

Roy Aguilera Jinesta  
Ana Gabriela Bejarano Salazar

## **Fundamentos de programación**

*Algoritmos, diagramas de flujo  
y pseudocódigo*

Versión preliminar



**UNED**

UNIVERSIDAD ESTATAL A DISTANCIA

Institución Benemérita de la Educación y la Cultura



Productora académica

*Kay Guillén Díaz*

Encargados de cátedra

*Ana Gabriela Bejarano Salazar*

*Carlos Andrés Morales Granados*

Especialista en contenidos

*Suhany Chavarría Artavia*

Corrección de estilo

*Vanessa Villalobos Rodríguez*

Figuras

Todas la imágenes e íconos utilizados en esta obra son de uso libre o de dominio público. Fueron tomadas de Canva con licencia Canva Pro y de Icon-Icons.com.

Esta unidad didáctica se confeccionó en la UNED, en el 2021, para utilizarse en las asignaturas Introducción a la Lógica, código 2109, del Bachillerato en Informática Educativa de la Escuela de Ciencias de la Educación; y Lógica para Computación, código 3071, del Diplomado en Informática de la Escuela de Ciencias Exactas y Naturales.

Universidad Estatal a Distancia

Vicerrectoría Académica

Escuela de Ciencias de la Educación

Escuela de Ciencias Exactas y Naturales



## **Presentación**

Estimada persona estudiante:

Este material educativo expone una temática que es la base del conocimiento para los primeros pasos en la programación. La lógica que se emplee para programar, o crear aplicaciones informáticas, trasciende niveles educativos, ya que muchos planes de estudio incluyen en su perfil alguna asignatura que involucre la aplicación de los conceptos lógico matemáticos para la resolución de problemas.

Dicho estudio de la lógica se lleva a cabo no solo a nivel universitario, sino también en educación primaria y secundaria. Inclusive, en carreras sin tendencia tecnológica, se dedica una pequeña parte del estudio al aprendizaje de los contenidos detallados en este material, pues la lógica fomenta el análisis de los problemas, ayuda a identificar relaciones de causa y efecto, potencia el pensamiento abstracto y facilita la visualización de soluciones. Todas esas acciones se pueden orientar a cualquier ámbito de trabajo, ya que, en todas las ciencias, se debe resolver algún problema.

Así pues, esas capacidades de pensamiento para analizar y resolver problemas de manera lógica se tornan trascendentales en carreras cuyo énfasis es la tecnología informática, por ejemplo: Ingeniería Informática o Informática Educativa.

Sin importar el programa, el lenguaje o el paradigma de programación que se utilice, los fundamentos de la lógica de programación son los mismos y forman la piedra angular donde se soporta la creación de programas que debemos tener claros, ya sea para aplicarlos en una organización o para explicarlos a otras personas.

Así que, si su interés es construir aplicaciones web o de escritorio para empresas o personales, hacer videojuegos, crear programas educativos, programar robots, generar animaciones, enseñar a programar, estimular el pensamiento lógico o simplemente conocer acerca del maravilloso mundo de la programación, este material educativo es ideal para dar los primeros pasos en firme hacia su nuevo reto de aprendizaje.

Esta unidad didáctica se compone de cinco capítulos. Cada uno de ellos brinda información acerca de un tema específico y se complementan con explicaciones y ejemplos claros fortalecidos con ejercicios de autoevaluación y sus respectivas respuestas.

El primer capítulo, llamado "Historia de la computación y la informática", se enfoca en la historia y el desarrollo estas ciencias, en las cuales se describen los principales avances tecnológicos y se brinda información complementaria de las mujeres y de los hombres que han contribuido a su desarrollo y que sentaron las bases de la tecnología que disfrutamos hoy, así como las innovaciones que surgen cada día.

El segundo capítulo se orienta al ámbito matemático y se denomina “Sistemas numéricos, operadores y precedencia”. En él, se explican cuatro sistemas numéricos: decimal, binario, octal y hexadecimal, y los métodos de conversión entre ellos. Luego, se destacan diversos tipos de operadores tanto matemáticos como relacionales y lógicos, detallando, de manera clara, su representación, funcionamiento y su orden jerárquico al momento de resolver expresiones.

El tercer capítulo, titulado “Resolución de algoritmos con diagramas de flujo y pseudocódigo”, es el más extenso y se adentra en los fundamentos de la lógica para programación. Se inicia con la conceptualización de algoritmo, sus características, estructura y formas de representarlo. Después, se explican detalladamente las variables y las constantes, considerando los tipos de datos, la forma en que se deben nombrar y su manipulación en los programas informáticos. Una vez que se estudian las variables y las constantes, se abordan las diversas estructuras de decisión y de repetición que se emplean al programar.

Las primeras —las estructuras de decisión— sirven para tomar decisiones, es decir, si el programa hace una u otra acción, y se conocen como los SI (Si, Si-Sino, Si anidado, Según); mientras que las segundas, se denominan ciclos (Mientras, Repetir y Para), y se emplean para repetir un grupo de instrucciones dependiendo del valor de una expresión.

El cuarto capítulo, identificado como “Arreglos unidimensionales y multidimensionales”, aborda el concepto de arreglos, los cuales son estructuras en la memoria de la computadora para almacenar datos temporalmente. En esta sección, se define qué es un arreglo, se caracterizan dos tipos (vectores

y matrices), se explica su manejo en la programación y su relación con las estructuras de repetición.

El quinto y último capítulo se llama “Subprocesos” y trata acerca de los subprocesos o subprogramas. Este concepto es muy utilizado en la programación y propone que un problema es más sencillo de resolver si se divide en partes más pequeñas y menos complejas. Esto se aplica a los programas informáticos al construirlos en secciones simples, que se enfocan en realizar una sola tarea, pero que se vinculan entre sí para formar un todo. En el capítulo, se definen y se caracterizan los dos tipos de subprocesos que existen: los procedimientos y las funciones. También, se explica su manejo en la programación y los detalles fundamentales en cuanto al envío de datos entre subprocesos.

Antes de iniciar su aventura de estudio de este texto, es recomendable que domine algunos conceptos matemáticos básicos, por ejemplo:

- Operaciones aritméticas: suma, resta, multiplicación, división, potenciación.
- Operadores matemáticos y relacionales ( $>$ ,  $<$ ,  $=$ ,  $>=$ ,  $<=$ ,  $<>$ ), tanto su representación como su concepto.
- Cálculo de sumatorias, promedios, porcentajes, fracciones y regla de tres.

Esos y otros conceptos útiles se pueden estudiar en asignaturas que se ofrecen en los primeros años de estudio de la carrera. Esto facilitará la comprensión del material con lo cual podrá obtener un mayor provecho de su proceso de aprendizaje. Sin embargo, el abordaje de los temas del presente

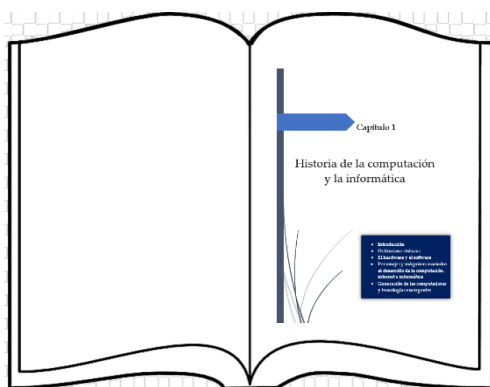
trabajo tiene explicaciones claras de los conceptos que serán fundamentales para la comprensión de todo el contenido.

Al finalizar el estudio de este material educativo, usted deberá ser capaz de construir algoritmos secuenciales en forma de pseudocódigos o de diagramas de flujo. También, se espera que pueda definir e identificar, con total claridad, la estructura de un algoritmo y posibles errores lógicos; para luego, proponer soluciones eficientes basadas en la lógica de programación. Además, deberá implementar, de manera eficiente, estructuras de decisión y de repetición en la construcción de algoritmos con una arquitectura que involucre subprocesos.

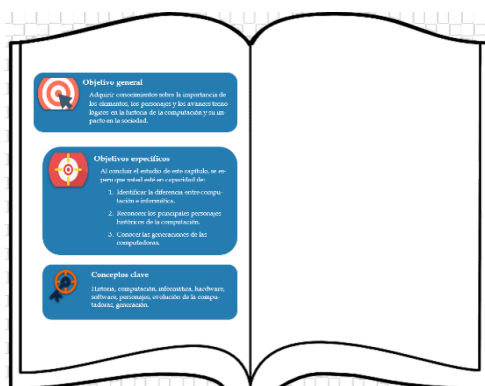
## **¿Cómo utilizar este libro?**

Con el fin de facilitar el proceso de autoaprendizaje de los temas presentados, se hace uso de una serie de componentes didácticos que le permiten al estudiante identificar algunos elementos importantes.

En la ventana de cada capítulo (la primera página), se presenta el título y el número del capítulo, además de un sumario en un recuadro. Este último consiste en los títulos principales que conforman el capítulo.



En el reverso, se encuentran los objetivos de aprendizaje que el estudiante debe alcanzar al finalizar el estudio del capítulo y se dividen en objetivo general y en objetivos específicos. También, se presenta una lista de conceptos clave.



Cada capítulo inicia con una pequeña introducción, así como una breve guía de lectura sobre su ordenamiento y que se encuentra señalado por el siguiente icono.

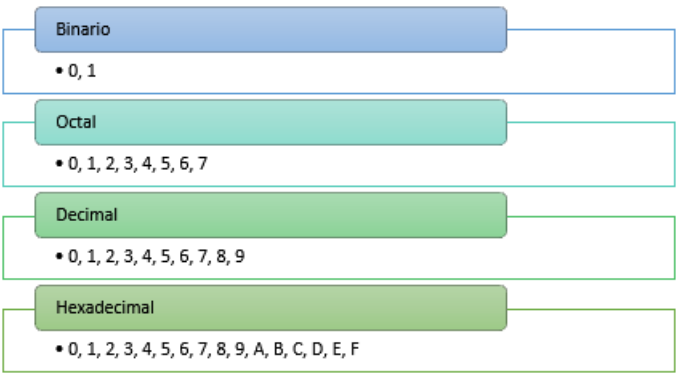




Tanto las figuras como los cuadros que se utilizan en el libro contienen una leyenda con un número y la descripción. El número está compuesto por dos números: el primero identifica el capítulo en el que se encuentra y el segundo es un consecutivo. La leyenda está al pie, en las figuras y en los cuadros en la parte superior.

