estratos se aplica un descuento del 2% si el consumo es superior a 200 kW. Se desea que el programa muestre el consumo y el valor a pagar por éste servicio. (El planteamiento de este ejercicio es meramente didáctico y no tiene relación alguna con la facturación de este servicio en el mundo real).

En una empresa real se mantiene los registros de las lecturas y por ende sólo es necesario hacer una lectura y se hace la diferencia con la última que se tenga almacenada, pero en este ejercicio es necesario leer los dos datos para calcular la diferencia. Para comprender mejor este planteamiento obsérvese los siguientes ejemplos:

José es un usuario del servicio de energía que reside en estrato 1, el mes anterior su medidor registró 12345 y en este mes, 12480. Dado que está en estrato 1 recibirá un descuento del 10%, y si el valor del kW es de $ 500 el consumo y el valor a pagar se obtienen de la siguiente forma:

Usuario: José
Lectura 1: 12345
Lectura 2: 12480
Consumo: 12480 – 12345 = 135
Valor kW: 500
Valor consumo: 67500
Estrato: 1
Porcentaje de descuento: 10%
Valor del descuento = 67500 * 10 / 100 = 6750
Valor a pagar = 67500 – 6750 = 60750

De este ejemplo se puede identificar los datos y los procesos necesarios para solucionar el ejercicio. Los datos de entrada son:

Usuario,
Estrato socioeconómico,
Lectura 1,

[**] kW es el símbolo de kilovatio. Un kilovatio es una unidad de medida de electricidad equivalente a 1000 vatios. Equivale, también, a la energía producida o consumida por una potencia de un kW durante una hora. (Círculo Enciclopedia Universal)

Lectura 2 y
valor del kW.

El porcentaje a aplicar como descuento es una decisión basada en el estrato socio-económico del usuario. Los valores a calcular son:

Consumo = lectura 2 – lectura 1
Valor consumo = consumo * valor kW
Valor descuento = valor consumo * porcentaje descuento / 100
Valor a pagar = valor consumo – valor descuento

El paso siguiente es organizar las acciones en forma algorítmica. El diagrama de flujo de la solución a este ejercicio se presenta en la figura 49.

Para verificar que la solución es correcta se ejecuta paso a paso, se introduce los datos que se utilizaron para analizar el problema y se confrontan los resultados generados por el algoritmo con los obtenidos manualmente.

Ejemplo 25. Calculadora

Este algoritmo lee dos números y un operador aritmético, aplica la operación correspondiente y muestra el nombre de la operación y el resultado.

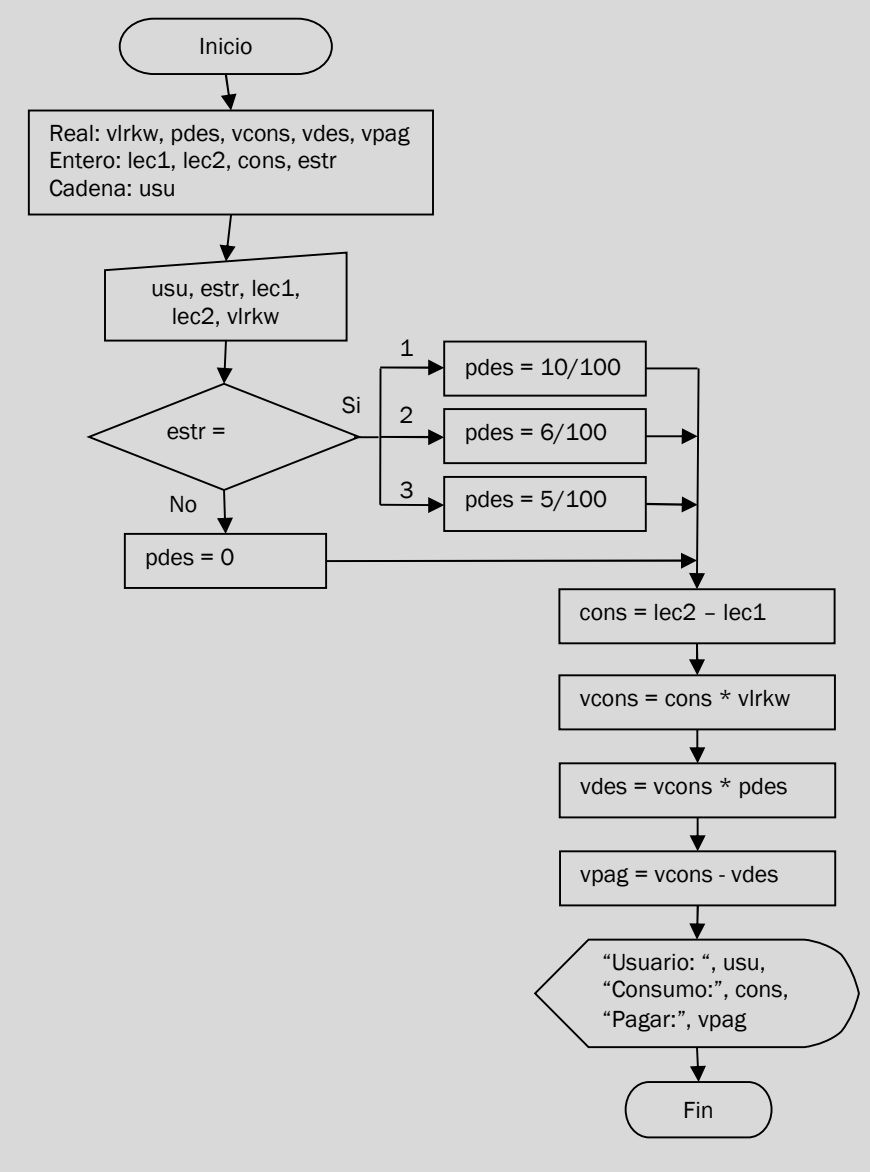
En este caso se trata de identificar el operador que el usuario ha introducido y realizar la operación correspondiente. Los operadores aritméticos a tener en cuenta son:

+ suma
- resta
* multiplicación
/ división

Por ejemplo:

Primer número: 3
Segundo número: 4
Operador: +

Figura 49. Facturación servicio de energía



El algoritmo debe realizar la operación suma y mostrar el resultado, así:

Suma: 3 + 4 = 7

Otro ejemplo:

Primer número: 6
Segundo número: 2
Operador: -

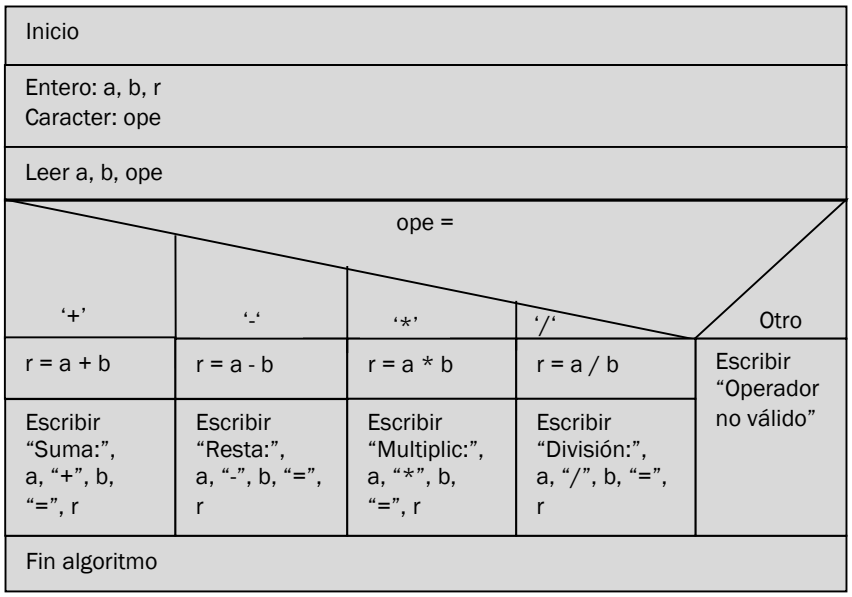
El resultado será:

Resta: 6 – 2 = 4

El algoritmo para solucionar este ejercicio requiere una decisión múltiple aplicada al operador y en cada alternativa de decisión se realiza la operación correspondiente y se muestra el resultado. El diagrama N-S de este algoritmo se presente en la figura 50.

Se declaran las variables: a, b, y r para primer número, segundo número y resultado, y ope para el operador. Al hacer la verificación del algoritmo, probando todos los operadores, se obtiene los resultados que se muestran en el cuadro 50.

Figura 50. Diagrama N-S del algoritmo Calculadora



Cuadro 50. Verificación del algoritmo calculadora

Ejec	a	b	ope	r	Salida
1	3	4	+	7	Suma: 3 + 4 = 7
2	6	2	-	4	Resta: 6 – 2 = 4
3	2	5	*	10	Multiplic: 2 * 5 = 10
4	27	9	/	3	División: 27 / 9 = 3
5	8	7	X		Operador no válido

4.2.7 Ejercicios propuestos

Diseñar algoritmos para solucionar los siguientes problemas y representarlos mediante pseudocódigo, diagramas de flujo y diagramas N-S.

1. Para integrar la selección de baloncesto, además de ser un buen jugador, es necesario una estatura de 1.70 m como mínimo. Dada la estatura de un aspirante decidir si es apto.

2. Dados dos enteros, se desea saber si alguno de ellos es múltiplo del otro.

3. Dado dos números racionales a/b y d/e se desea saber si son equivalentes y si no los son, cuál es el mayor y cuál el menor.

4. Una empresa está pagando una bonificación a sus empleados dependiendo del tiempo de servicio y del estado civil, así: para empleados solteros: si llevan hasta cinco años, el 2% del sueldo; entre 6 y 10 años el 5%; más de 10 años el 10%. Para empleados casados: si llevan hasta 5 años, el 5% del sueldo; entre 6 y 10 años el 10%, más de 10 años el 15%.

5. En un almacén se hace un descuento del 8% a los clientes cuya compra sea superior a un millón y del 5% si es superior $500.000 y menor o igual a un millón ¿Cuánto pagará una persona por su compra?

6. Se sabe que un año es bisiesto cuando es divisible para cuatro y no para 100 o cuando es divisible para cuatrocientos. Se desea determinar si un año *n* es bisiesto.

7. Se requiere un algoritmo que lea tres números y los ordene de forma ascendente.

8. Se tiene un recipiente cilíndrico de radio *r* y altura *h* y una caja de ancho *a*, largo *b* y altura *c*. Se desea saber cuál de ellos tiene mayor capacidad de almacenamiento.

9. Un supermercado hace un descuento del 10% por la compra de 10 unidades o más del mismo artículo ¿Cuánto deberá pagar un cliente por su compra?

10. Una compañía de seguros abrió una nueva agencia y estableció un programa para captar clientes, que consiste en lo siguiente: si el monto por el que se contrata el seguro es menor que $5.000. 000 pagará el 3%, si el monto es mayor o igual a $5.000 000 y menor de 20.000.000 pagará el 2% y si el monto es mayor o igual a 20.000.000 el valor a pagar será el 1.5 % ¿Cuál será el valor de la póliza?

11. La universidad ABC tiene un programa de estímulo a estudiantes con buen rendimiento académico. Si el promedio en las cuatro materias que se cursan en cada semestre es mayor o igual a 4.8, el estudiante no debe pagar matrícula para el siguiente semestre; si el promedio es superior o igual a 4.5 y menor de 4.8, el estudiante tendrá un descuento del 50%; para promedios mayores o iguales a 4.0 y menores a 4.5 se mantiene el valor;

mientras que para promedios menores se incrementa en un 10% respecto al semestre anterior. Dadas las notas definitivas de las materias determinar el valor de la matrícula para el siguiente semestre.

12. Una distribuidora de computadores hace un descuento dependiendo de la cantidad comprada. En compras de menos de cinco equipos no hay descuento, en compras de cinco y hasta nueve equipos hará un descuento del 5% y en compras de 10 o más el descuento será del 10%. ¿Cuál será el valor del descuento? ¿Cuánto pagará el cliente?

13. Un restaurante ofrece servicio a domicilio con las siguientes condiciones. Si el pedido es superior a $ 20.000 el servicio a domicilio no tiene ningún costo adicional, si es mayor a $ 10.000 y hasta $20.000 se cobrará un incremento de $2.000, y si es menor a 10.000 tendrá un incremento de $4.000. ¿Qué valor deberá cancelar el cliente?
14. Una distribuidora tiene tres vendedores a los que paga un sueldo básico más una comisión del 5% sobre las ventas, siempre que éstas sean mayor o iguales a $ 1.000.000. Además, el vendedor que haya vendido más durante el mes recibirá un 2% adicional sobre el valor de las ventas, indiferente del monto de éstas. ¿Cuál será el sueldo de cada vendedor?

15. En una entrevista para contratar personal se tienen en cuenta los siguientes criterios: educación formal, edad y estado civil. Los puntajes son: para edades entre 18-24 años, 10 puntos; entre 25 - 30, 20 puntos; 31 - 40 años, 15 puntos; mayores de 40, 8 puntos. Para estudios de bachillerato 5 puntos, tecnológicos 8 puntos, profesionales 10 puntos, postgrado 15 puntos. Estado civil soltero 20 puntos, casado 15 puntos, unión libre 12 puntos, separado 18 puntos. Se requiere calcular el puntaje total para un entrevistado.

16. Dado un año y un mes, se desea saber cuántos días tiene dicho mes, teniendo en cuenta que febrero cambia dependiendo si el año es bisiesto.

17. Dado un número entero entre 1 y 9999 expresarlo en palabras

18. Una joyería establece el valor de sus productos según el peso y el material. Se requiere un algoritmo que lea un valor de referencia *x* y a partir de éste calcule el valor del gramo de material utilizado y el valor del producto. El gramo de plata procesado tiene un costo de 3x, el platino 5x y el oro 8x.

19. Una empresa otorga una prima especial de maternidad / paternidad a sus empleados, dependiendo del número de hijos: para empleados que no tienen hijos, no hay prima, para un hijo la prima será del 5% del sueldo, para dos hijos será del 8%, para tres hijos el 10%, para cuatro el 12%, para un número mayor el 15%.