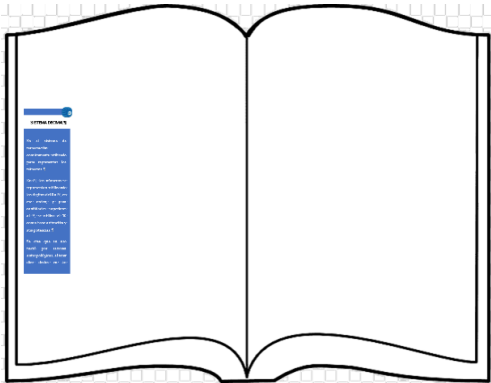
**Cuadro 2.1.**  
Operadores aritméticos, su símbolo y forma de uso

Nombre	Símbolo	Sintaxis
Exponenciación	$\wedge$	$3 \wedge 2$
Multiplicación	$\times$	$3 \times 2$
División	$/$	$3 / 2$
Módulo	$\%$ o también MOD	$3 \% 2$ o $3 \text{ MOD } 2$
Suma	$+$	$3 + 2$
Resta	$-$	$3 - 2$




**FIGURA 2.7.** Componentes de los sistemas numéricos binario, octal, decimal y hexadecimal

Otro recurso que el lector encontrará es el cuadro al margen, en el cual encontrará la definición de un concepto importante que se utiliza en el texto y que está resaltado en negrita.




Como parte del proceso de enseñanza-aprendizaje guiado, se presentan a lo largo del texto algunos recuadros, titulados “Para saber más”, en los que se les motiva a investigar y a profundizar sobre algunos temas. Asimismo, proporciona lugares en internet donde podrá encontrar más información con sus respectivos enlaces y el código QR del sitio.


**Para saber más**

Observa los siguientes videos acerca de la historia de la computación y del Museo de Historia de la computación sobre los pioneros.

✓ *Historia de la computación*  
Míralo en <<https://www.uned.cr/qr/1lowe4uOg8>>, o escanea el siguiente código QR:




✓ *Computer Pioneers: Pioneer Computers Part 1*  
Míralo en <<https://www.uned.cr/qr/aspP5XQsl>> o escanea el siguiente código QR:



Los códigos QR (Quick Response Code o “código de respuesta rápida”), son un “atajo” para acceder a las direcciones en la red sugeridas. Debe disponer de conexión a internet, una tableta o teléfono inteligente (*smartphone*) y una *app* –o aplicación– para leer este tipo de imágenes. Por lo general, los sistemas operativos las incluyen, o bien, se pueden descargar gratuitamente de tiendas y de repositorios.


En ocasiones, es necesario resaltar algún dato o situación importante que no se debe obviar, y lo encontrará en un recuadro como este:



**Para tomar en cuenta**

Debido a que este documento explica en su prosa el funcionamiento de los algoritmos, no se hará este comentario en el pseudocódigo ni en los diagramas de flujo. Tampoco se utilizarán en cada variable, ya que se prioriza la simplicidad visual de las imágenes para evitar posibles distractores. Los comentarios que se pondrán se emplearán solo en los pseudocódigos, ya que la herramienta empleada para los diagramas de flujo (FreeDFD) no admite el uso de estos elementos.


En los casos en que es necesario llamar la atención sobre un tema por considerar, se encuentra el siguiente icono:



**Nota**

Podemos observar que, con solo un operador (interruptor) OR esté activado (T), el bombillo se encenderá (T).






O bien, si se desea que se tome un momento para reflexionar en lo que se ha leído, se plantea de la siguiente forma:



**Para reflexionar...**

- ¿Conocía el aporte de las mujeres a la historia de la computación y de la ciencia?
- ¿Qué opina del papel de la mujer actual en estas áreas?

Por último, se utilizan una serie de iconos en la columna auxiliar (columna que se deja libre en el margen externo de la hoja) para resaltar algunas secciones esenciales en el texto, que se muestran a continuación:

Icono	Descripción
	<p>Señala el inicio de los ejercicios de autoevaluación. Consiste en una serie de actividades que se le plantean con el fin de que medir su nivel de aprendizaje al finalizar la lectura del capítulo o del apartado.</p>
	<p>Aquí encontrará las respuestas de los ejercicios de autoevaluación. No haga trampa y no los revise hasta no haber intentado resolverlos usted mismo; solo así podrá saber qué puntos debe repasar antes de seguir con su estudio.</p>
	<p>Este ícono se encuentra en “Referencias” y contiene el nombre de los textos que se utilizan como respaldo teórico.</p>
	<p>Un apartado básico es el “Glosario”, en el cual localiza la definición de palabras que debe dominar en el área de la programación.</p>
	<p>“Lecturas recomendadas”, como su nombre lo indica, tiene una lista de texto, digitales o físicos, que se recomienda consultar.</p>

Esperamos que esta obra le ayude a alcanzar sus objetivos de estudio.

Atentamente, el equipo de producción.



## **¿Sabe qué es Promade? ¿Quiere colaborar en la mejora de este material didáctico?**

El libro que está en sus manos fue especialmente diseñado para usted. Un equipo multidisciplinario de profesionales veló por su calidad académica, gracias a un riguroso proceso de revisiones y a una mediación didáctica apropiada, de acuerdo con las necesidades propias de una persona que estudia en el sistema de educación a distancia.

En el Programa de Producción de Material Didáctico Escrito (Promade), se elaboran los materiales escritos que las asignaturas de la UNED requieren. Desde la fundación de la Universidad, en 1977, este departamento ha sido el eje de la producción de aquellos materiales que son el principal objeto de consumo didáctico de nuestros estudiantes, lo cual nos compromete a una producción intensa y permanente. Estas obras han llegado a constituir un acervo nacional e internacional: se utilizan con gran éxito en la UNED y, también, en diferentes instituciones educativas públicas y privadas de nivel superior y medio del país, así como fuera de Costa Rica.

Usted puede contribuir con el mejoramiento de los materiales que producimos, enviando sus observaciones y comentarios sobre la unidad didáctica al correo [infopromade@uned.ac.cr](mailto:infopromade@uned.ac.cr). Recuerde incluir el nombre del material

y del autor o autores. Y si el libro le gustó, también cuéntenos... ¡Nos encantaría conocer su experiencia!



<<https://www.uned.ac.cr/dpmd/promade>



## Contenido

5	Subprocesos .....	565
5.1	Introducción .....	567
5.2	Definición .....	569
5.2.1	Subprocesos en PSeInt .....	571
5.3	Características de los subprocesos .....	579
5.4	Llamada o invocación de un subproceso .....	579
5.5	Paso de parámetros .....	579
5.5.1	Parámetros por valor (copia) .....	581
5.5.1.1	Parámetros por valor en PSeInt .....	581
5.5.1.2	Parámetros por valor en DFD .....	583
5.5.2	Parámetros por referencia (original) .....	586
5.5.2.1	Parámetros por referencia en PSeInt .....	586

5.5.2.2	Parámetros por referencia en DFD .....	588
5.5.3	Variables globales .....	588
5.5.4	Variables locales.....	589
5.6	Tipos de subprocesos.....	589
5.6.1	Procedimientos en PSeInt.....	590
5.6.2	Procedimientos en DFD.....	592
5.6.3	Funciones en PSeInt.....	593
5.6.4	Funciones en DFD.....	594
5.7	Ejemplos de algoritmos con subprocesos.....	595
5.7.1	Ejemplo 1. Algoritmo para dividir dos números .....	596
5.7.2	Ejemplo 2. Venta de entradas al cine .....	600
5.7.3	Ejemplo 3. Cálculo del índice de masa corporal (IMC) .....	601
5.7.4	Ejemplo 4. Elevar un número a un exponente con el ciclo while 607	
5.7.5	Ejemplo 5. Calculadora geométrica .....	610
5.7.6	Ejemplo 6. Modificación de matriz.....	615
5.8	Ejercicios de autoevaluación.....	621
5.9	Respuesta a los ejercicios de autoevaluación .....	625
5.10	Referencias.....	644





## Índice de figuras

FIGURA 5.1. Esquema de un algoritmo principal.....	569
FIGURA 5.2. Esquema de un algoritmo con un subproceso .....	570
FIGURA 5.3 Pseudocódigo de división de números sin subprocesos.....	571
FIGURA 5.4. Pseudocódigo de división de números con subprocesos .....	572
FIGURA 5.5 Botón para crear un subproceso en PSeInt.....	573
FIGURA 5.6. Ejemplos de nombres de subprocesos en PSeInt .....	573
FIGURA 5.7. Botón para crear un subproceso en DFD.....	574
FIGURA 5.8. Estructura de un subproceso en DFD .....	574
FIGURA 5.9. Ventana de datos de un subproceso en DFD .....	575
FIGURA 5.10. Botones para navegar entre subprocesos y diagrama principal en DFD .....	575
FIGURA 5.11. Botón para agregar una llamada a un subproceso en DFD..	575
FIGURA 5.12. Símbolo de subproceso en DFD.....	576

<b>FIGURA 5.13.</b> Ventana de datos de llamada a subproceso en DFD.....	576
<b>FIGURA 5.14.</b> Diagrama de división de números con subprocesos en DFD .....	577
<b>FIGURA 5.15.</b> Detalle del subproceso en DFD que divide dos números...	578
<b>FIGURA 5.16.</b> Detalle del subproceso en DFD de validación de divisor ....	578
<b>FIGURA 5.17.</b> Nombres de parámetros en un algoritmo principal y un subproceso en PSeInt .....	580
<b>FIGURA 5.18.</b> Paso de parámetros por Valor en PSeInt .....	582
<b>FIGURA 5.19.</b> Salida de pantalla del tratamiento de paso de parámetros Por Valor en PSeInt .....	583
<b>FIGURA 5.20.</b> Paso de parámetros Por Valor en DFD .....	584
<b>FIGURA 5.21.</b> Salida de pantalla de parámetros Por Valor antes del subproceso en DFD .....	585
<b>FIGURA 5.22.</b> Salida de pantalla de parámetros Por Valor en el subproceso en DFD .....	585
<b>FIGURA 5.23.</b> Salida de pantalla de parámetros Por Valor luego del subproceso en DFD .....	586
<b>FIGURA 5.24.</b> Paso de parámetros Por Referencia en PSeInt .....	587
<b>FIGURA 5.25.</b> Salida de pantalla de tratamiento de parámetros Por referencia en PSeInt .....	588
<b>FIGURA 5.26.</b> Esquema de retorno de datos y diferencia entre procedimientos y funciones.....	590
<b>FIGURA 5.27.</b> Multiplicación de números usando un procedimiento en PSeInt .....	591
<b>Figura 5.28.</b> Pseudocódigo en PSeInt usando procedimientos para leer datos y multiplicarlos .....	592

FIGURA 5.29. Diagrama en DFD usando procedimientos para leer datos y multiplicarlos.....	592
FIGURA 5.30. Detalle de la estructura de una función en PSeInt .....	593
FIGURA 5.31. Pseudocódigo en PSeInt que usa una función para sumar dos números.....	593
FIGURA 5.32. Salida de pantalla del pseudocódigo que suma dos números con una función .....	594
FIGURA 5.33. Diagrama en DFD que suma dos números usando el equivalente a una función.....	595
FIGURA 5.34. Pseudocódigo con división de números y verificación de múltiplos de 3 y 5.....	596
FIGURA 5.35. Pseudocódigo con división de números y verificación de múltiplos de 3 y 5 con subprocesos.....	597
FIGURA 5.36. Diagrama con división de números y verificación de múltiplos de 3 y 5 con subprocesos.....	599
FIGURA 5.37. Pseudocódigo del cálculo de costo de entradas al cine sin subprocesos.....	600
FIGURA 5.38. Pseudocódigo cálculo de costo entradas al cine con subprocesos.....	601
FIGURA 5.39. Salida en pantalla del diagrama de flujo para determinar el IMC.....	605
FIGURA 5.40. Primera parte diagrama de flujo con subprocesos para determinar el IMC.....	605
FIGURA 5.41. Segunda parte diagrama de flujo con subprocesos para determinar el IMC.....	606
FIGURA 5.42. Pseudocódigo sin subprocesos para calcular potencias.....	608
FIGURA 5.43. Pseudocódigo con subprocesos para calcular potencias.....	608

<b>FIGURA 5.44.</b> Pseudocódigo con subprocesos modificado para calcular potencias .....	610
<b>FIGURA 5.45.</b> Diagrama principal de la calculadora geométrica .....	611
<b>FIGURA 5.46.</b> Diagrama se los subprocesos MuestraMenu y ValidaDato de la calculadora geométrica .....	612
<b>FIGURA 5.47.</b> Diagramas de los subprocesos AreaCuadrado y AreaRectangulo de la calculadora geométrica.....	613
<b>FIGURA 5.48.</b> Diagrama se los subprocesos AreaTriangulo y AreaCirculo de la calculadora geométrica.....	614
<b>FIGURA 5.49.</b> Pseudocódigo del algoritmo principal para modificar una matriz .....	616
<b>FIGURA 5.50.</b> Pseudocódigo del subproceso MuestraMenu algoritmo para modificar una matriz .....	616
<b>FIGURA 5.51.</b> Pseudocódigo del subproceso LlenaMatriz algoritmo para modificar una matriz .....	617
<b>FIGURA 5.52.</b> Pseudocódigo del subproceso MuestraMatriz algoritmo para modificar una matriz .....	618
<b>FIGURA 5.53.</b> Pseudocódigo del subproceso ModificaMatriz algoritmo para modificar una matriz .....	619
<b>FIGURA 5.54.</b> Pseudocódigo del subproceso Salida algoritmo para modificar una matriz .....	620
<b>FIGURA 5.55.</b> Pseudocódigo del subproceso Mensaje algoritmo para modificar una matriz .....	620
<b>FIGURA 5.56.</b> Pseudocódigo en PSeInt que lee un nombre digitado por el usuario; y, luego, muestra el saludo “Hola” y el nombre digitado .....	625

<b>FIGURA 5.57.</b> Procedimiento que muestra el comportamiento de dos variables al ser modificadas en un subproceso. Una de ellas enviada como parámetro Por Valor; y la otra, Por Referencia.....	626
<b>FIGURA 5.58.</b> DFD que lee el valor de dos variables y calcula su diferencia, asignando el resultado a una tercera variable e indica si el valor es positivo o negativo.....	627
<b>FIGURA 5.59.</b> Pseudocódigo en PSeInt que evalúa, a través de una función, si un número es múltiplo de 3.....	628
<b>FIGURA 5.60.</b> Pseudocódigo en PSeInt que solicita el tipo de cambio y convierte un monto en colones a dólares .....	629
<b>FIGURA 5.61.</b> Pseudocódigo en PSeInt que solicita el tipo de cambio y convierte un monto en colones a dólares, solicitando los datos a través de una instrucción de asignación.....	630
<b>FIGURA 5.62.</b> Diagrama de flujo en DFD que solicita el nombre de una persona y despliega un saludo.....	631
<b>FIGURA 5.63.</b> Diagrama de flujo principal en DFD que calcula el salario semana con base en una serie de parámetros .....	632
<b>FIGURA 5.64.</b> Diagrama de flujo de los subprocesos LecturaDatos y CalculoHorasExtra del algoritmo de cálculo de salario de la figura 5.63 .....	633
<b>FIGURA 5.65.</b> Diagrama de flujo de los subprocesos CalculoSalariBruto y CalculoDeducciones del algoritmo de cálculo de salario de la figura 5.63.....	634
<b>FIGURA 5.66.</b> Diagrama de flujo subprocesos CalculoSalarioNeto y MuestraInforme del algoritmo de cálculo de salario de la figura 5.63.....	635
<b>FIGURA 5.67.</b> Pseudocódigo principal en PSeInt del algoritmo búsqueda en un vector con números entre 100 y 500, uno indicado por el usuario ..	636

<b>FIGURA 5.68.</b> Subprocesos <code>LlenaVector</code> , <code>MuestraVector</code> y <code>SolicitaNumero</code> del algoritmo búsqueda en un vector de la figura 5.67 .....	637
<b>FIGURA 5.69.</b> Subprocesos <code>Busqueda</code> , <code>MuestraResultados</code> del algoritmo búsqueda en un vector de la figura 5.67 .....	638
<b>FIGURA 5.70.</b> Pseudocódigo principal de un algoritmo en <code>PSeInt</code> que calcula el promedio y encuentra el mayor y el menor número en una matriz .....	639
<b>FIGURA 5.71.</b> Procedimiento <code>LlenaMatriz</code> y función <code>SumatoriaMatriz</code> del algoritmo de la figura 5.70.....	640
<b>FIGURA 5.72.</b> Procedimiento <code>DeterminaMayMen</code> del algoritmo de la figura 5.70.....	640
<b>FIGURA 5.73.</b> Subprocesos <code>MuestraMatriz</code> y <code>MuestraInforme</code> algoritmo de trabajos con una matriz del algoritmo de la figura 5.70 .....	641
<b>FIGURA 5.74.</b> Pseudocódigo en <code>PSeInt</code> que intercambia los valores de dos celdas de un vector lleno de nombres .....	642
<b>FIGURA 5.75.</b> Procedimientos <code>LlenaVector</code> y <code>MuestraVector</code> del algoritmo de la figura 5.74 .....	642
<b>FIGURA 5.76.</b> Procedimientos <code>SolicitudCeldas</code> y <code>CambiaContenido</code> del algoritmo de la figura 5.74.....	643

# Índice de cuadros

Cuadro 5.1. Índice de masa corporal.....	602
------------------------------------------	-----

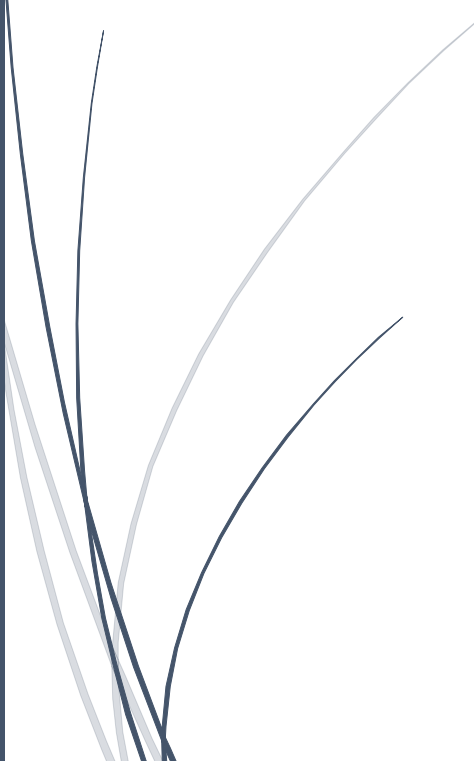




A thick dark blue vertical bar is on the left. A blue arrow points right from the bar, containing the chapter title.

## Capítulo 5

# Subprocesos

- 
- Several thin, curved, light blue lines originate from the bottom left and sweep upwards and to the right.
- Definición
  - Características de los subprocesos
  - Llamada o invocación de un subproceso
  - Paso de parámetros
  - Tipos de subprocesos



## Objetivo general

Aplicar subprocesos en la construcción de algoritmos para la resolución de problemas.



## Objetivos específicos

Al concluir el estudio de este capítulo, se espera que usted esté en capacidad de:

1. Identificar los tipos de subprocesos que se implementan en los algoritmos.
2. Identificar los tipos de pasos de parámetros.



## Conceptos clave

Subprocesos, llamada, llamada de un subproceso, invocación, invocación de un subproceso, parámetros por valor, parámetros por referencia, variables locales, variables globales, procedimientos, funciones.