20. Un almacén de electrodomésticos tiene la siguiente política de venta: en ventas de contado se hace un descuento del 10%, mientras que en ventas a crédito se carga un porcentaje del valor del producto, dependiendo del número de cuotas mensuales, y se divide para el número de cuotas. Para 2 meses no hay interés de financiación, para 3

meses se incrementa el valor en 5%, para 4 meses el 10%, para 5 meses el 15% y para 6 meses el 20%. Se requiere conocer el valor del producto, el valor a pagar por el cliente, el valor del descuento o incremento y el valor de cada cuota en caso de que las haya.

21. Se requiere un algoritmo para calcular el área de una de las principales figuras geométricas: triangulo, rectángulo, círculo, rombo y trapecio. Se sabe que las áreas se obtienen con las siguientes fórmulas: área triángulo = b*h/2, área rectángulo = b*h, área círculo = Pi * radio^2, área rombo = D*d/2, área trapecio = ½h(a+b)

22. Se requiere un algoritmo para facturar llamadas telefónicas. El valor del minuto depende del tipo de llamada, así: local = $ 50, regional = 100, larga distancia nacional = 500, larga distancia internacional = 700, a celular = 200. Las llamadas de tipo regional y larga distancia nacional tienen un descuento del 5% si su duración es de 5 minutos o más.

23. Una empresa dedicada a alquilar vehículos requiere un programa para calcular el valor de cada uno de los préstamos. El valor se determina teniendo en cuenta el tipo de vehículo y el número de días que el usuario lo utiliza. El valor diario por categoría es: automóvil = $ 20000, campero = $ 30000, microbús = $ 50000 y camioneta = $ 40000.

24. La universidad José Martí otorga una beca por el 100% del valor de la matrícula al estudiante con mayor promedio en cada grupo y por el 50% al segundo mejor promedio. Conociendo los nombres y promedios de los cuatro mejores estudiantes de un grupo se desea conocer los nombres de quienes recibirán la beca.

25. Una empresa patrocinadora de atletas apoya a los deportistas que en las marcas de entrenamiento del último mes cumplen estas condiciones: el tiempo mayor no supera los 40 minutos, el tiempo menor sea inferior a 30 minutos y el tiempo promedio es menor a 35. Se requiere un algoritmo para decidir si un deportista cumple las condiciones para ser patrocinado.

26. Un estudiante de ingeniería necesita tres libros: Cálculo, Física y Programación, cuyos precios son: *v1, v2, v3* respectivamente. El estudiante cuenta con una cantidad *x* de dinero que ha recibido como apoyo económico para este fin. Se requiere un algoritmo que le ayude a decidir si puede comprar los tres, dos o un libros. El estudiante desea dar prioridad a los libros de mayor valor, ya que el dinero sobrante deberá devolverlo y los libros de menor valor serán más fácil de adquirir con recursos propios.

4.3 ESTRUCTURAS DE ITERACIÓN

Este tipo de estructuras también se conocen como ciclos, bucles o estructuras de repetición y son las que se utilizan para programar que una o más acciones se ejecuten repetidas veces. Son muy importantes en programación ya que hay muchas actividades que deben ejecutarse más de una vez.

Considere el caso de una aplicación para elaborar una factura. El cliente puede comprar un producto, pero también puede comprar dos, tres o más. Las mismas operaciones ejecutadas para facturar el primer producto deben ejecutarse para el segundo, el tercero, …, y el enésimo producto. Las estructuras iterativas permiten que se escriba las instrucciones una vez y se utilicen todas las que sea necesario.

Todas las estructuras de repetición tienen como mínimo dos partes: definición y cuerpo. La definición incluye una condición que indica cuántas veces ha de repetirse el cuerpo del ciclo; y, si no se conoce con anticipación el número de repeticiones, la condición indicará las circunstancias en que el cuerpo del ciclo se repite, por ejemplo utilizando una variable de tipo conmutador. El cuerpo del ciclo está conformado por la instrucción o el conjunto de instrucciones que incluye la estructura de repetición y que serán ejecutadas repetidas veces

4.3.1 Control de iteraciones

Para establecer la condición de iteración y tener control sobre la misma, casi siempre, es necesario utilizar una variable, ya se trate de un contador o un conmutador. A esta variable se la denomina **variable de control de ciclo** (Joyanes y Zahonero, 2002).

Una variable de control de iteraciones tiene tres momentos:

Inicialización. Para que una variable sea utilizada en la definición de una estructura iterativa es necesario conocer su valor inicial. Ya que se trata siempre de contadores o conmutadores, su valor no será leído en la ejecución del programa y es necesario que el programador lo asigne antes de incluir la variable en una condición. Si la variable no se inicializa puede ocurrir que el ciclo no se ejecute ninguna vez o que se ejecute infinitamente.

Evaluación. La evaluación de la variable puede realizarse antes o después de cada iteración, esto depende de la estructura utilizada, para decidir si ejecuta una vez más el cuerpo de ciclo o si sale y continúa con la ejecución de las demás instrucciones.

Actualización. Si la variable se inicializa en un valor que permite la ejecución del ciclo es necesario que, en cada iteración, se actualice de manera que en algún momento de la

ejecución se produzca la salida del ciclo. Si el bucle está controlado por una variable y esta no se actualiza se genera un ciclo infinito.

Al especificar iteraciones controladas por una variable es importante formularse las preguntas: ¿Cuál es el valor inicial de la variable? ¿Qué valor debe tener la variable para que el cuerpo del ciclo se repita? ¿Cómo cambia la variable? Las respuestas a estas preguntas obligarán a definir los tres momentos referidos y evitar errores de lógica.

4.3.2 Estructura MIENTRAS

Este ciclo consiste en un conjunto de instrucciones que se repiten mientras se cumpla una condición. De igual manera que en las decisiones, la condición es evaluada y retorna un valor lógico que puede ser verdadero o falso. En el caso del ciclo *mientras* las instrucciones contenidas en la estructura de repetición se ejecutarán solamente si al evaluar la condición se genera un valor *verdadero*; es decir, si la condición se cumple; en caso contrario, se ejecutará la instrucción que aparece después de *Fin mientras*.

A diferencia de otros ciclos, la estructura *mientras* comienza evaluando la expresión condicional, si el resultado es *verdadero* se ejecutan las instrucciones del cuerpo del ciclo; al encontrarse la línea *fin mientras* se vuelve a evaluar la condición, si ésta se cumple se ejecutan nuevamente las instrucciones y así sucesivamente hasta que la condición deje de cumplirse, en cuyo caso, el control del programa pasa a la línea que aparece después de *fin mientras***.** Si en la primera pasada por el ciclo la condición no se cumple las instrucciones que están dentro del ciclo no se ejecutarán ni una sola vez.

En pseudocódigo, el ciclo mientras se escribe de la siguiente forma:

Mientras *<condición>* ***hacer***
    Instrucciones que se repiten . . .
Fin mientras

En diagrama de flujo el ciclo mientras puede representarse mediante una decisión y un conector como se aprecia en la figura 51 o mediante un hexágono con dos entradas y dos salidas como se muestra en la figura 52. La primera forma es cómo se ha utilizado tradicionalmente, en este caso el ciclo no es explícito en el diagrama y sólo se identifica cuando se ejecuta paso a paso el algoritmo; en la segunda forma, utilizando un símbolo que denota iteración, el ciclo es evidente. El programa DFD que se utiliza comúnmente para elaborar diagramas de flujo utiliza la segunda forma, quizá por ello cada día es más común la utilización del símbolo de iteración.

En diagrama de Nassi – Shneiderman se representa mediante una caja para la definición del ciclo y cajas internas para las instrucciones que se repiten, como se muestra en la figura 53.

Una vez conocida la representación de esta estructura iterativa es necesario aprender cómo se utiliza en el diseño de algoritmos, para ello se presentan algunos ejemplos.

Figura 51. Representación ciclo mientras en diagrama de flujo (versión 1)

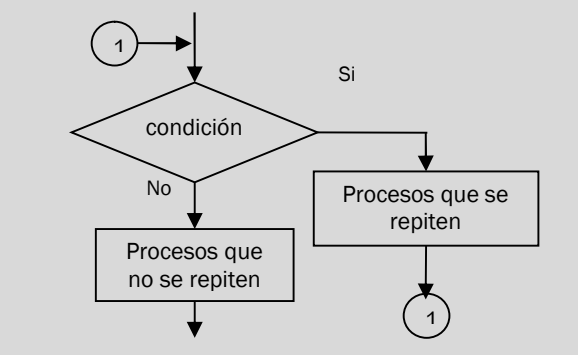


Figura 52. Representación ciclo mientras en diagrama de flujo (versión 2)

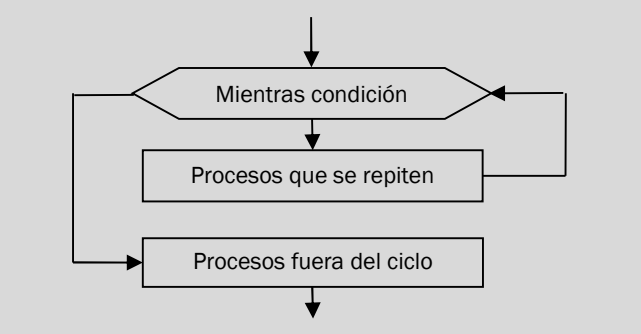
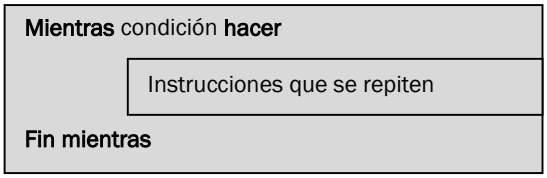


Figura 53. Representación ciclo mientras en diagrama N-S



Ejemplo 26. Generar números

Este algoritmo utiliza iteraciones para generar y mostrar los números del 1 al 10.

Lo primero por hacer es declarar una variable de tipo entero para el control del ciclo y la misma se utilizará para mostrar los números. La variable se inicializa en 1, luego se define el

ciclo, dentro de éste se muestra el número y se incrementa la variables (contador), de esta manera se llevan a cabo las tres operaciones fundamentales para controlar un ciclo mediante una variable de tipo contador: se inicializa, se evalúa en la definición del ciclo y se actualiza aproximándose progresivamente a 10, para dar fin al ciclo. El pseudocódigo se presenta en el cuadro 51, el diagrama de flujo en la figura 54 y el diagrama N-S en la figura 55.

Cuadro 51. Pseudocódigo del algoritmo para generar números

```
1.  Inicio
2.      Entero:  num = 1
3.      Mientras num <= 10 hacer
4.          Escribir num
5.          num = num + 1
6.      Fin mientras
7.  Fin algoritmo
```

En el pseudocódigo, el ciclo se encuentra entre las líneas 3 y 6, en la 3 se define el ciclo y en la 6 termina, las líneas 4 y 5 corresponden al cuerpo del ciclo; es decir, las instrucciones que se repiten. Como el ciclo *mientras* necesita que la variable esté inicializada, esta operación se lleva a cabo en la línea 2. En este caso particular, cada vez que la ejecución llegue a la línea 6 se producirá un retorno a la línea 3 para evaluar la condición del ciclo, si la condición es verdadera se ejecutan las líneas 4 y 5, pero si es falsa la ejecución salta a la línea 7.

Figura 54. Diagrama de flujo del algoritmo para generar números

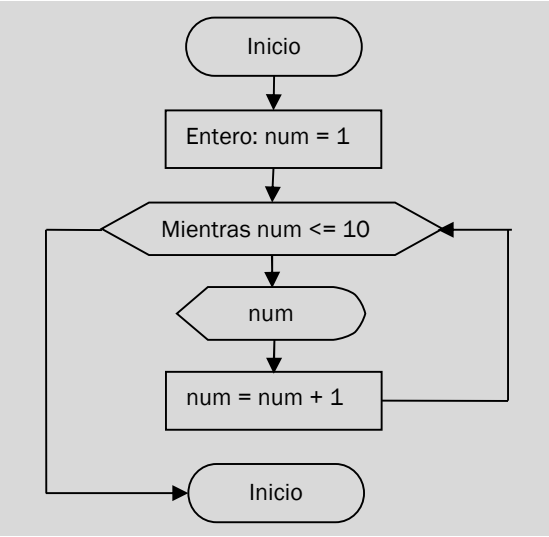
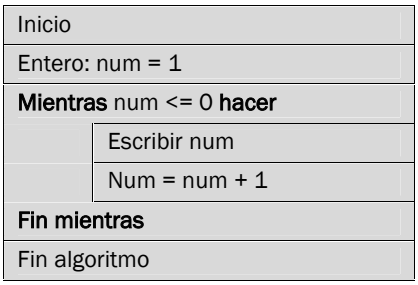


Figura 55. Diagrama N-S del algoritmo para generar números



En el diagrama de flujo se aprecia con más claridad el comportamiento de la estructura iterativa. Al declarar la variable se le asigna un valor inicial (1), en la definición del ciclo se establece la condición (num <= 10), si ésta se cumple la ejecución continúa por el flujo que sale hacia abajo, se muestra la variable y se incrementa en uno, luego vuelve a la definición del ciclo para evaluar la condición, si es verdades se ejecutan nuevamente las instrucciones, si es falsa sale por la izquierda y va al fin del algoritmo. El ciclo termina cuando num =11.

En el diagrama N-S también es fácilmente identificable el ciclo ya que toda la estructura se encuentra en una misma caja y el cuerpo del ciclo aparece internamente. Sólo cuando el ciclo ha terminado se pasa a la siguiente caja.

Para verificar el funcionamiento de este algoritmo se realiza la prueba de escritorio cuyos resultados se muestran en el cuadro 52.

Cuadro 52. Verificación algoritmo para generar números

Iteración	Num	Salida
	1	
1	2	1
2	3	2
3	4	3
4	5	4
5	6	5
6	7	6
7	8	7
8	9	8
9	10	9
10	11	10

Ejemplo 27. Divisores de un número

Se requiere un algoritmo para mostrar los divisores de un número en orden descendente.

Antes de comenzar a solucionar el ejercicio es necesario tener claridad sobre lo que se entiende por divisor de un número.