## 5.1 Introducción

En esta sección del libro, se tratan los subprocesos y su implementación en los algoritmos en pseudocódigo y en diagrama de flujo empleando herramientas como `PSeInt` y `DFD` respectivamente.

En los capítulos anteriores se han desarrollado algoritmos en un solo bloque; pero en este, se harán ejemplos en los que se divide el algoritmo en pequeños segmentos llamados subprocesos. Para iniciar, se aborda el concepto de subprocesos, sus características, diferencias y tipo de pasos de parámetros.

Cada significado se apoya en ejemplos que evidencian la implementación de los subprocesos tanto `PSeInt` como en `DFD`; además, se analizan las particularidades de trabajar procedimientos y funciones en estas herramientas de programación.

Finalmente, se desarrollan ejemplos con explicación y ejercicios de autoevaluación con soluciones. En algunos casos, se toman las actividades de los capítulos previos, pero implementando subprocesos.



## Guía de lectura

Este capítulo se enfoca en la implementación de subprocesos en algoritmos en pseudocódigo y diagramas de flujo. Primero, se aclara el concepto de subprocesos y su esquema de trabajo en los algoritmos.

Luego, se reconocen los dos tipos de subprocesos que existen: procedimientos y funciones, así como sus diferencias. Después, se debe entender las particularidades de los tipos de pasos de parámetros: por valor y por referencia.

Por último, se analiza la implementación de subprocesos en algoritmos que originalmente no los tenían, de forma que se pueda comparar cada solución. Se recomienda prestar especial atención a estos desarrollos y a las soluciones de los ejercicios de autoevaluación.

## 5.2 Definición

Un **subproceso** se define como una sección de código que forma parte de un algoritmo principal, pero separado de este. Cuando se emplean subprocesos, el algoritmo principal se divide en partes; no obstante, estas partes se vinculan con el principal. Los subprocesos son simples en su funcionamiento y tienen como objetivo ejecutar una única tarea.

Si lo analizamos de forma gráfica, este es un algoritmo sin subprocesos:

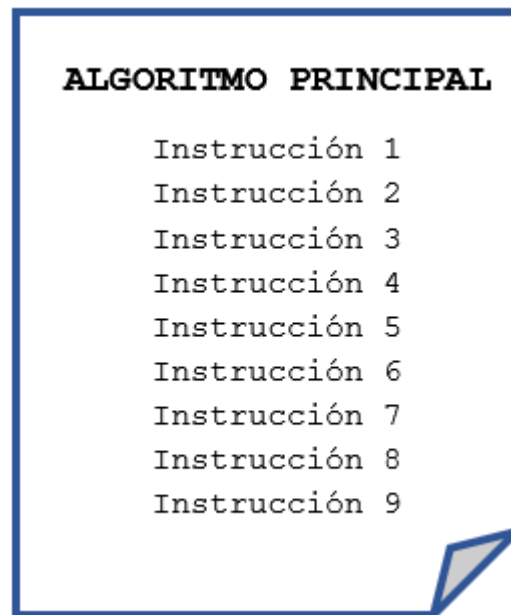


FIGURA 5.1. Esquema de un algoritmo principal

Nótese que todas las instrucciones están en un solo bloque de código. Sin embargo, si utilizamos subprocesos, puede que algunas instrucciones ya no formen parte del algoritmo principal, sino de uno nuevo que se encuentra relacionado con el principal; este nuevo algoritmo es un subproceso. De forma gráfica, podemos representarlo de la siguiente manera:

### Subproceso

Es una sección de código que forma parte de un algoritmo principal, pero separado de este. Tienen como objetivo realizar una única tarea.

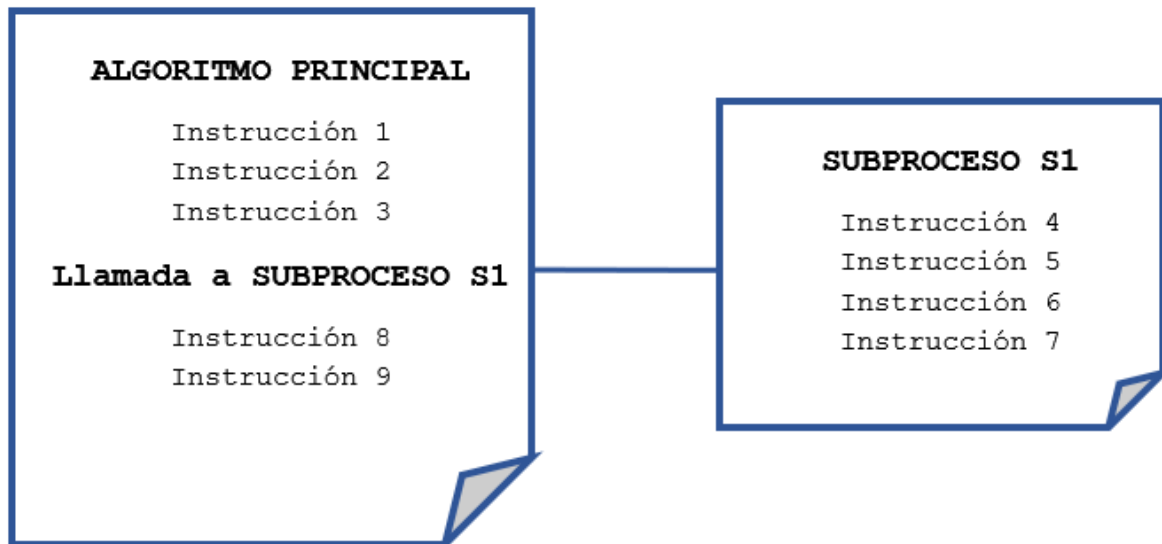


FIGURA 5.2. Esquema de un algoritmo con un subproceso

En la representación gráfica, podemos advertir cómo las instrucciones 4, 5, 6, y 7 no forman parte del algoritmo principal, sino que conforman el Subproceso S1. Debemos tener claro que el subproceso es un componente del algoritmo principal, pero no de su bloque de instrucciones.

En cuanto a la relación de ambas estructuras (el algoritmo principal y el subproceso), esta se establece por medio de una "llamada" o "invocación" al subproceso dentro el algoritmo principal. Veamos que luego de la línea 3 hay una instrucción que llama al Subproceso S1. En ese momento, el flujo de datos pasa a ejecutar las instrucciones del subproceso; y, una vez que las cumple, regresa a la instrucción 8 del algoritmo principal.

Los subprocesos son útiles porque, al dividir el algoritmo, lo descomponen en partes más simples y esto facilita su comprensión y su mantenimiento. Se destaca que, a mayor envergadura tenga el algoritmo, más necesaria y justificada será la implementación de los subprocesos.

Como aclaración, para el desarrollo de los ejemplos de subprocesos, se emplearán los algoritmos hechos en capítulos anteriores. Algunos de estos son cortos por lo que no muestran la utilidad a fondo de los subprocesos; sin embargo, son funcionales para explicar y ejemplificar su implementación, estructura y funcionamiento.

### 5.2.1 Subprocesos en PSeInt

A continuación, se comparan dos códigos en PseInt: uno no emplea subprocesos; y otro, sí. Veamos el primero:

```
1  Algoritmo Division
2  Definir Dividendo, Divisor, Cociente Como Real;
3  Dividendo=0;
4  Divisor=0;
5  Cociente=0;
6
7  Escribir "Digite el dividendo.";
8  Leer Dividendo;
9  Escribir "Digite el divisor.";
10 Leer Divisor;
11
12 Si Divisor=0 entonces
13     Escribir "Error, el divisor NO puede ser cero.";
14 FinSi
15
16 Si Divisor!=0 entonces
17     Cociente= Dividendo/Divisor;
18     Escribir "Cociente es igual a: ",Cociente;
19 FinSi
20
21 FinAlgoritmo
```

FIGURA 5.3 Pseudocódigo de división de números sin subprocesos

Ahora, se muestra el mismo algoritmo, pero se emplean subprocesos:

```
1  Algoritmo Division
2      Definir Dividendo, Divisor Como Real;
3      Dividendo=0;
4      Divisor=0;
5
6      Escribir "Digite el dividendo.";
7      Leer Dividendo;
8      Escribir "Digite el divisor.";
9      Leer Divisor;
10
11     ValidaDivisor(Divisor);
12     Divide(Dividendo,Divisor);
13
14 FinAlgoritmo
15
16
17
18 SubProceso ValidaDivisor(Dvsor)
19     Si Dvsor=0 entonces
20         Escribir "Error, el divisor NO puede ser cero.";
21     FinSi
22 FinSubProceso
23
24
25 SubProceso Divide(Dvendo,Dvsor)
26     Definir Cociente Como Real;
27     Cociente=0;
28     Si Dvsor!=0 entonces
29         Cociente= Dvendo/Dvsor;
30         Escribir "Cociente es igual a: ",Cociente;
31     FinSi
32 FinSubProceso
```

FIGURA 5.4. Pseudocódigo de división de números con subprocesos

Notemos que, en las líneas 11 y 12, se hace la llamada a los subprocesos. El subproceso `ValidaDivisor` inicia en la línea 18 y finaliza en la 22, mientras que el subproceso `Divide` inicia y finaliza en las líneas 25 y 32 respectivamente.



Veamos que cada subproceso tiene las mismas instrucciones que tenía el primer algoritmo. La única diferencia es la declaración de la variable `Cociente`. En el primer caso (figura 5.3), se declara en el algoritmo principal (que es el único); pero en el segundo ejemplo (figura 5.4), se declara en el subproceso `Divide`, ya que solo ahí se utilizará, por ende, no es necesario declararlo en el algoritmo principal. En cuanto al orden de los elementos, los subprocesos pueden estar antes o después del algoritmo principal. El sistema buscará primero este último y de ahí inicia la ejecución.

Tome en cuenta que `PseInt`, tanto el algoritmo principal como los subprocesos, se visualiza en la misma pantalla; además, el botón para insertar un subproceso se muestra en la figura 5.5.

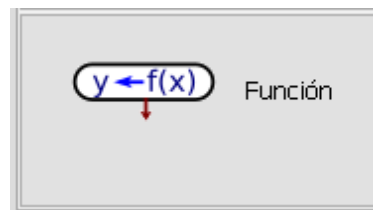


FIGURA 5.5 Botón para crear un subproceso en PSeInt

Al presionarlo, se generará un subproceso llamado de forma genérica `Función`, pero se recomienda cambiarla por `SubProceso` o por `SubAlgoritmo`, como se muestra en la figura 5.6:

```

Funcion Nombre ()
FinFuncion

SubAlgoritmo Nombre ()
FinSubAlgoritmo

SubProceso Nombre ()
FinSubProceso

```

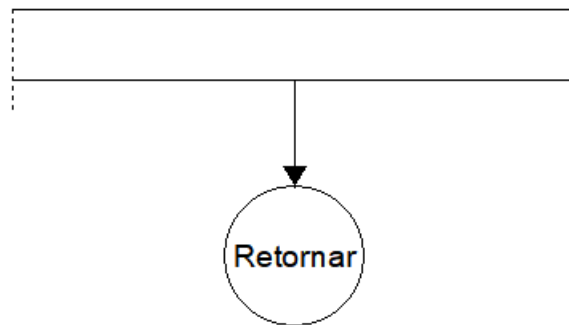
FIGURA 5.6. Ejemplos de nombres de subprocesos en PSeInt

Para los algoritmos en diagrama de flujo, el programa DFD organiza el algoritmo principal y los subprocesos en pantallas diferentes, pero en el mismo archivo. Si tenemos un diagrama de flujo y deseamos agregar un subproceso, empleamos el siguiente botón (figura 5.7):



**FIGURA 5.7.** Botón para crear un subproceso en DFD

Esto abrirá una nueva ventana con el inicio del diagrama para el subproceso, en la cual se trabajará la solución lógica para este, cuyo primer diagrama luce de la siguiente manera (figura 5.8):

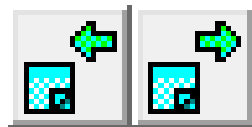


**FIGURA 5.8.** Estructura de un subproceso en DFD

Al presionar dos veces el rectángulo del subproceso, se abrirá una ventana donde digitaremos el Nombre del subprograma y los Parámetros o argumentos que recibirá. También, se puede agregar una Descripción de la funcionalidad del subprograma. La ventana luce así (figura 5.9):

**FIGURA 5.9.** Ventana de datos de un subproceso en DFD

Para navegar entre los diagramas del algoritmo principal y los de otros subprogramas, se utilizan los siguientes botones (figura 5.10):



**FIGURA 5.10.** Botones para navegar entre subprocesos y diagrama principal en DFD

Si deseamos agregar una llamada al subproceso en el diagrama del algoritmo principal, se utiliza el siguiente botón:



**FIGURA 5.11.** Botón para agregar una llamada a un subproceso en DFD

Al oprimir el botón anterior estando posicionado en el proceso, se crea un subproceso, el cual puede colocar en el lugar deseado y, al dar clic, se insertará en el diagrama el siguiente símbolo:

Si presionamos dos veces el botón izquierdo del ratón (doble clic) en el símbolo de subprograma, se desplegará una ventana donde debemos colocar el nombre del subproceso y los parámetros o argumentos que le enviaremos.

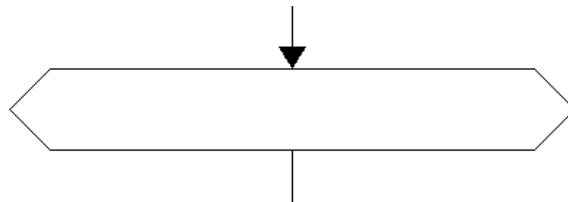


FIGURA 5.12. Símbolo de subproceso en DFD

mos. Esta luce de la siguiente forma:

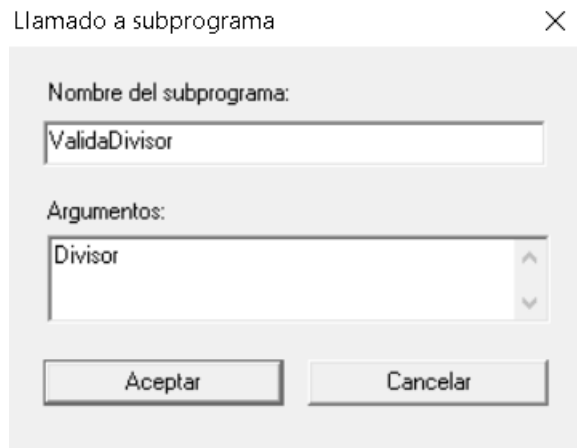


FIGURA 5.13. Ventana de datos de llamada a subproceso en DFD

Teniendo todos los aspectos anteriores claros, se presentan, a continuación, los diagramas de flujo del ejemplo de la división. Primero, el algoritmo principal (figura 5.14); y, posteriormente, los dos subprocesos `ValidaDivisor` y `Divide` (figuras 5.15 y 5.16, respectivamente):

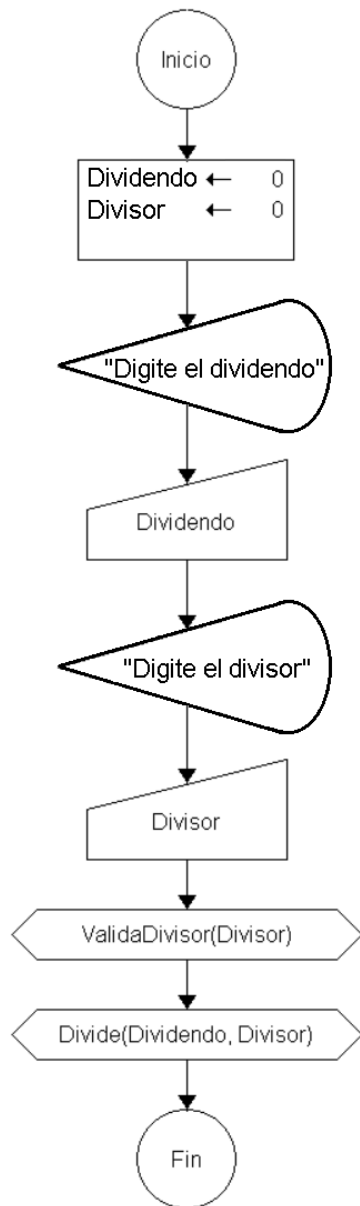


FIGURA 5.14. Diagrama de división de números con subprocesos en DFD

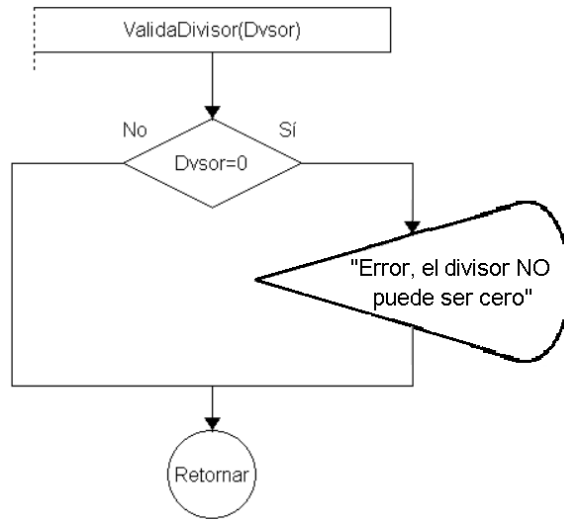


FIGURA 5.16. Detalle del subproceso en DFD de validación de divisor

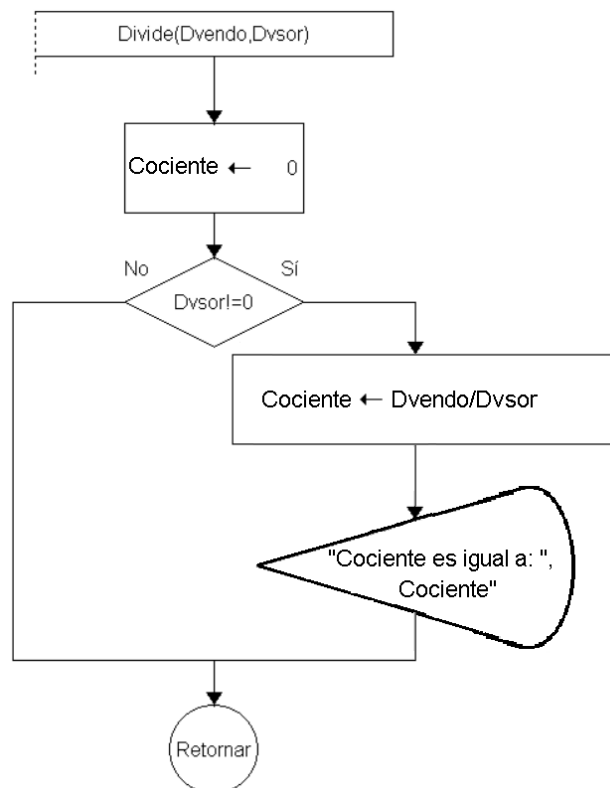


FIGURA 5.15. Detalle del subproceso en DFD que divide dos números

### 5.3 Características de los subprocesos

Los subprocesos, al igual que un algoritmo principal, tienen inicio y fin; además, poseen un **nombre** que se rige por las mismas normas de los nombres de variables, constantes o arreglos. Otra característica importante es que, para que un subproceso se ejecute, debe ser **llamado** desde el algoritmo principal o desde otro subproceso.

### 5.4 Llamada o invocación de un subproceso

La manera de llamar a un subproceso es escribir su nombre en el algoritmo de origen donde se requiere que se ejecute la funcionalidad del primero, ya sea desde el proceso principal u otro subproceso. Esta llamada se hace las veces que necesitemos, es decir, en una parte del algoritmo, podemos invocar al subproceso; y en otra más adelante, podemos invocarlo de nuevo.

### 5.5 Paso de parámetros

En la mayoría de las situaciones, cuando se llama o invoca a un subproceso, se deben enviar datos; a esto se le llama paso de parámetros.