Dados dos números enteros *a* y *b*, se dice que *b* es divisor de *a* si se cumple que al efectuar una división entera *a/b* el residuo es 0. Por ejemplo, si se toman los números 27 y 3, se puede comprobar que 3 es divisor de 27 dado que al hacer la división 27/3 = 9 y el residuo es 0. Ahora bien, para mayor facilidad, el residuo de una división entera se puede obtener directamente utilizando el operador Mod.(Los operadores aritméticos se tratan en la sección 2.3.1)

Volviendo al ejercicio, en éste se pide que se muestren todos los divisores de un número comenzando por el más alto; por tanto, la solución consiste en tomar los números comprendidos entre el número ingresado y el número uno, y para cada valor examinar si cumple con la condición de ser divisor del número. Para ello es necesario utilizar una variable de control que comience en el número y disminuya de uno en uno hasta 1.

Si se toma, por ejemplo, el número 6, se deben examinar todos los números entre 6 y 1 y seleccionar aquellos que sean divisores de seis, así:

Número	contador	Número Mod contador	Divisor
6	6	0	Si
	5	1	No
	4	2	No
	3	0	Si
	2	0	Si
	1	0	Si

En la última columna se puede constatar que los divisores de seis, en orden descendente son: 6, 3, 2, 1.

Un segundo ejemplo: si se toma el número 8, se genera la lista de números desde ocho hasta uno y se verifica para cada uno si es o no divisor de ocho.

Número	Contador	Número Mod contador	Divisor
8	8	0	Si
	7	1	No
	6	2	No
	5	3	No
	4	0	Si
	3	2	No
	2	0	Si
	1	0	Si

De esto se obtiene que los divisores de 8, en orden descendente, son: 8, 4, 2, 1.

Como se observa en los ejemplos anteriores la clave para encontrar los divisores está en generar la lista de números, y para ello es apropiado utilizar una estructura iterativa y una variable de tipo contador, como se aprecia en el pseudocódigo del cuadro 53.

Cuadro 53. Pseudocódigo algoritmo divisores de un número

```
1.  Inicio
2.      Entero: num, cont
3.      Leer num
4.      cont = num
5.      Mientras cont >= 1 Hacer
6.          Si num Mod cont = 0 entonces
7.              Escribir cont
8.          Fin si
9.          cont = cont – 1
10.     Fin mientras
11. Fin algoritmo
```

Ya se mencionó la importancia de utilizar un ciclo para generar una lista de números, pero no se muestran todos los números ya que sólo se requieren los que cumplen la condición de ser divisores del número leído. La variable de control del ciclo se inicializa en el número ingresado, el ciclo se ejecuta mientras esta variable sea mayor a 1 (línea 5) y la variable se decrementa de uno en uno (línea 9). En el cuadro 54 se muestra los resultados de la verificación con los números 6 y 8.

Cuadro 54. Verificación algoritmo divisores de un número

Ejecución	num	cont	Salida
1	6	6	6
		5	
		4	
		3	3
		2	2
		1	1
		0	
2	8	8	8
		7	
		6	
		5	
		4	4
		3	
		2	2
		1	1
		0	

Las representaciones mediante diagrama de flujo y diagrama N-S se presentan en las figuras 56 y 57.

Figura 56. Diagrama de flujo algoritmo divisores de un número

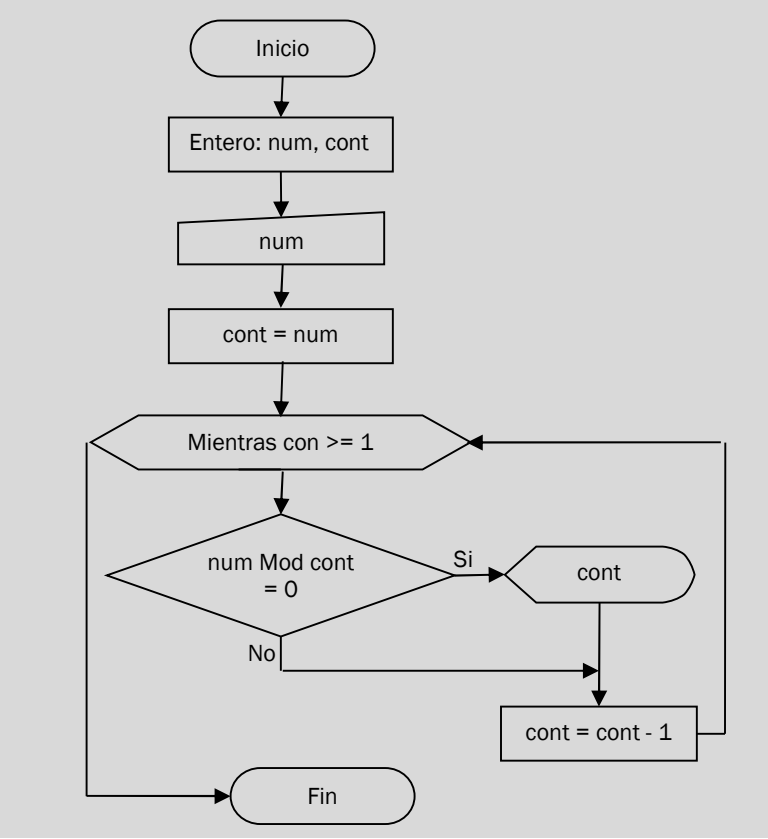
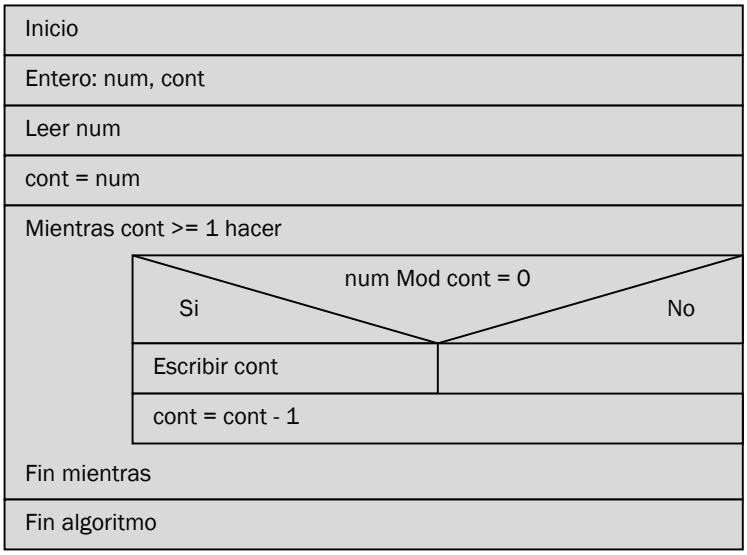


Figura 57. Diagrama N-S del algoritmo divisores de un número



4.3.3 Estructura HACER MIENTRAS

Esta estructura se propone como alternativa al ciclo *Repita – Hasta que*, por cuanto éste ya no figura en los nuevos lenguajes de programación. El ciclo *Hacer - Mientras*, permite ejecutar repetidas veces una instrucción o un grupo de instrucciones, con la particularidad de que evalúa la condición que controla el ciclo después de cada iteración; es decir que, la primera evaluación de la condición se hace después de haber ejecutado las instrucciones del ciclo, lo que garantiza que el cuerpo del bucle se ejecutará al menos una vez.

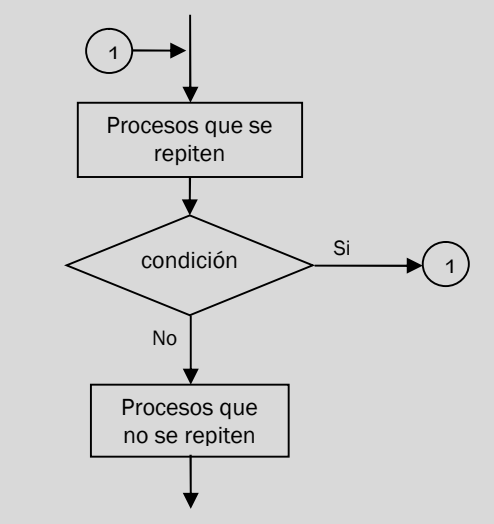
Cuando en un programa se encuentra la palabra *Hacer*, se siguen ejecutando las líneas que están a continuación, en el momento en que se encuentra la instrucción *Mientras*, se evalúa la condición asociada, la cual debe retornar un valor lógico (verdadero o falso). Si el valor es verdadero el control regresa a la instrucción *Hacer* y ejecuta nuevamente las instrucciones que le siguen. Si la condición es falsa la ejecución continúa en la instrucción que está después de *Mientras*. En pseudocódigo se escribe de así:

Hacer
Instrucción 1
Instrucción 2
. . .
Instrucción n
Mientras **<*condición*>**

En diagrama de flujo no se conoce un símbolo para representar este tipo de estructura; por ello, se lo implementa mediante un símbolo de decisión y un conector para regresar a un

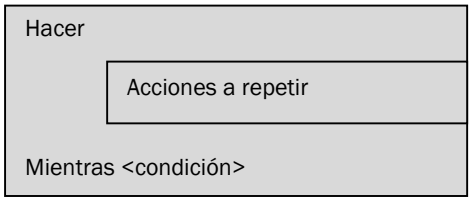
proceso anterior y repetir la secuencia mientras la condición sea verdadera, como se muestra en la figura 58.

Figura 58. Ciclo Hacer - mientras en Diagrama de Flujo



En diagrama N-S se utiliza una caja para representar el ciclo y cajas internas para las acciones que conforman el cuerpo del ciclo, como se presenta en la figura 59.

Figura 59. Ciclo Hacer – Mientras en Diagrama N-S



Ejemplo 28. Sumar enteros positivos

Este algoritmo lee números enteros y los suma mientras sean positivos. Cuando se introduce un número negativo o el cero el ciclo termina, se muestra la sumatoria y termina el algoritmo. El pseudocódigo se muestra en el cuadro 55.

Cuadro 55. Pseudocódigo del algoritmo sumar enteros

```
1.  Inicio
2.      Entero: num = 0, suma = 0
3.      Hacer
4.          suma = suma + num
5.          Leer num
6.      Mientras num > 0
7.      Escribir suma
8.  Fin algoritmo
```

En este algoritmo se declara dos variables y se inicializan, esto debido a que la primera instrucción a ejecutar es la suma y es necesario tener control sobre el contenido de las variables. En la línea 5 se lee un número, pero para efectuar la suma es preciso volver atrás, lo cual está sujeto a la condición de ser entero positivo. Si se cumple la condición la ejecución regresa a la línea 3, en la 4 efectúa la suma y vuelve a leer un número, y así sucesivamente hasta que se ingrese el cero o un número negativo. Sólo al terminar el ciclo muestra el resultado de la sumatoria y termina la ejecución. En la Figura 60 se presenta el diagrama de flujo y en la 61 el diagrama N-S.

Figura 60. Diagrama de Flujo del algoritmo sumar enteros

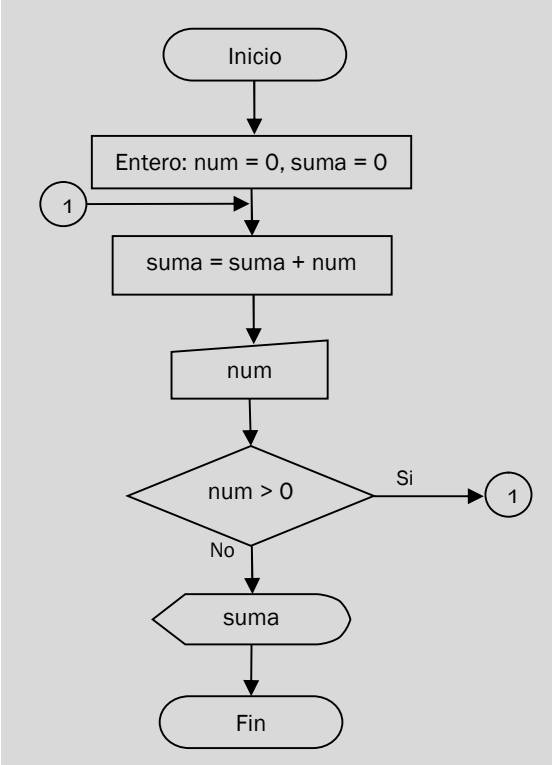
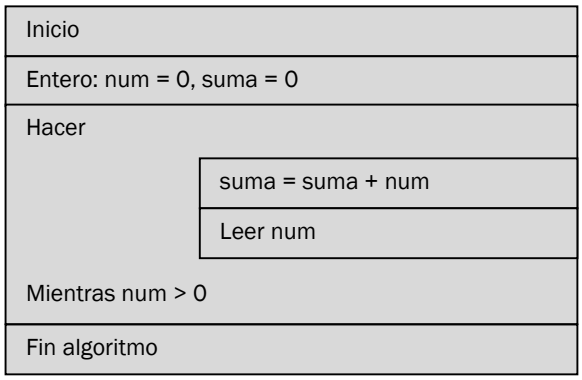


Figura 61. Diagrama N-S del algoritmo sumar enteros



En el cuadro 56 se muestra los resultados de la verificación del algoritmo.

Cuadro 56. Verificación del algoritmo sumar enteros

Iteración	Num	Suma	num > 0	Salida
	0	0		
1	4	4	V	
2	8	12	V	
3	1	13	V	
4	5	18	V	
5	0		F	
				18

Ejemplo 29. Número binario

Dado un número en base 10, este algoritmo calcula su equivalente en base 2; es decir, convierte un decimal a binario.

Una forma de pasar un valor en base 10 (decimal) a base 2 (binario) consiste en dividir el valor sobre 2 y tomar el residuo, que puede ser 0 o 1, retomar el cociente y dividir nuevamente para 2. Esta operación se efectúa repetidas veces hasta que el resultado sea 0. No obstante, esta solución tiene una breve dificultad que hay que resolver y es que los residuos forman el número en binario, pero deben leerse en sentido contrario; es decir, el primer número que se obtuvo ocupa la posición menos significativa mientras que el último debe ocupar la primera posición en el número binario.

Considérese el número 20

20 / 2 = 10 Residuo = 0

10 / 2 = 5 Residuo = 0
5 / 2 = 2 Residuo = 1
2 / 2 = 1 Residuo = 0
1 / 2 = 0 Residuo = 1

Si los números se toman en el orden que se generan se obtiene: 00101, este número equivale a 5 en decimal, el cual está muy lejos del valor original 20.

Los números que se generan se toman en orden inverso, el último residuo es el primer dígito del binario; por lo tanto, el número binario generado en esta conversión es: 10100. Este sí es equivalente a 20.