El paso de parámetros, también llamados argumentos, es el envío de las variables, constantes o arreglos que necesite el subproceso para realizar su trabajo. Al remitir estos parámetros desde un algoritmo o subproceso, se debe mantener el mismo orden de envío y recepción. Por ejemplo, si en la llamada al subproceso `SubProc`, se envían los parámetros `Numero1`, `Numero2`, `Numero3` (el llamado se escribe `SubProc(Numero1, Numero2, Numero3);`) los que se deben recibir en ese mismo orden, así: `SubProc(Num1, Num2, Num3)`.

Un detalle básico, en la recepción de parámetros, es que se reciben con un nombre diferente al enviado. Nótese que, en el ejemplo, se envió Numero1, Numero2 y Numero3, pero se recibieron Num1, Num2 y Num3; son los mismos parámetros que se enviaron, pero en el subproceso, se llaman diferente para distinguirlos de los originales, y en el subproceso, se usarán Num1, Num2 y Num3 en las instrucciones. Analicemos el siguiente caso:

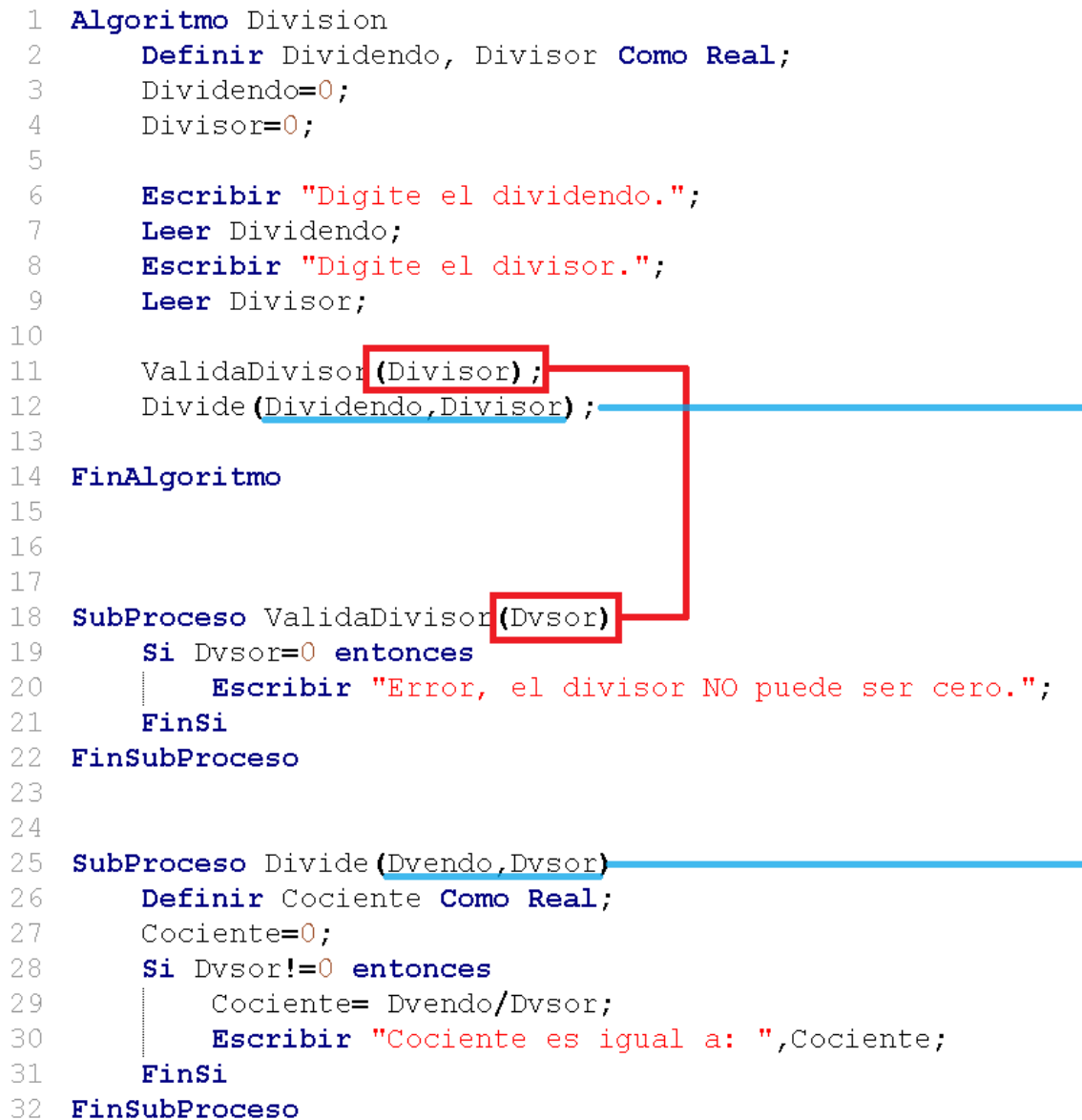


FIGURA 5.17. Nombres de parámetros en un algoritmo principal y un subprocesso en PSeInt



Como podemos ver en la línea 11, se invoca al subproceso `ValidaDivisor` y se le envía, entre paréntesis, la variable `Divisor`. En la línea 18, inicia el subproceso y recibe la variable `Divisor`, pero con el nombre `Dvsor`; ese nombre (`Dvsor`) solo se usará en el subproceso.

De igual forma trabaja el subproceso `Divide`. La diferencia es que recibe dos parámetros en lugar de uno. Nótese que el orden de envío y de recepción de parámetros es el mismo. Además, se declara la variable `Cociente`, la cual se emplea únicamente en ese subproceso.

Existen dos tipos de paso de parámetros: **por valor** y **por referencia**. A continuación, se explicarán en detalle cada uno de ellos.

### 5.5.1 Parámetros por valor (copia)

Los pasos de parámetros por valor trabajan con una **copia de la variable**, constante o arreglo enviada al subproceso. Si a ese parámetro se le cambia el valor en el subproceso, **no afectará su valor en el algoritmo de origen**, es decir, el cambio de valor solo rige en el subproceso.

#### 5.5.1.1 Parámetros por valor en `PSeInt`

En `PSeInt`, para indicar que un parámetro se pasa por valor, se digita la instrucción `Por Valor` a la derecha de cada variable que se recibe en el subproceso; pero, si no se digita nada, se asumirá que se pasaron por valor.

Veamos el siguiente ejemplo:

```

1 Algoritmo Principal
2   Definir NumeroUsuario Como Entero;
3   Definir Nombre Como Caracter;
4
5   NumeroUsuario=10;
6   Nombre="Ana";
7
8   Escribir "Variables en el principal{ANTES DEL SUBPROCESO}";
9   Escribir "NumeroUsuario: ",NumeroUsuario;
10  Escribir"Nombre: ",Nombre;
11  Escribir "";
12
13  ModificaVariables(Nombre,NumeroUsuario);
14
15  Escribir "Variables en el principal{DESPUÉS DEL SUBPROCESO}";
16  Escribir "NumeroUsuario: ",NumeroUsuario;
17  Escribir"Nombre: ",Nombre;
18
19 FinAlgoritmo
20
21
22 SubProceso ModificaVariables(Nomb,NumUsu Por Valor)
23   NumUsu= 44;
24   Nomb= "Elena";
25
26   Escribir "Variables{EN EL SUBPROCESO}";
27   Escribir "NumeroUsuario: ",NumUsu;
28   Escribir"Nombre: ",Nomb;
29   Escribir "";
30
31 FinSubProceso

```

FIGURA 5.18. Paso de parámetros por Valor en PSeInt

En el código anterior, se inicializan las variables `NumeroUsuario` y `Nombre` con 10 y Ana respectivamente (líneas 5 y 6); luego de la inicialización, se muestran sus valores.

Posterior a la muestra, se llama al subproceso `ModificaVariables` al que se le envían las variables `NumeroUsuario` y `Nombre`. Estos parámetros llegan al subproceso con los valores que se les asignó en el algoritmo principal, es decir, `Nombre` y `NumeroUsuario` llegan con los valores Ana y 10 respectivamente. Ahora, en el subproceso, se reciben ambos parámetros

Por valor (línea 22) con los nombres `Nomb` y `NumUsu`, o sea, como copias de las variables que se enviaron desde el algoritmo principal.

Dentro del subproceso, se les asignan nuevos valores a las variables. `NumUsu` pasa a tener 44 y `Nomb` tendrá Elena. Luego, se muestran estos nuevos valores (líneas 26 a 29) y se regresa el flujo de datos al algoritmo principal. Como mencionamos, este cambio en las variables solo rige para el subproceso. En las líneas 16 y 17, se despliegan los valores de `NumeroUsuario` y `Nombre` luego de regresar al subproceso, los cuales serán 10 y Ana, ya que la modificación en el subproceso no les afecta en el principal.

Los resultados del algoritmo anterior quedan evidenciados en la siguiente salida:

```
*** Ejecución Iniciada. ***
Variables en el principal{ANTES DEL SUBPROCESO}
NumeroUsuario: 10
Nombre: Ana

Variables{EN EL SUBPROCESO}
NumeroUsuario: 44
Nombre: Elena

Variables en el principal{DESPUÉS DEL SUBPROCESO}
NumeroUsuario: 10
Nombre: Ana
*** Ejecución Finalizada. ***
```

**FIGURA 5.19.** Salida de pantalla del tratamiento de paso de parámetros Por Valor en PSeInt

### 5.5.1.2 Parámetros por valor en DFD

En DFD, al recibir los parámetros, no es necesario especificar su paso, inclusive se podría decir que solo hay paso de parámetros por referencia y no por valor, ya que si se modifica el valor de una variable recibida en un subproceso, este cambio se verá también en el algoritmo de origen.

Veamos un ejemplo de paso de parámetros por referencia en la figura 5.20.