Ahora, la pregunta es ¿cómo hacer que cada nuevo dígito que se obtiene se ubique a la izquierda de los anteriores?

Para solucionar este asunto, se utiliza una variable para llevar control de la posición que debe ocupar el dígito dentro del nuevo número. La variable inicia en 1 y en cada iteración se la multiplica por 10 para que indique una posición cada vez más significativa. Entonces se tiene que:

Entrada: número en base 10 (decimal)
Salida: número en base 2
Proceso:
dígito = decimal Mod 2
binario = binario + digito * posición
decimal = decimal / 2
posición = posición * 10

La solución a este problema se presenta en el cuadro 57.

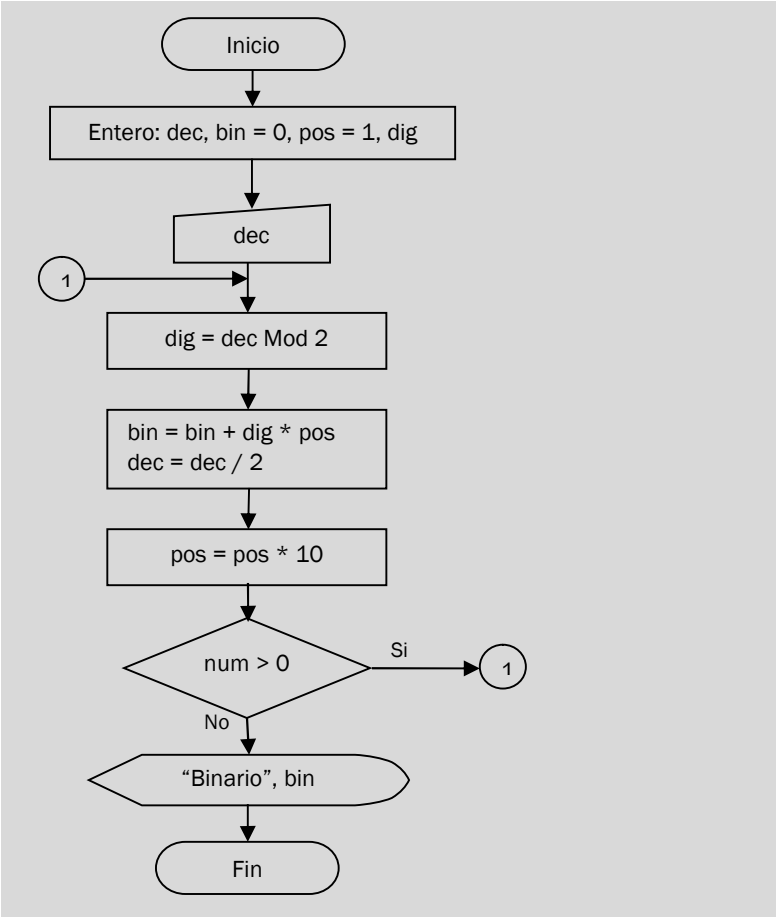
Cuadro 57. Pseudocódigo del algoritmo número binario

```
1.  Inicio
2.      Entero:  dec, bin = 0, pos = 1, dig
3.      Leer dec
4.      Hacer
5.          dig = dec Mod 2
6.          bin = bin + dig * pos
7.          dec = dec / 2
8.          pos = pos * 10
9.      Mientras dec > 0
10.     Escribir “Binario:”, bin
11. Fin algoritmo
```

En el algoritmo se aprecia que las operaciones de las líneas 5 a 8 se repiten mientras haya un número para dividir (*dec > 0*). En la línea 5 se obtiene un digito (0 o 1) que corresponde al residuo de dividir el número decimal sobre dos, ese dígito se multiplica por la potencia de 10 para convertirlo en 10, 100, 1000 o cualquiera otra potencia si se trata de un uno, para luego sumarlo con los dígitos que se hayan obtenido anteriormente (línea 6). El número decimal se reduce cada vez a la mitad, de esta manera tiende a cero (línea 7), al tiempo que la potencia que indica la posición del siguiente uno se multiplica por 10 en cada iteración (línea 8).

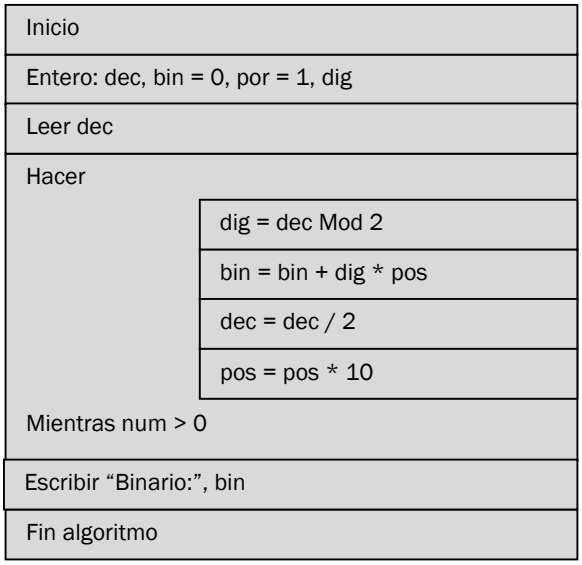
Los diagramas de flujo y N-S de este algoritmo se presentan en las figuras 62 y 63 respectivamente.

Figura 62. Diagrama de flujo del algoritmo número binario



Para probar este algoritmo se calcula el equivalente binario para los números 20 y 66 obteniendo como resultados 10100 y 1000010 respectivamente, los cuales son correctos. Los datos generados en las pruebas se presentan en el cuadro 58.

Figura 63. Diagrama N-S del algoritmo número binario



Cuadro 58. Verificación del algoritmo número binario

Iteración	dig	dec	Bin	pos	Salida
			0	1	
		20	0		
1	0	10	0	10	
2	0	5	0	100	
3	1	2	100	1000	
4	0	1	100	10000	
5	1	0	10100	100000	Binario: 10100
		66	0	1	
1	0	33	0	10	
2	1	16	10	100	
3	0	8	10	1000	
4	0	4	10	10000	
5	0	2	10	100000	
6	0	1	10	1000000	
7	1	0	1000010		Binario: 1000010

4.3.4 Estructura PARA

Esta estructura, al igual que las anteriores, permite ejecutar repetidas veces una instrucción o un grupo de ellas, pero a diferencia de otras instrucciones de repetición, ésta maneja el valor inicial, el valor de incremento o decremento y el valor final de la variable de control como parte de la definición del ciclo.

Cuando al ejecutarse un algoritmo se encuentra una instrucción *para* la variable de control (contador) toma el valor inicial, se verifica que el valor inicial no sobrepase el valor final y luego se ejecutan las instrucciones del ciclo. Al encontrar la instrucción *fin para*, se produce el incremento y se vuelve a verificar que la variable de control no haya superado el límite admitido, y se vuelven a ejecutar las instrucciones que están dentro del ciclo, y así sucesivamente tantas veces como sea necesario hasta que se supere el valor final establecido.

El ciclo *para* termina en el momento en que la variable de control (contador) sobrepasa el valor final; es decir, que la igualdad está permitida y las instrucciones se ejecutan cuando el contador es igual al valor final.

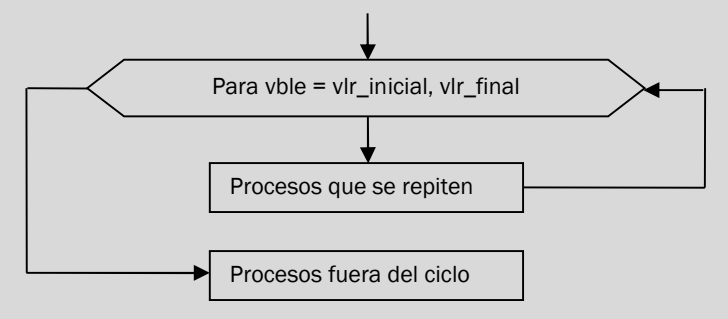
Algunos lenguajes de programación definen la sintaxis de este ciclo incluyendo una condición que se debe cumplir, de forma similar al ciclo mientras, y no un valor final para la variable, como se propone en pseudocódigo.

Este ciclo puede presentarse de tres maneras: la primera es la más común, cuando se produce un incremento de 1 en cada iteración, en cuyo caso no es necesario escribir explícitamente este valor. En pseudocódigo se expresa de la siguiente forma:

Para *variable = valor_inicial* ***hasta*** *valor_final* ***hacer***
Instrucciones
Fin para

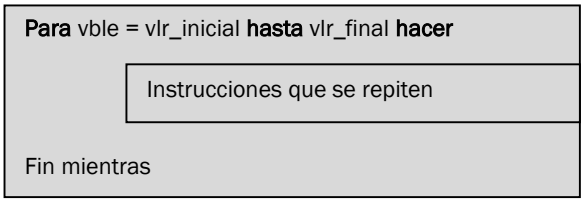
En diagrama de flujo se representa como se muestra en la figura 64 y en diagrama N-S, como la figura 65.

Figura 64. Ciclo para en Diagrama de Flujo (Versión 1)



En el diagrama de flujo no es necesario escribir todas las palabras, éstas se reemplazan por comas (,) excepto la primera.

Figura 65. Ciclo para en Diagrama N-S (Versión 1)



El segundo caso de utilización del ciclo *para* se presenta cuando el incremento es diferente de 1, en cuyo caso se escribirá la palabra *incrementar* seguida del valor a sumar en cada iteración.

En pseudocódigo se escribe:

Para *vble = vlr_inicial* ***hasta*** *vlr_final* ***incrementar*** *valor* ***hacer***
Instrucciones que se repiten
Fin para

Diagrama de flujo y en diagrama N-S se representa como en la figuras 66 y 67.

Figura 66. Ciclo Para en Diagrama de Flujo (Versión 2)

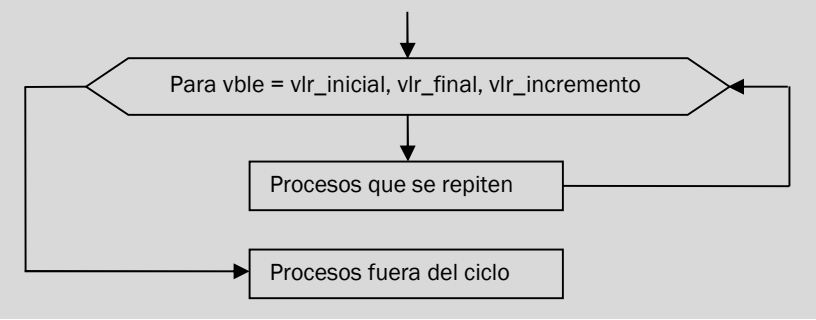
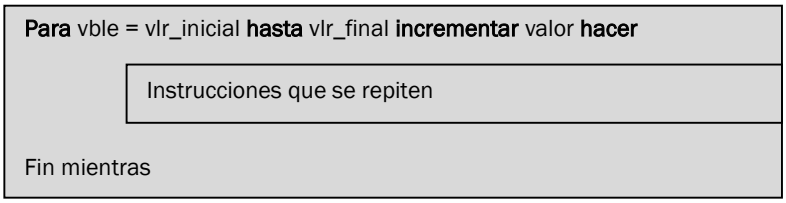


Figura 67. Ciclo Para en N-S (Versión 2)



El tercer caso se presenta cuando el ciclo *para* no se incrementa desde un valor inicial hasta un valor mayor, sino que disminuye desde un valor inicial alto hasta un valor menor. Para ello es suficiente con escribir *decrementar* en vez de *incrementar*.

En pseudocódigo se tiene:

Para contador = valor_inicial hasta valor_final decrementar valor hacer
    Instrucciones
Fin para

En este caso para que se ejecute la primera iteración será necesario que el valor inicial sea mayor que el valor final, en caso contrario, simplemente pasará a ejecutarse la instrucción que sigue al *fin para***.**

Es importante tener en cuenta que cuando la variable de control debe disminuir en cada iteración, siempre hay que escribir *decrementar* y el valor, aunque sea *-1*.

En diagrama de flujo será suficiente con anteponer el signo menos (-) al valor a decrementar.

Ejemplo 30. Sumatoria iterativa

Dado un número *n*, entero positivo, se calcula y se muestra la sumatoria de los números desde 1 hasta *n*.

En este caso se requiere una estructura iterativa para generar los números desde el uno hasta *n* y cada valor comprendido en este intervalo se almacena en una variable de tipo acumulador.

Ejemplo: si se introduce el número 3, la sumatoria será:

1 + 2 + 3 = 6

Si se introduce el número 6, la sumatoria será:

1 + 2 + 3 + 4 + 5 + 6 = 21

El ejercicio se puede sistematizar de la siguiente forma:

Entrada: número
Salida: sumatoria
Proceso:

$$\text{sumatoria} = \sum_{i=1}^{i=n} i$$

La solución de este ejercicio se presenta en el cuadro 59 en notación de pseudocódigo y en las figuras 68 y 69 los diagramas de flujo y N-S.

Cuadro 59. Pseudocódigo algoritmo sumatoria

```
1.  Inicio
2.      Entero: num, i, suma = 0
3.      Leer num
4.      Para i = 1 hasta num hacer
5.          suma = suma + i
6.      Fin para
7.      Escribir “Sumatoria:”, suma
8.  Fin algoritmo
```

Figura 68. Diagrama de flujo de algoritmo sumatoria

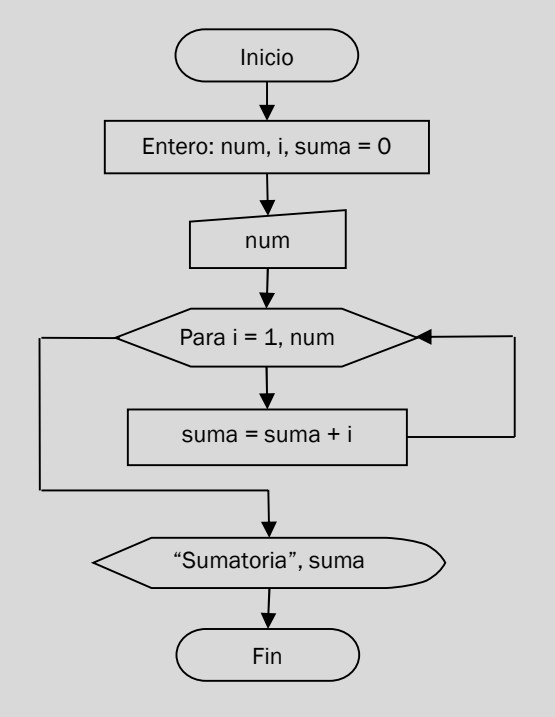
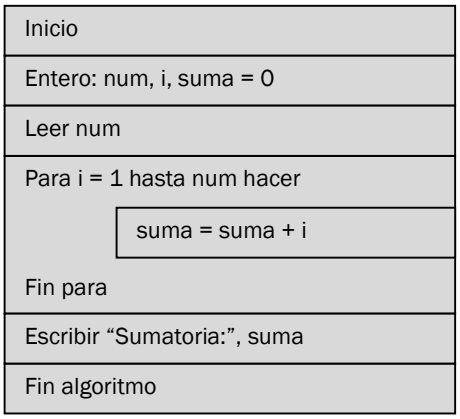


Figura 69. Diagrama N-S del algoritmo sumatoria



Dado que el ciclo *para* incrementa automáticamente la variable, dentro del ciclo solo aparece la instrucción para sumar los números generados.

En el cuadro 60 se muestra el comportamiento de las variables al ejecutar el algoritmo para calcular la sumatoria de los números del 1 al 6.

Cuadro 60. Verificación de algoritmo sumatoria

Iteración	Num	i	suma	Salida
			0	
	6		0	
1		1	1	
2		2	3	
3		3	6	
4		4	10	
5		5	15	
6		6	21	Sumatoria: 21

Ejemplo 31. Tabla de multiplicar

Comúnmente se conoce como tabla de multiplicar de un número a la lista de los primeros 10 múltiplos. Por ejemplo, la tabla del 2:

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12

2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

Para mostrarla de esta forma se puede utilizar la estructura iterativa *para*, la que hace variar el multiplicador desde 1 hasta 10 y en cada iteración se calcula el producto y muestra los datos. El pseudocódigo se muestra en el cuadro 61, los diagramas de flujo y N-S en las figuras 70 y 71.

Cuadro 61. Pseudocódigo del algoritmo tabla de multiplicar

```
1.  Inicio
2.      Entero:  m, n, r
3.      Leer m
4.      Para n = 1 hasta 10 hacer
5.          r = m * n
6.          Escribir m, '*', n, '=', r
7.      Fin para
8.  Fin algoritmo
```

Figura 70. Diagrama de flujo del algoritmo tabla de multiplicar

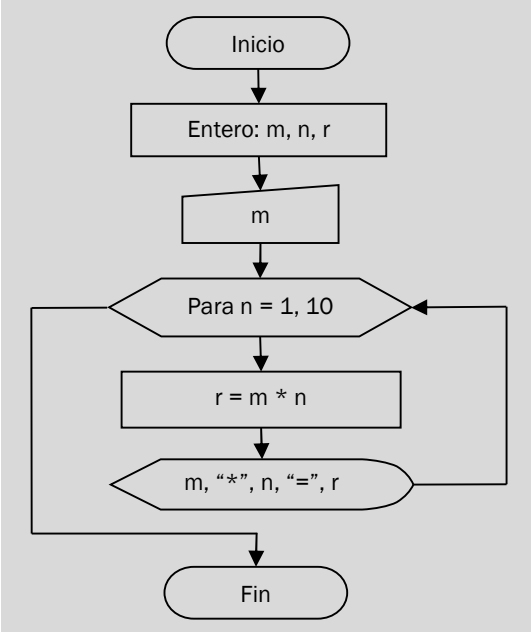
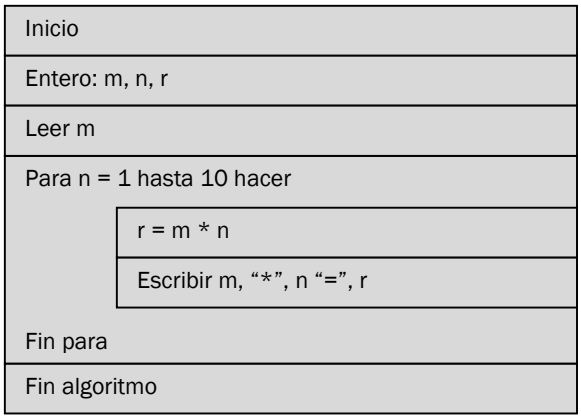


Figura 71. Diagrama N-S del algoritmo tabla de multiplicar



Para verificar el funcionamiento del algoritmo se genera la tabla del 4, los resultados se presentan en el cuadro 62.

Cuadro 62. Verificación del algoritmo tabla de multiplicar

Iteración	m	n	r	Salida
	4			
1		1	4	4 * 1 = 4
2		2	8	4 * 2 = 8
3		3	12	4 * 3 = 12
4		4	16	4 * 4 = 16
5		5	20	4 * 5 = 20
6		6	26	4 * 6 = 24
7		7	28	4 * 7 = 28
8		8	32	4 * 8 = 32
9		9	36	4 * 9 = 36
10		10	40	4 * 10 = 40

4.3.5 Estructuras Iterativas anidadas

Las estructuras iterativas, al igual que las selectivas, se pueden anidar; es decir, se pueden colocar una dentro de otra.

El anidamiento de ciclos se conforma por un ciclo externo y uno o más ciclos internos, donde cada vez que se repite el ciclo externo, los internos se reinician y ejecutan todas las iteraciones definidas (Joyanes, 2000).

Un ejemplo que permite ilustrar fácilmente el concepto de ciclos anidados es el reloj digital. El tiempo se mide en horas, minutos y segundos, las horas están conformadas por minutos y

los minutos por segundos, de manera que si se escribe un ciclo para las horas, otro para los minutos y otro para los segundos, el ciclo de los minutos en cada iteración deberá esperar a que se ejecuten las 60 iteraciones del ciclo de los segundos (0..59) y el de las horas deberá esperar que se ejecute el de los minutos (0..59).

Ejemplo 32. Reloj digital

Para implementar un reloj digital es necesario declarar tres variables, para controlar: horas, minutos y segundos. Cada una de estas variables controla un ciclo, así: los segundos se incrementan desde 0 a 59, los minutos también desde 0 a 59 y las horas desde 1 a 12 o de 1 a 24.

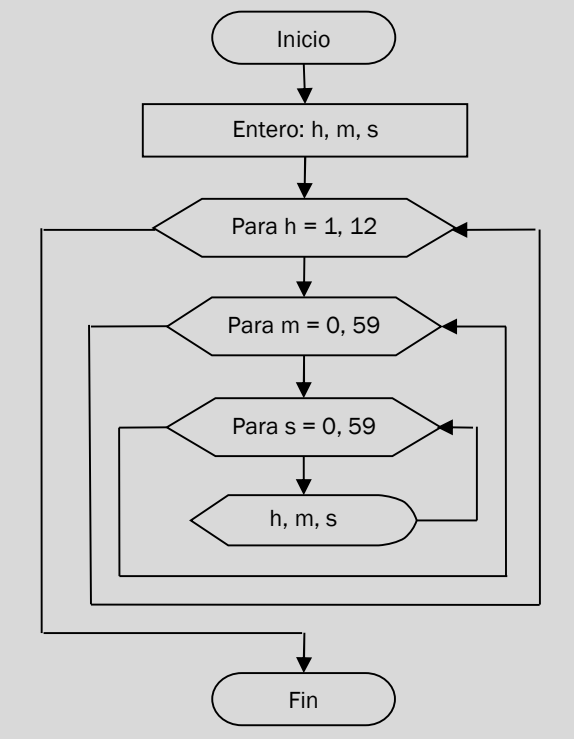
La variable que controla las horas se incrementa en uno cada vez que la variable que controla los minutos ha hecho el recorrido completo desde 0 a 59, a su vez, la variable de los minutos se incrementa en uno cada vez que la variable de los segundos ha completado su recorrido desde 0 a 59. Para que esto ocurra es necesario que el ciclo de las horas contenga al de los minutos y éste, al de los segundos, como se aprecia en el cuadro 63.

Cuadro 63. Pseudocódigo algoritmo Reloj digital

```
1.  Inicio
2.      Entero: h, m, s
3.      Para h = 1 hasta 12 hacer
4.          Para m = 0 hasta 59 hacer
5.              Para s = 0 hasta 59 hacer
6.                  Escribir h,":", m, ":", s
7.              Fin para
8.          Fin para
9.      Fin para
10. Fin algoritmo
```

Este mismo algoritmo expresado en diagrama de flujo se presenta en la figura 72 y en diagrama N-S en la 73.

Figura 72. Diagrama de flujo del algoritmo Reloj digital

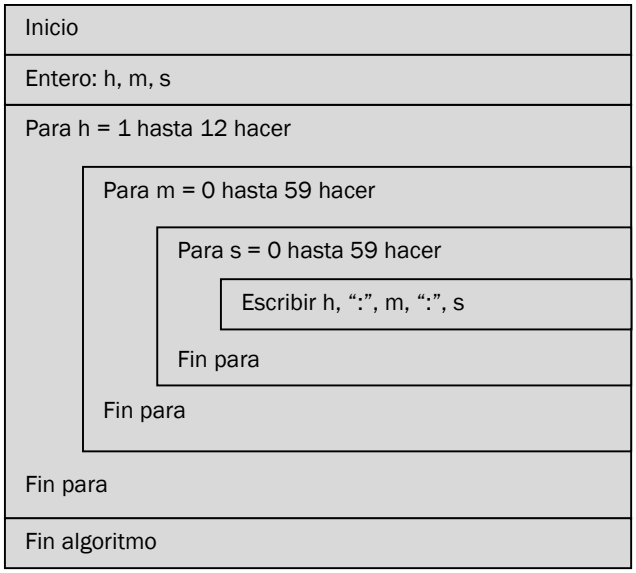


Al ejecutar este algoritmo se tendría una salida de la forma:

1:0:0
1:0:1
1:0:2
…
1:0:58
1:0:59
1:1:0
1:1:1
1:1:2
…
1:59:58
1:59:59
2:0:0
2:0:1
2:0:2
…

Este algoritmo está diseñado para terminar la ejecución al llegar a 12:59:59. Para hacer que el reloj se ejecute indefinidamente sería necesario colocar los tres ciclos dentro de otro, para que al llegar a 12:59:59 se reinicie la variable h y vuelva a comenzar el conteo. El ciclo externo tendría que ser un ciclo infinito de la forma: *mientras verdadero hacer*.

Figura 73. Diagrama N-S del algoritmo Reloj digital



Ejemplo 33. Nueve tablas de multiplicar

En un ejemplo anterior se utilizó un ciclo para generar la tabla de multiplicar de un número, en éste se utiliza ciclos anidados para generar las tablas desde dos hasta 10. El primer ciclo está asociado con el multiplicando y hace referencia a cada una de las tablas a generar, por tanto toma valores entre 2 y 10; el ciclo interno está relacionado con el multiplicador, se encarga de generar los múltiplos en cada una de las tablas, en consecuencia toma valores entre 1 y 10. El pseudocódigo se presenta en el cuadro 64.

Cuadro 64. Pseudocódigo para Nueve tablas de multiplicar

```
1.  Inicio
2.      Entero: m, n, r
3.      Para m = 2 hasta 10 hacer
4.          Escribir "Tabla de multiplicar del ", m
5.          Para n = 1 hasta 10 hacer
6.              r = m * n
7.              Escribir m, "*", n, "=", r
8.          Fin para
9.      Fin para
10. Fin algoritmo
```

Al ejecutar este algoritmo se tendrá una salida de la forma:

Tabla de multiplicar del 2

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
…
Tabla de multiplicar del 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
…
Y así sucesivamente hasta

10 * 10 = 100

Los diagramas de flujo y N-S para este algoritmo se presentan en las figuras 74 y 75.

Figura 74. Diagrama de flujo para 9 tablas de multiplicar

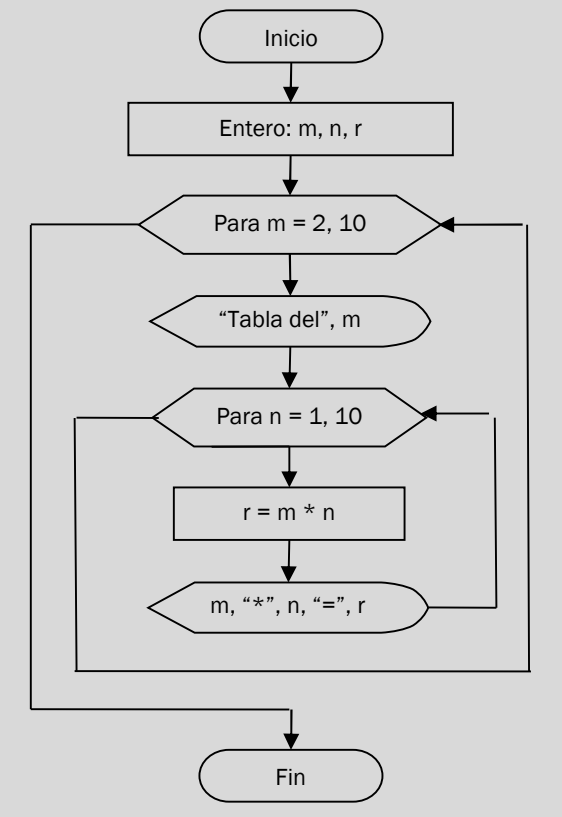
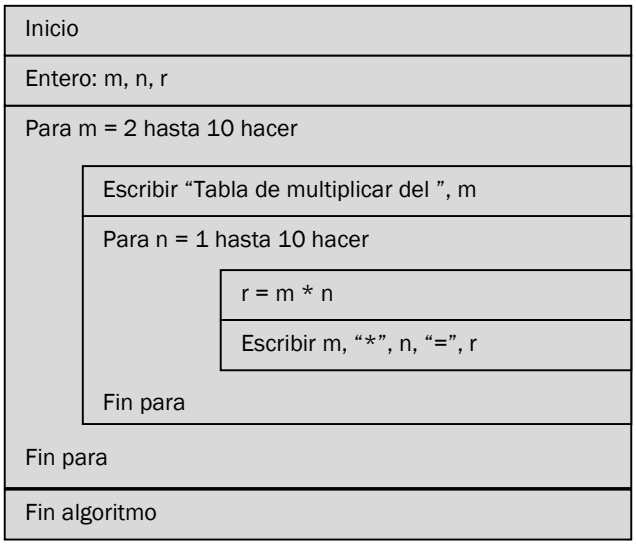


Figura 75. Diagrama N-S para Nueve tablas de multiplicar



4.3.6 Más ejemplos de iteraciones

Ejemplo 34. Mayor, menor y promedio de *n* números

Este ejercicio consiste en leer *n* números y luego reportar el número menor y el número mayor de la lista y calcular el valor promedio.