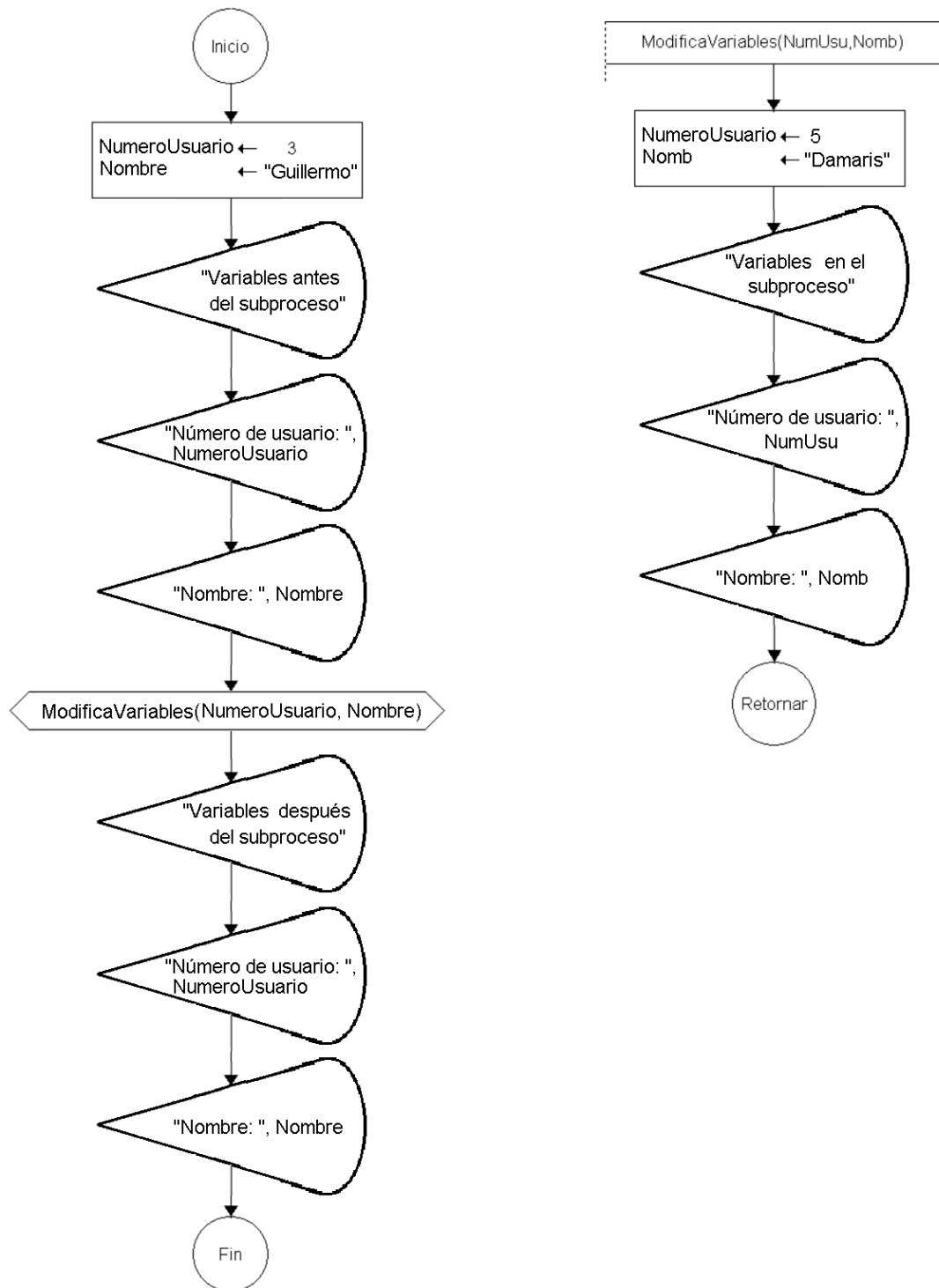


FIGURA 5.20. Paso de parámetros Por Valor en DFD

Vemos que las variables `NumeroUsuario` y `Nombre` tienen los valores 3 y Guillermo respectivamente; luego, se muestra estos valores por pantalla antes de la llamada al subproceso `ModificaVariables`.

Las salidas de pantalla **antes** de esa llamada son las siguientes:

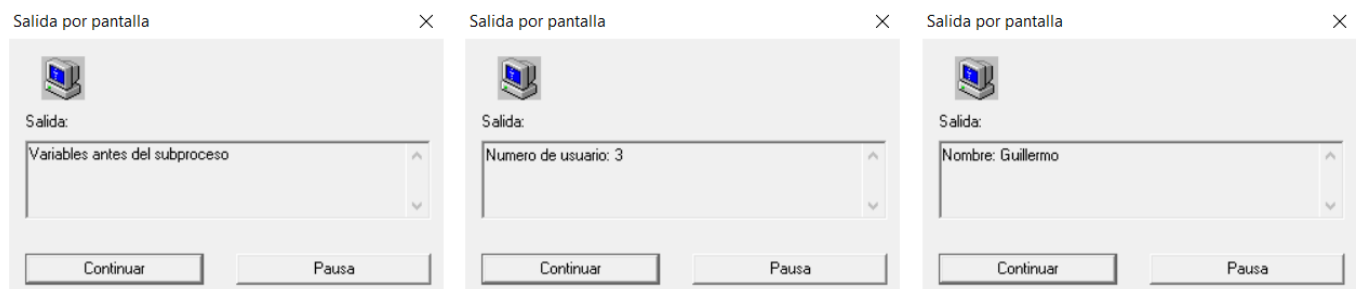


FIGURA 5.21. Salida de pantalla de parámetros Por Valor antes del subprocesso en DFD

En la llamada al subprocesso, se envían las variables `NumeroUsuario` y `Nombre`, y se reciben esos parámetros como `NumUsu` y `Nomb`. En `ModificaVariables`, se cambian los valores a 5 y Damaris respectivamente, y se muestra en pantalla la siguiente salida (figura 5.22):

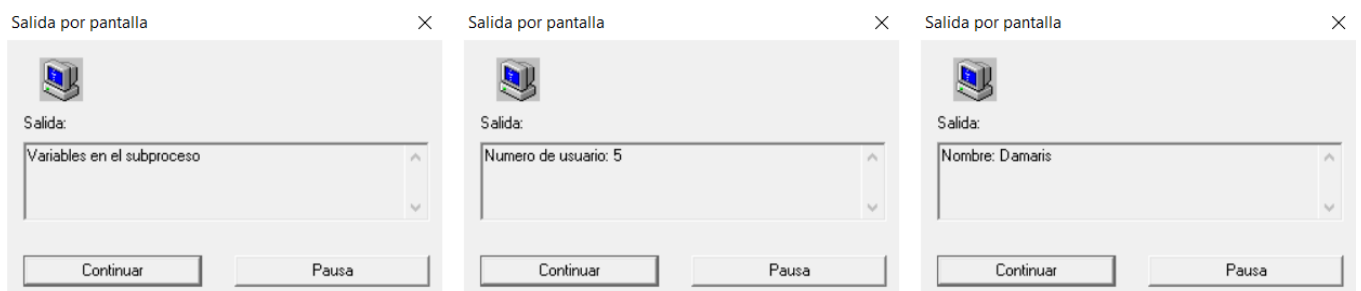


FIGURA 5.22. Salida de pantalla de parámetros Por Valor en el subprocesso en DFD

Una vez finalizado el subprocesso `ModificaVariables`, nos regresamos al algoritmo principal y se enseñan de nuevo los valores de las variables `NumeroUsuario` y `Nombre`, pero con los nuevos datos, es decir, ya no tienen los iniciales (3 y Guillermo), sino que poseen 5 y Damaris, como se muestra en figura 5.33:

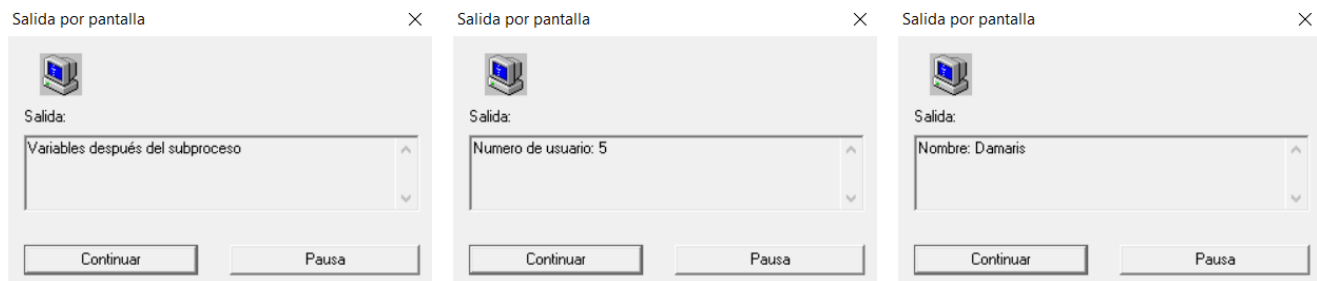


FIGURA 5.23. Salida de pantalla de parámetros Por Valor luego del subproceso en DFD

Como podemos analizar, en DFD, toda variable que se reciba en un subproceso, y a la cual se le modifique su valor, variará también ese valor en el algoritmo de origen.

### 5.5.2 Parámetros por referencia (original)

El paso de parámetros por referencia trabaja con la variable original enviada desde el origen, por lo que al hacer alguna modificación a su valor, este cambio registrará, asimismo, en el algoritmo de origen.

#### 5.5.2.1 Parámetros por referencia en PSeInt

En PSeInt, para pasar un parámetro por referencia, solo debemos escribir la instrucción `Por Referencia` al momento de recibirlo en el subproceso. Analicemos el ejemplo de la figura 5.24, en el cual se trabaja un parámetro por valor; y, otro, por referencia:

```

1 Algoritmo Principal
2   Definir NumeroUsuario Como Entero;
3   Definir Nombre Como Caracter;
4
5   NumeroUsuario=10;
6   Nombre="Ana";
7
8   Escribir "Variables en el principal{ANTES DEL SUBPROCESO}";
9   Escribir "NumeroUsuario: ",NumeroUsuario;
10  Escribir"Nombre: ",Nombre;
11  Escribir "";
12
13  ModificaVariables(Nombre,NumeroUsuario);
14

```

```

15     Escribir "Variables en el principal{DESPUÉS DEL SUBPROCESO}";
16     Escribir "NumeroUsuario: ",NumeroUsuario;
17     Escribir"Nombre: ",Nombre;
18
19 FinAlgoritmo
20
21
22 SubProceso ModificaVariables(Nomb Por Valor,NumUsu Por Referencia)
23     NumUsu= 44;
24     Nomb= "Elena";
25
26     Escribir "Variables{EN EL SUBPROCESO}";
27     Escribir "NumeroUsuario: ",NumUsu;
28     Escribir"Nombre: ",Nomb;
29     Escribir "";
30
31 FinSubProceso

```

FIGURA 5.24. Paso de parámetros Por Referencia en PSeInt

En este ejemplo, se declaran dos variables `NumeroUsuario` y `Nombre`, y se inicializan en 10 y Ana respectivamente (líneas 5 y 6). En las líneas 9 y 10, se muestran sus valores; luego, en la línea 13, se invoca al subproceso `ModificaVariables`.