Supóngase que el usuario ingresa la siguiente lista de datos:
2
14
55
6
12
34
17

Se tiene que:
Cantidad de números ingresados: 7
Número mayor: 55
Número menor: 2
Sumatoria: 140
Promedio = 140 / 7 = 20

Para solucionar este problema, lo primero que hay que plantear es cómo se sabrá cuando terminar la lectura de datos, ya que no se especifica la cantidad de números que serán ingresados y en el planteamiento del problema no se da ninguna condición que permita saber

cuándo terminar el ciclo. Este tipo de problemas es muy común, no se conoce con antelación la cantidad de datos a procesar.

Para implementar ciclos cuando no se conoce el número de iteraciones a ejecutar hay al menos tres formas muy sencillas de hacerlo: la primera consiste en definir una variable para almacenar este dato (*n*) y se pide al usuario que la ingrese, de la forma "*cuántos números desea ingresar?*" y el dato ingresado se utiliza como valor final para implementar una variable de control; la segunda, utilizar un conmutador o bandera, se trata de definir una variable con dos posibles valores: si y no o 0 y 1, por ejemplo, preguntar al usuario "*Otro número (S/N) ?*", si ingresa "*S*" se continúa leyendo datos, en caso contrario el ciclo termina; y la tercera, hacer uso de un centinela que termine el ciclo cuando se cumpla una condición, por ejemplo, el ciclo termina cuando se ingresa el número 0, y se presenta al usuario un mensaje de la forma: "*Ingrese un número (0 para terminar): "*.

Para solucionar este ejercicio se opta por la primera propuesta. Se cuenta con la siguiente información:

Datos de entrada: cantidad de números, números
Datos de salida: promedio, mayor y menor

Procesos:
suma = suma + número
promedio = suma / cantidad de números

Para identificar los números menor y mayor se puede optar por una de estas dos alternativas: primera: inicializar la variable del número menor en un número muy grande y la variable del número mayor en 0, de manera que en la primera iteración se actualicen; segunda: no inicializar las variables y asignarles el primera dato que se ingrese tanto a la variable del mayor como del menor. En este caso se optó por la segunda, el algoritmo se presenta en el cuadro 65.

En este algoritmo se utilizan las variables: *can, num, mayor, menor* y *prom* que son variables de trabajo; *cont* es el contador que permite controlar el ciclo, *suma* es el acumulador que totaliza los números ingresados para luego poder obtener el promedio.

Es importante comprender qué instrucciones deben ir dentro del ciclo y cuales fuera, ya sea antes o después. No hay una regla de oro que se pueda aplicar a todos los casos, pero puede ayudar el recordar que todo lo que está dentro del ciclo se repite; es decir, hay que preguntarse cuántas veces tendrá que ejecutarse la instrucción, si la respuesta es solo una vez, no hay razón para que esté dentro del bucle. En este ejemplo, leer la cantidad de números a trabajar sólo debe ejecutarse una vez ya que esta cantidad permitirá establecer el límite de las repeticiones y calcular el promedio también debe hacerse solo una vez, por lo tanto se ubican fuera del ciclo, pero sumar el número entrado debe realizarse tantas veces como números se ingresen, por ende esta instrucción aparece dentro del bucle.

Cuadro 65. Pseudocódigo para número menor, mayor y promedio

```
1.  Inicio
2.     Entero: can, num, suma=0, menor, mayor, cont = 0
3.     Real: prom
4.     Leer can
5.     Mientras cont < can hacer
6.          Leer num
7.          suma = suma + num
8.          Si cont = 0 entonces
9.               menor = num
10               mayor = num
11          Si no
12               Si num < menor  entonces
13                    menor = num
14               Fin si
15               Si num > mayor entonces
16                    mayor = num
17               Fin si
18          Fin si
19          cont = cont + 1
20     Fin mientras
21     prom = suma / can
22     Escribir "Número menor:", menor
23     Escribir "Promedio:", prom
24     Escribir "Número mayor:", mayor
25  Fin algoritmo
```

El ciclo se ejecutará siempre que *cont* sea menor que el contenido de la variable *can*, como *cont* inicia en 0, el ciclo se ejecuta para cualquier valor de *can* que sea mayor o igual a 1, pero si el valor ingresado para *can* fuera 0, el ciclo no se ejecutaría y se generaría un error de división sobre 0 al calcular el promedio. La última instrucción del ciclo es *cont = cont + 1*, ésta es indispensable, ya que el contador debe incrementarse en cada iteración para que alcance el valor de la variable *can*. No se debe olvidar que todo ciclo debe tener un número limitado de iteraciones, por ello al diseñar el algoritmo se debe prever la forma en que el ciclo terminará.

En el cuadro 66 se presenta el comportamiento de las variables y los resultados de una ejecución.

Cuadro 66. Verificación del algoritmo Número menor, mayor y promedio

Iteración	Can	num	menor	Mayor	Cont	suma	Prom	Salida
	7				0	0		
1		2		2	1	2		
2		14	2	14	2	16		
3		55	2	55	3	71		
4		6	2	55	4	77		
5		12	2	55	5	89		
6		34	2	55	6	123		
7		17	2	55	7	140		
							20	Número menor: 2 Promedio = 20 Número mayor: 55

Ejemplo 35. Procesamiento de notas

En un grupo de *n* estudiantes se realizaron tres evaluaciones parciales. Se requiere calcular la nota definitiva de cada estudiante y conocer la cantidad de estudiantes que aprobaron, la cantidad de estudiantes que reprobaron y la nota promedio del grupo, considerando que el valor porcentual de cada parcial lo acordaron entre el docente y el grupo, y la nota mínima aprobatoria es 3,0.

Aplicando la estrategia que se ha propuesto para comprender el problema supóngase un caso particular, como éste:

Grupo de segundo semestre: 20 estudiantes
Materia: programación
Porcentaje para primera evaluación: 30%
Porcentaje para segunda evaluación: 30%
Porcentaje para tercera evaluación: 40%

Ahora la solución para un estudiante:

El estudiante Pedro Pérez obtiene las siguientes notas:
Evaluación 1: 3,5
Evaluación 2: 4,0
Evaluación 3: 2,8
Cuál es la nota definitiva de Pedro?

Aplicando los porcentajes previamente mencionados se tiene que:

Nota definitiva = 3,5 * 30% + 4,0 * 30% + 2,8 * 40%

Nota definitiva = 1,05 + 1,2 + 1,12
Nota definitiva = 3,4

El siguiente paso es decidir si el estudiante aprueba o reprueba la materia:

Nota definitiva >= 3,0? (3,4 >= 3.0?) Si

Pedro Pérez aprobó el curso; en consecuencia se lo cuenta como uno de los que aprobaron, y para poder obtener el promedio del curso es necesario sumar su nota a una variable de tipo acumulador que al final se dividirá sobre el contador de estudiantes (20).

Este mismo procedimiento se tiene que llevar a cabo para cada uno de los estudiantes del grupo.

A continuación se identifican y clasifican los datos del problema:

Datos de entrada: nombre del estudiante, nota1, nota2, nota3, porcentaje1, porcentaje2, porcentaje3.

Datos de salida: nota definitiva, promedio, número pierden, número ganan

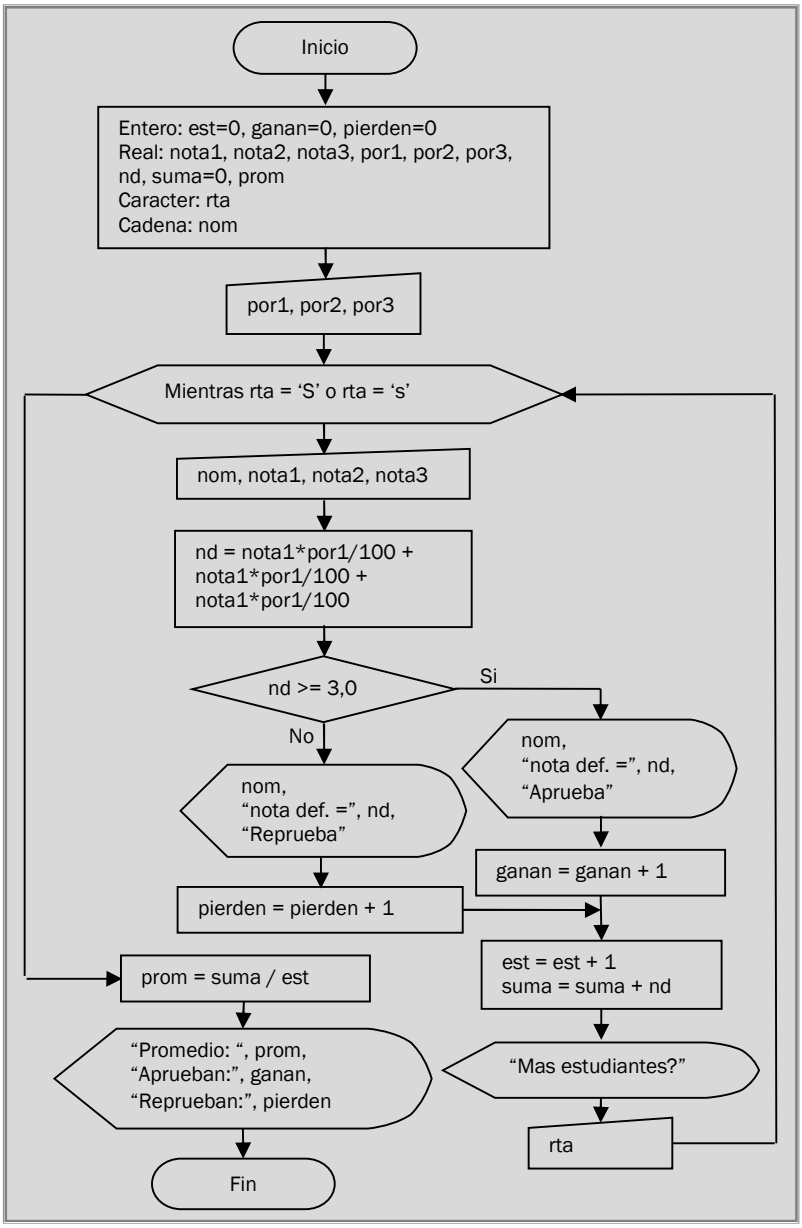
Proceso:
nota definitiva = nota1 * porcentaje1 + nota2 * porcentaje2 + nota3 * porcentaje3
suma = suma + nota definitiva
promedio = suma / numero estudiantes

En el análisis del ejercicio anterior se mencionó que hay tres formas de programar un ciclo cuando no se conoce por anticipado el número de iteraciones y se implementó la primera; en este ejemplo, se utilizará la segunda, que consiste en preguntar al usuario si desea continuar entrando datos. En el momento en que responda negativamente el ciclo termina.

En este algoritmo se utiliza tres contadores y un acumulador además de las variables de trabajo. Las variables utilizadas para registrar el número de estudiantes, el número de estudiantes que ganan y el número que pierden la materia son contadores, mientras que la variable utilizada para totalizar las notas definitivas es un acumulador. El ciclo no está controlado por la cantidad predefinida de estudiantes, sino por el contenido de la variable *rta* de tipo caracter que debe tomar como valores '*S*' o '*N*'.

El diseño de la solución se presenta en la figura 76 en notación de diagrama de flujo y los resultados de una ejecución de prueba con cuatro estudiantes se muestran en el cuadro 67.

Figura 76. Diagrama de flujo para procesamiento de notas



Cuadro 67. Verificación del algoritmo procesamiento de notas

por1	por2	Por3	nom	Nota 1	Nota 2	Nota 3	Nd	ganan	pierden	est	suma	rta	prom
								0	0	0	0	S	
30	30	40											
			A	3.0	3.5	4.0	3.5	1		1	3.5	S	
			B	2.0	2.5	2.0	2.5		1	2	6	S	
			C	4.0	3.2	2.0	3.0	2		3	9	S	
			D	2.2	3.5	2.5	2.7		2	4	11.7	n	2.9

La salida en pantalla es la siguiente:

```
Nombre: A
Nota def: 3,5
Aprueba

Nombre: B
Nota def: 2,5
Reprueba

Nombre: C
Nota def: 3,0
Aprueba

Nombre: D
Nota def: 2,7
Reprueba

Promedio: 2,9
Aprueban: 2
Reprueban: 2
```

Ejemplo 36. Serie Fibonacci

Esta conocida serie propuesta por el matemático italiano del mismo nombre corresponde a un modelo matemático que explica la reproducción de los conejos y fue publicada por primera vez en 1202 en una obra titulada *Liberabaci.*

Fibonacci planteó el problema de la siguiente manera: supóngase que una pareja de conejos da lugar a dos descendientes cada mes y cada nuevo ejemplar comienza su reproducción después de dos meses. De manera que al comprar una pareja de conejos, en los meses uno y dos se tendrá una pareja, pero al tercer mes se habrán reproducido y se contará con dos parejas, en el mes cuatro solo se reproduce la primera pareja, así que el número aumentará a tres parejas; en el mes cinco, comienza la reproducción de la segunda pareja, con lo cual se obtendrán cinco parejas, y así sucesivamente. Si ningún ejemplar muere, el número de conejos que se tendrán cada mes está dado por la sucesión: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …,

Si se asume que los dos primeros números son constantes; es decir, si se establece que los dos primeros números de la sucesión son 0 y 1, los siguientes se pueden obtener sumando los dos anteriores, así:

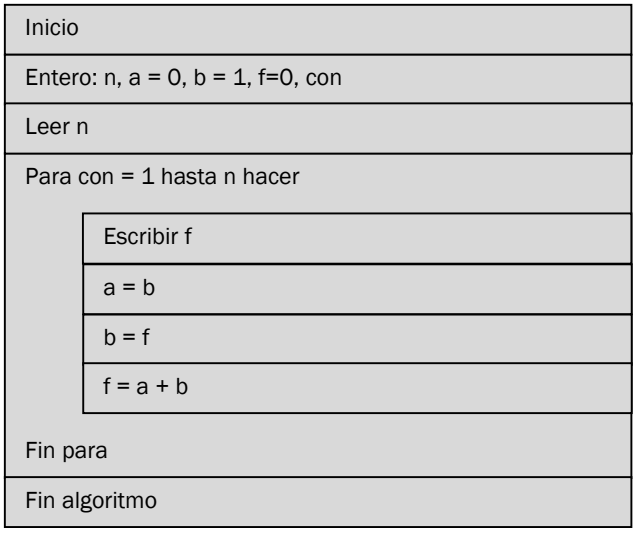
Término	Valor	Obtenido de
1	0	Constante
2	1	Constante
3	1	0 + 1
4	2	1 + 1
5	3	2 + 1
6	5	3 + 2
7	8	5 + 3
8	13	8 + 5
9	21	13 + 8
10	34	21 + 13

Este tema se desarrolla con mayor detalle en la sección 8.7 donde se propone una solución recursiva.

En este ejemplo se desea diseñar un algoritmo que genere *n* primeros términos de esta serie.

Para solucionar este ejercicio haciendo uso de una estructura iterativa es necesario, en primera instancia, determinar la cantidad de números a generar o sea conocer el valor de *n*; en segunda, y dado que cada número se obtiene por la suma de los dos términos anteriores, se utiliza tres variables: una para el valor generado y dos más para mantener en memoria los dos últimos datos, adicionalmente se define una variable para controlar el ciclo. La solución se presenta mediante un diagrama N-S en la figura 77.

Figura 77. Diagrama N-S para Serie Fibonacci



Para mirar el comportamiento de las variables en el cuadro 68 se presenta los resultados de la verificación del algoritmo, obsérvese como cambian los valores de una variable a otra.

Cuadro 68. Verificación del algoritmo serie Fibonacci

Iteración	n	Con	A	B	f	Salida
	10	0	0	1	0	
1		1	1	0	1	0
2		2	0	1	1	1
3		3	1	1	2	1
4		4	1	2	3	2
5		5	2	3	5	3
6		6	3	5	8	5
7		7	5	8	13	8
8		8	8	13	21	13
9		9	13	21	34	21
10		10	21	34	55	34

Ejemplo 37. Máximo común divisor

Dados dos números enteros, se requiere encontrar el máximo común divisor de los mismos.

El máximo común divisor (MCD) entre dos números es el número más grande que los divide a ambos, incluso puede ser uno de ellos dado que todo número es divisor de sí mismo. El MCD se puede obtener de forma iterativa o recursiva, en esta sección se soluciona aplicando la primera y en la sección 8.7 se presenta la solución recursiva aplicando el algoritmo de Euclides.

Al abordar este problema, seguramente, la primera idea que se le ocurre al lector es descomponer los números en sus divisores y tomar los comunes, tal como se lo enseñaron en el colegio. Esa es una solución válida; sin embargo, es difícil de implementar de forma algorítmica.

Aplicando el algoritmo de Euclides, el MCD se obtiene de la siguiente manera: se divide el primer número sobre el segundo, si la división es exacta (residuo = 0) el MCD es el segundo número, si la división no es exacta se divide el divisor sobre el residuo y si el módulo es cero el MCD es el número que se colocó como divisor; en caso contrario, se repite esta operación hasta obtener una división entera.

Por ejemplo, se busca el MCD de los números 12 y 8

```
12 | 8
4  | 1
```

Como la división es inexacta se efectúa una segunda división tomando como dividendo el anterior divisor y como divisor el residuo.

```
 8 | 4
0  | 2
```

Esta segunda operación es una división exacta (residuo = 0) por tanto el divisor es la solución del ejercicio; es decir, el MCD entre 12 y 8 es 4.

Con base en el ejemplo anterior se puede organizar los datos del ejercicio así:

Datos de entrada: a, b (dos números enteros)
Datos de salida: MCD
Proceso: c = a Mod b

La solución se presenta mediante pseudocódigo en el cuadro 69

Cuadro 69. Pseudocódigo del algoritmo Máximo Común Divisor

```
1.  Inicio
2.     Entero:  num1, num2, a, b, c
3.     Leer num1, num2
4.     a = num1
5.     b = num2
6.     Hacer
7.          c = a Mod b
8.          a = b
9.          b = c
10     Mientras c != 0
11     Escribir "MCD:", a
12  Fin algoritmo
```

La verificación de este algoritmo con los números 12 y 8, 6 y 20 genera los datos que se presentan en el cuadro 70.

Cuadro 70. Verificación del algoritmo Máximo Común Divisor

Ejecución	Iteración	a	b	c	Salida
1		12	8		
1	1	12	8	4	
1	2	8	4	0	
1		4	0		MCD: 4
2		6	20		
2	1	6	20	6	
2	2	20	6	2	
2	3	6	2	0	
		2	0		MCD: 2

Ejemplo 38. Invertir dígitos

Dado un número entero, se desea invertir el orden de sus dígitos. Supóngase que se toma el número 123 al invertirlo se obtiene el número 321.

Para cambiar el orden de los dígitos es necesario separarlos comenzando por el último, para ello se divide sobre 10 y se toma el residuo utilizando el operador módulo.

```
123 | 10
 3  | 12
```

Esta operación es equivalente a la expresión: 123 Mod 10 = 3

De esta manera se obtiene el dígito de la última posición del número original, el 3, que será el primero en el número invertido; luego se obtiene el siguiente dígito, el 2, dividiendo el cociente sobre 10, y así sucesivamente.

```
 12 | 10
 2  | 1
```

En la medida en que se extraen los dígitos del número original se van organizando de izquierda a derecha para conformar el nuevo número, para ello es necesario multiplicar por 10 el número que va generando y sumarle el último dígito obtenido, así:

3 * 10 + 2 = 32

Luego se obtiene el tercero y último número, el 1, y se realiza la operación:

$$\begin{array}{r|l} 1 & 10 \\ \hline 1 & 0 \end{array}$$

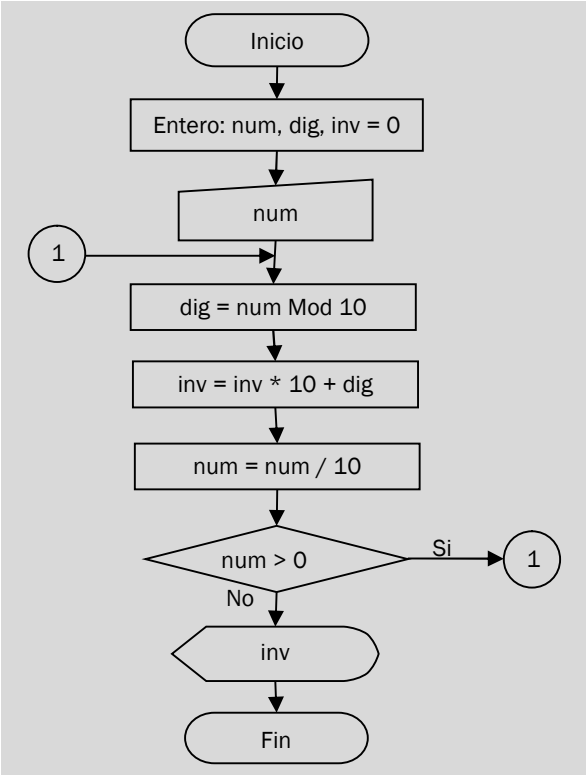
32 * 10 + 1 = 321

Las operaciones se repiten hasta que el cociente de la división sea 0, en cuyo caso se habrán invertido todos los dígitos. En este orden de ideas, se tienen los siguientes datos

Datos de entrada: número
Datos de salida: número invertido
Procesos:
- Dígito = número Mod 10
- Número = número / 10
- Número invertido = número invertido * 10 + digito

El diagrama de flujo se presenta en la figura 78.

Figura 78. Diagrama de flujo para invertir los dígitos de un número



En este algoritmo se utiliza la estructura iterativa *hacer – mientras*, como en la notación de diagrama de flujo no se cuenta con un símbolo para esta estructura se modela mediante una decisión y un conector que lleva el control de la ejecución hasta el primer proceso que se repite.

Los datos obtenidos en la verificación paso a paso del algoritmo se muestran en el cuadro 71.

Cuadro 71. Verificación del algoritmo invertir dígitos de un número

Iteración	num	dig	inv	Salida
	123			
1	123	3	3	
2	12	2	32	
3	1	1	321	
	0			321

Ejemplo 39. Número perfecto

Un número es perfecto si se cumple que la sumatoria de los divisores menores al número da como resultado el mismo número. Se desea un algoritmo para determinar si un número *n* es perfecto.

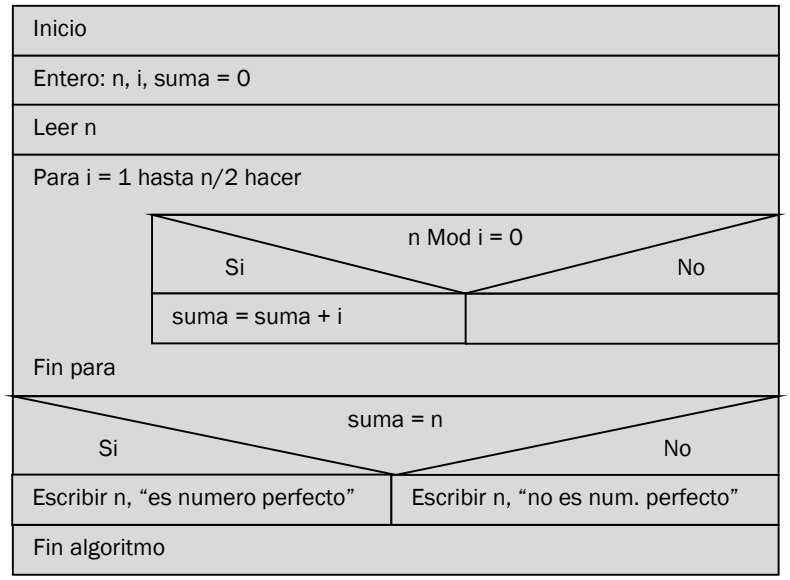
Para comprender mejor el concepto de número perfecto considérese dos casos: 6 y 8. Los divisores de 6 son: 1, 2, 3 y 6; y los divisores de 8 son: 1, 2, 4 y 8 (todo número es divisor de sí mismo). Si se suman únicamente los divisores menores a cada número se tiene:

$$\sum_{D_6 = 1}^{D_6 < 6} D_6 = 1 + 2 + 3 = 6 \qquad \sum_{D_8 = 1}^{D_8 < 8} D_8 = 1 + 2 + 4 = 7$$

Léase D_n como divisor de *n*. Por tanto, se tiene que: la sumatoria de los divisores de 6, menores a 6, es igual a 6, de lo que se concluye que 6 es número perfecto; con respecto al segundo número, se lee: la sumatoria de los divisores de 8, menores a 8, es igual a 7, en consecuencia 8 no es número perfecto.

Ahora, para saber si un número es perfecto se requieren tres procesos básicos: identificar los divisores, sumarlos y verificar si la sumatoria es igual al número. Para identificar los divisores menores a *n* es preciso hacer un recorrido entre 1 y *n*/2 y verificar para cada número si es o no es divisor de *n*, para esto se utiliza una estructura iterativa; para sumarlos se utiliza una variable de tipo acumulador que se actualiza dentro del ciclo; y finalmente, una estructura de decisión ubicada fuera del ciclo determinará si el número es o no es perfecto. La solución a este ejercicio se presenta en la figura 79.

Figura 79. Diagrama N-S para número perfecto



En el cuadro 72 se presenta los resultados de la verificación de este algoritmo con los números 6 y 8.

Cuadro 72. Verificación del algoritmo Número perfecto

Ejecución	n	i	suma	Salida
1	6		0	
1		1	1	
1		2	3	
1		3	6	6 es número perfecto
2	8		0	
2		1	1	
2		2	3	
2		3	3	
		4	7	8 no es número perfecto

Ejemplo 40. Potenciación iterativa

Dados dos números enteros: *b* que es la base y *e* que es el exponente, se requiere calcular el resultado de la potenciación.

La potenciación es una operación matemática cuyo resultado es el producto de multiplicar la base por sí misma tantas veces como indique el exponente. Entre sus propiedades se tienen: si el exponente es 0, el resultado es 1 y si el exponente es 1, el resultado es el mismo

valor de la base. En este ejercicio, para facilitar la solución, se limita el exponente a los números enteros positivos. Por ejemplo:

$2^3 = 2 * 2 * 2 = 8$
$3^5 = 3 * 3 * 3 * 3 * 3 = 243$

De manera que para desarrollar una potenciación se debe implementar una estructura iterativa y dentro de ella realizar multiplicaciones sucesivas de la base. El exponente indica el número de iteraciones a realizar. Los datos son:
Datos de entrada: base, exponente
Datos de salida: resultado
Proceso: producto = producto * base

En el cuadro 73 se presenta el pseudocódigo de este ejercicio.

Cuadro 73. Pseudocódigo Potencia iterativa

```
1.  Inicio
2.      Entero: b, e, p = 1, con
3.      Leer b, e
4.      Para con = 1 hasta e hacer
5.          p = p * b
6.      Fin para
7.      Escribir p
8.  Fin algoritmo
```

Cualquier número multiplicado por 0 da como resultado 0; por esto, la variable *p* (producto), que es un acumulador de resultados de multiplicación, se inicializa en 1, pues si se inicializara en 0 al final se tendría como resultado el 0. Los resultados de la verificación de este algoritmo se presentan en el cuadro 74.

Cuadro 74. Verificación del algoritmo potencia iterativa

Ejecución	Iteración	b	e	Con	P	Salida
1					1	
1	1	2	4	1	2	
1	2			2	4	
1	3			3	8	
1	4			4	16	16
2		3	5		1	
2	1			1	3	
2	2			2	9	
2	3			3	27	
2	4			4	81	
	5			5	243	243

Ejemplo 41. Número primo

Se denomina primo al número entero que solo cuenta con dos divisores: el uno y el mismo número. Se solicita un algoritmo que dado un número *n* decida si es o no es primo.

Para determinar si un número cumple con esta propiedad se procede a verificar si tiene algún divisor diferente de uno y el mismo número, en cuyo caso se demuestra que no es primo, en caso contrario se dice que si es primo; pero no es necesario examinar todos los números, para saber si es o no primo un número basta con buscar divisores entre dos y la raíz cuadrada del número[††].

Como ejemplos se examinan los números 9 y 19. La raíz cuadrada de 9 es 3, por tanto se busca divisores entre 2 y 3 y se encuentra que el 3 es divisor, por tanto 9 no es número primo. Para 19 se tiene que la raíz cuadrada entera es 4, por tanto se busca divisores entre 2 y 4, obteniéndose que los números 2, 3 y 4 ninguno es divisor de 19, por tanto se concluye que 19 es número primo. Podría verificarse para los números que siguen hasta 18, pero el resultado será el mismo.

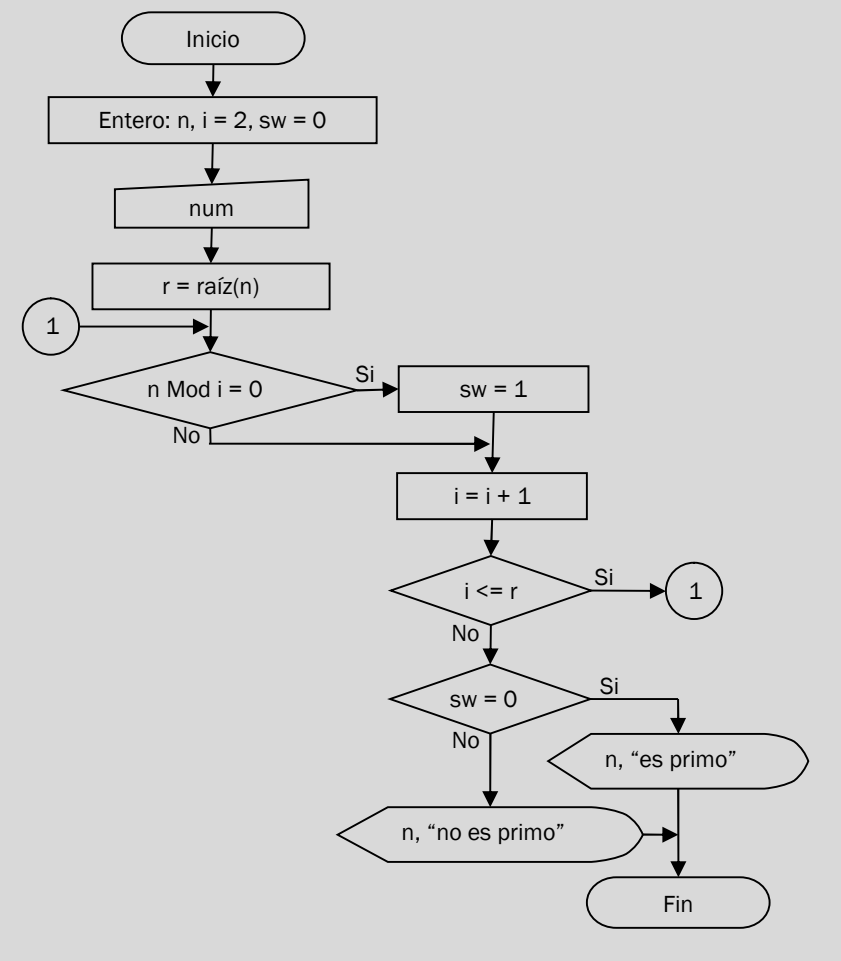
Para determinar si un número es primo se tienen los siguientes datos y operaciones:

Datos de entrada: número
Datos de salida: mensaje “número primo” ó “número no primo”
Proceso: número Mod i donde i: 2 ... raíz(número)

La solución algorítmica se presenta en la figura 80 en notación de diagrama de flujo.

[††] El propósito de este ejercicio es aplicar una estructura iterativa en la solución, por ello se procede a buscar divisores entre todos los números comprendidos entre 2 y la raíz entera del número. Existe otro método más rápido para comprobar si un número es primo, se trata de la solución propuesta por Eratóstenes (matemático griego del siglo III a. C) quien propone que para verificar si un número es primo solo hay que dividir para: 2, 3, 5 y 7. (Gran Enciclopedia Espasa, 2005).

Figura 80. Diagrama de flujo para número primo



En este algoritmo se han declarado cuatro variables: *n, r, i, sw*; *n* para el número, *r* para calcular la raíz cuadrada de *n*, *i* para hacer el recorrido entre 2 y *r* y *sw*. Esta última es una variable tipo conmu-tador o bandera, su propósito es informar si al hacer las iteraciones para la variable *i* se encontró algún divisor de *n*, la variable se inicializa en 0, si al finalizar el ciclo se mantiene en 0 significa que no se encontró ningún divisor y por tanto el número es primo, pero si la variable ha tomado el valor 1 significa que hay al menos un divisor y por tanto el número no es primo.

En el cuadro 75 se muestra los resultados de la ejecución paso a paso para los números 9 y 19.

Cuadro 75. Verificación del algoritmo Número primo

Ejecución	Iteración	n	r	i	sw	Salida
1		9	3		0	
	1			2		
	2			3	1	9 no es primo
2		19	4		0	
	1			2		
	2			3		
	3			4		19 es primo

Ejemplo 42. Puntos de una línea

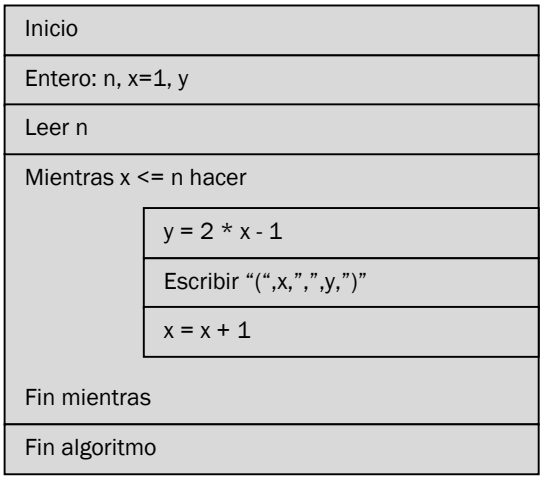
Dada la ecuación lineal *y = 2x – 1* se desea un algoritmo para calcular *n* puntos por los que pasa la línea a partir de *x = 1*.

Si se toma *n* = 4, los puntos serían:

X = 1 → y = 2(1) – 1 = 1 → (1,1)
X = 2 → y = 2(2) – 1 = 3 → (2,3)
X = 3 → y = 2(3) – 1 = 5 → (3,5)
X = 4 → y = 2(4) – 1 = 7 → (4,7)

En este caso los datos de entrada corresponde a *n*, los datos de salida a los puntos (*x,y*), tantos como indique *n* y el proceso se limita al desarrollo de la ecuación reemplazando *x* por el valor que corresponda. Para ello se utiliza una estructura iterativa, como se aprecia en el diagrama N-S de la figura 81.

Figura 81. Diagrama N-S para Puntos de una línea



Los datos generados al verificar la corrección de este algoritmo se presentan en el cuadro 76.

Cuadro 76. Verificación del algoritmo Número primo

Ejecución	N	x	y	Salida
1	4	1	1	(1,1)
1		2	3	(2,3)
1		3	5	(3,5)
		4	7	(4,7)

Ejemplo 43. Raíz cuadrada

Calcular la raíz cuadrada entera de un número.

Se sabe que la raíz cuadrada de un número es otro número que al elevarse al cuadrado es igual al primero.

$\sqrt{4} = 2 \quad \rightarrow 4 = 2^2$

$\sqrt{25} = 5 \quad \rightarrow 25 = 5^2$

Algunos números, como los anteriores, tienen una raíz cuadrada exacta, entera; mientras que para otros es un número real, como la raíz de 10 y para los números negativos la raíz es un número complejo o imaginario.

$\sqrt{10} = 3{,}16227766$

La raíz entera de un número, en el caso de los números que no tienen una raíz exacta, se obtiene truncando la parte decimal, por ejemplo, la raíz entera de 10 es 3.

En este ejercicio se tiene: dato de entrada es el número, dato de salida es la raíz y el proceso es calcular el cuadrado de los números, desde 1 hasta la raíz cuadrada del número. Se sabe que se ha encontrado la raíz entera cuando el siguiente número al elevarse al cuadrado da un valor superior al número ingresado. La solución a este ejercicio se presenta en notación de pseudocódigo en el cuadro 77.

Cuadro 77. Pseudocódigo del algoritmo raíz cuadrada

```
1  Inicio
2      Entero: n, i = 0, c
3      Leer n
4      Hacer
5          i = i + 1
6          c = (i+1) * (i+1)
7      Mientras (c <= n)
8      Escribir “La raíz de”, n, “es”, i
9  Fin algoritmo
```

En el cuadro 78 se presenta los resultados de la ejecución paso a paso de este algoritmo.

Cuadro 78. Verificación del algoritmo raíz cuadrada

Ejecución	N	I	C	Salida
1	25	0		
1		1	4	
1		2	9	
1		3	16	
1		4	25	
1		5	36	La raíz de 25 es 5
2	10	0		
2		1	4	
2		2	6	
2		3	16	La raíz de 10 es 3

4.3.7 Ejercicios propuestos

Diseñar los algoritmos para solucionar los problemas que se plantean a continuación. Se sugiere intercalar notación de pseudocódigo, diagrama de flujo y diagrama N-S; y de igual manera, las tres estructuras iterativas estudiadas en este capítulo.

1. Un estudiante cuenta con siete notas parciales en el curso de Introducción a la programación, se requiere un algoritmo para calcular el promedio de dichas notas.

2. Diseñar un algoritmo para leer números enteros hasta que se introduzca el 0. Calcular el cuadrado de los números negativos y el cubo de los positivos.

3. Diseñar un algoritmo para ingresar números, tantos como el usuario desee. Al finalizar el ciclo reportar cuántos números pares y cuántos impares se registraron, cuánto suman los pares y cuánto los impares.

4. El director de la escuela Buena Nota desea conocer el promedio de edad de sus estudiantes en cada grado. La escuela ofrece educación desde transición a quinto de primaria y cuenta con un grupo de estudiantes en cada grado. Diseñar un algoritmo que lea la edad y el grado de cada estudiante de la escuela y genere el correspondiente informe.

5. Un entrenador le ha propuesto a un atleta recorrer una ruta de cinco kilómetros durante 10 días, para determinar si es apto para la prueba de 5 kilómetros. Para considerarlo apto debe cumplir las siguientes condiciones:

   - Que en ninguna de las pruebas haga un tiempo mayor a 20 minutos.
   - Que al menos en una de las pruebas realice un tiempo menor de 15 minutos.
   - Que su promedio sea menor o igual a 18 minutos.

   Diseñar un algoritmo para registrar los datos y decidir si es apto para la competencia.

6. Una compañía de seguros tiene contratados a *n* vendedores. Cada vendedor recibe un sueldo base y un 10% extra por comisiones de sus ventas. Se requiere un algoritmo para calcular el valor a pagar a cada empleado y los totales a pagar por concepto de sueldos y comisiones.

7. Elaborar un algoritmo para procesar las notas definitivas de Biología para un grupo de *n* estudiantes. Se desea conocer el promedio del grupo, clasificar a los estudiantes en: excelentes, buenos, regulares y descuidados, según la nota obtenida y contar cuántos pertenecen a cada categoría. La escala es:

   Nota >= 4.8: excelente
   4.0 <= nota <= 4.7: bueno
   3.0 <= nota <= 3.9: regular
   nota <= 2.9: descuidado

8. Diseñar un algoritmo que genere el n-ésimo número de la serie Fibonacci.

9. Se aplicó una encuesta a *n* personas solicitando su opinión sobre el tema del servicio militar obligatorio para las mujeres. Las opciones de respuesta fueron: a favor, en contra y no responde. Se solicita un algoritmo que calcule qué porcentaje de los encuestados marcó cada una de las respuestas.

10. Se requiere un algoritmo que mediante un menú cumpla las funciones de una calculadora: suma, resta, multiplicación, división, potenciación y porcentaje. El menú contará con la opción apagar para terminar la ejecución del algoritmo.

11. Se requiere un algoritmo para facturar una venta con *n* artículos. En cantidades mayores a 10 unidades de un mismo artículo se aplicará un descuento del 5%.

12. Diseñar un algoritmo que lea un número entero y sume los dígitos que lo componen.

13. En el programa Ingeniería de Sistemas se necesita un algoritmo para conocer los porcentajes de estudiantes que trabajan y de los que se dedican únicamente a sus estudios, discriminados por género.

14. El almacén Buena Ropa cuenta con los registros mensuales de ventas y desea un algoritmo para determinar: en qué més se tuvo las ventas más altas, en cuál las más bajas y el promedio mensual de ventas.

15. Diseñar un algoritmo para calcular el factorial de un número *n*.

16. Se requiere un algoritmo para encontrar los números primos existentes entre 1 y *n*

17. Dado un grupo de 20 estudiantes que cursaron la materia Algoritmos, se desea saber cuál es el promedio del grupo, cuál fue la nota más alta y cuál la más baja, cuántos aprobaron el curso y cuántos reprobaron.

18. Dada la ecuación $y = 2x^2+3x - 4$ calcular los puntos por los que pasa la parábola en el intervalo -5, 5.

19. Diseñar un algoritmo para encontrar los primeros *n* números perfectos.

20. Diseñar un algoritmo para validar una nota. La nota debe ser un valor real entre 0 y 5.0. Si se ingresa un valor fuera de este intervalo se vuelve a leer, hasta que se ingrese una nota válida.

21. Diseñar un algoritmo para validar una fecha en el formato dd/mm/aaaa. Se considera que una fecha es válida si está entre 01/01/1900 y 31/12/2100, el mes entre 1 y 12 y el día entre 1 y 31, teniendo en cuenta que algunos meses tienen 28 (o 29), 30 o 31 días. Si la fecha no es válida se muestra un mensaje de error y se vuelve a leer los datos. El algoritmo termina cuando la fecha ingresada es válida.