El subproceso recibe el parámetro `Nomb Por valor`; y, el parámetro, `NumUsu Por referencia`. Después, modifica los valores de ambos. La diferencia será que `NumUsu` cambiará a 44 no solo en el subproceso, sino también en el algoritmo principal, mientras que `Nomb` lo hará únicamente en el subproceso.

Al regresar a las líneas 16 y 17, los valores que se mostrarán serán 44 y Ana, por ende, el 10 que tenía `NumeroUsuario` al inicio ya no existe; pero el Ana de `Nombre`, sí. Esta secuencia la observamos en la salida de pantalla:

```

*** Ejecución Iniciada. ***
Variables en el principal{ANTES DEL SUBPROCESO}
NumeroUsuario: 10
Nombre: Ana

Variables{EN EL SUBPROCESO}
NumeroUsuario: 44
Nombre: Elena

Variables en el principal{DESPUÉS DEL SUBPROCESO}
NumeroUsuario: 44
Nombre: Ana
*** Ejecución Finalizada. ***

```

**FIGURA 5.25.** Salida de pantalla de tratamiento de parámetros Por referencia en PSeInt

### 5.5.2.2 Parámetros por referencia en DFD

En DFD, cuando se recibe un parámetro, se hace automáticamente por referencia, es decir, toda variable que se reciba y se modifique dentro de un subproceso cambiará también su valor en el algoritmo de origen.

### 5.5.3 Variables globales

Una variable global es reconocida tanto por el algoritmo principal como por todos los subprocesos sin necesidad de que se reciba como parámetro.

Las variables globales se declaran fuera del algoritmo principal y su valor puede ser manipulado por cualquier fragmento del algoritmo, ya sea el proceso principal o un subproceso.

En este sentido, se resalta que el uso de variables globales no es recomendado, ya que si se hace un cambio en su valor, afectará a todas las partes del algoritmo donde se utilice y esto puede traer consecuencias en caso de no tener un control adecuado del código.



Si se requiere que un subproceso tenga datos externos, lo mejor es pasarlos como parámetros. Según Hernández (2013-2014), las excepciones para utilizar variables globales son las constantes globales, las cuales contienen valores inalterables.

En `PSeInt` y en `DFD` no existen variables globales. En general, todas las variables que se empleen en el proceso principal o en algún subproceso deben declararse a lo interno de cada uno o recibirse como parámetros.

#### 5.5.4 Variables locales

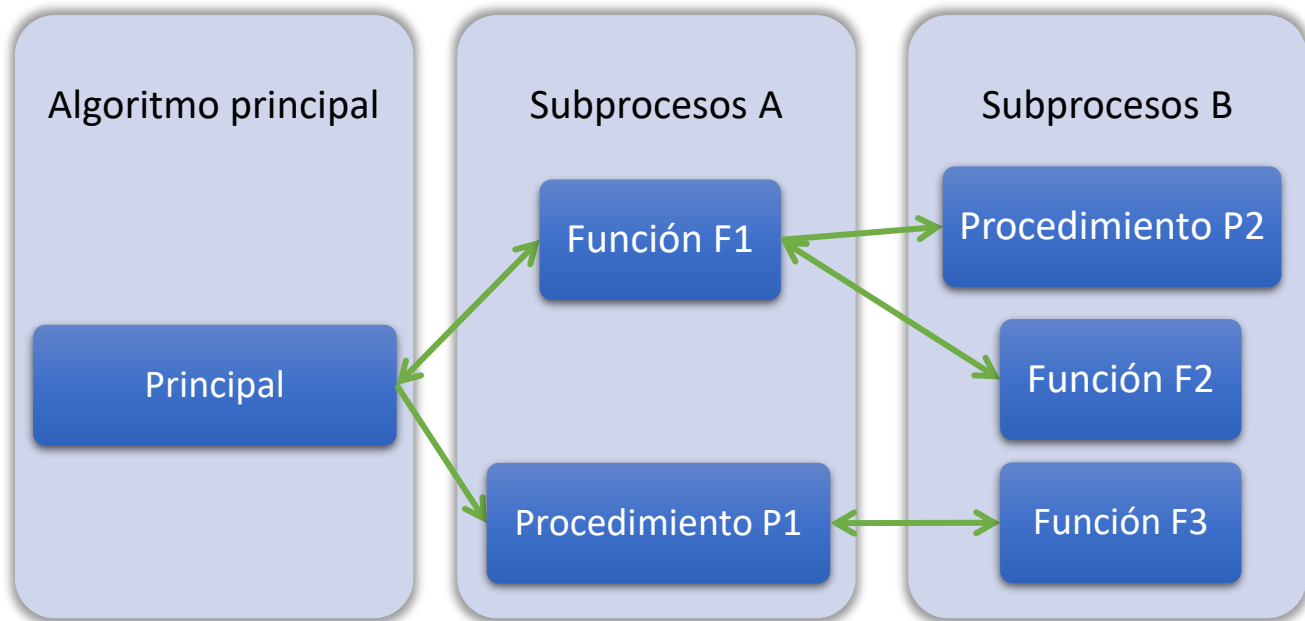
Una variable local se declara dentro del proceso principal o de un subproceso y su uso se restringe a esa parte del código; es decir, no puede usarse en otra parte, a menos que se vuelva a declarar en la nueva ubicación donde se desee emplear.

Tanto en `PSeInt` como en `DFD`, todas las variables se toman como locales, ya que se deben declarar dentro del proceso o subproceso donde se utilizarán y no se pueden emplear en otras partes del algoritmo.

### 5.6 Tipos de subprocesos

Existen dos tipos de subprocesos: los procedimientos y las funciones; ambos son similares en sus características de nombre y llamado, pero varían en su funcionalidad.

Los **procedimientos** efectúan un trabajo y **no regresan** valores o datos al algoritmo de origen, mientras que las **funciones sí retornan** un valor o dato.



**FIGURA 5.26.** Esquema de retorno de datos y diferencia entre procedimientos y funciones

Como se observa en la figura 5.26, los procedimientos tienen flechas en una sola dirección. Esto significa que únicamente reciben datos del algoritmo principal o de algún otro subproceso, mientras que las funciones tienen flechas dobles por lo que reciben y retornan datos. Asimismo, se aclara que puede suceder que los subprocesos, procedimientos y funciones, no reciban datos. Si es un procedimiento, hará lo que debe ejecutar, sin retornar valor alguno, y la función efectuará su trabajo y siempre regresará algún valor.

### 5.6.1 Procedimientos en PSeInt

En PSeInt, un procedimiento tiene la estructura semejante a la que se muestra en la figura 5.27.

El subproceso `Multiplicacion` solo lleva el nombre y, luego, los parámetros que recibe entre paréntesis. En el ejemplo, son parámetros `Por Valor`. En PSeInt, si no se indica, se asume que son con ese tipo de paso.

```

1 Proceso PrincMultiplicacion
2   Definir Numero1, Numero2, Respuesta como real;
3   Numero1=0;
4   Numero2=0;
5   Respuesta=0;
6   Escribir "Digite el valor del primer número.";
7   Leer Numero1;
8   Escribir "Digite el valor del segundo número.";
9   Leer Numero2;
10  Multiplicacion(Numero1,Numero2,Respuesta);
11 FinProceso
12
13 SubProceso Multiplicacion(Num1,Num2,Resp)
14   Resp=Num1*Num2;
15   Escribir "Resultado";
16   Escribir Num1," * ",Num2," = ",Resp;
17 FinSubProceso

```

FIGURA 5.27. Multiplicación de números usando un procedimiento en PSeInt

En el algoritmo principal, se leen las variables Numero1 y Numero2; y, en la línea 10, se llama al procedimiento Multiplicacion, en el cual se hace la multiplicación y se muestra el resultado de una vez, por cuanto no devuelve valor alguno al proceso principal.

Hagamos una modificación al algoritmo. Crearemos un procedimiento, llamado LecturaDatos, donde se lean las variables, pero se deben modificar en su valor original, por lo que se reciben Por Referencia. Todo el código sería el siguiente:

```

1 Proceso PrincMultiplicacionV2
2   Definir Numero1, Numero2, Respuesta como real;
3   Numero1=0;
4   Numero2=0;
5   Respuesta=0;
6
7   LecturaDatos(Numero1, Numero2);
8   Multiplicacion(Numero1,Numero2,Respuesta);
9 FinProceso
10

```

```

11 SubProceso LecturaDatos (Num1 Por Referencia, Num2 Por Referencia)
12     Escribir "Digite el valor del primer número.";
13     Leer Num1;
14     Escribir "Digite el valor del segundo número.";
15     Leer Num2;
16 FinSubProceso
17
18 SubProceso Multiplicacion (Num1, Num2, Resp)
19     Resp=Num1*Num2;
20     Escribir "Resultado";
21     Escribir Num1, " * ", Num2, " = ", Resp;
22 FinSubProceso

```

Figura 5.28. Pseudocódigo en PSeInt usando procedimientos para leer datos y multiplicarlos

### 5.6.2 Procedimientos en DFD

Un procedimiento en DFD funciona igual que en PSeInt. Veamos el diagrama del ejemplo de multiplicación de dos números con lectura de los valores en un procedimiento (figura 5.29):

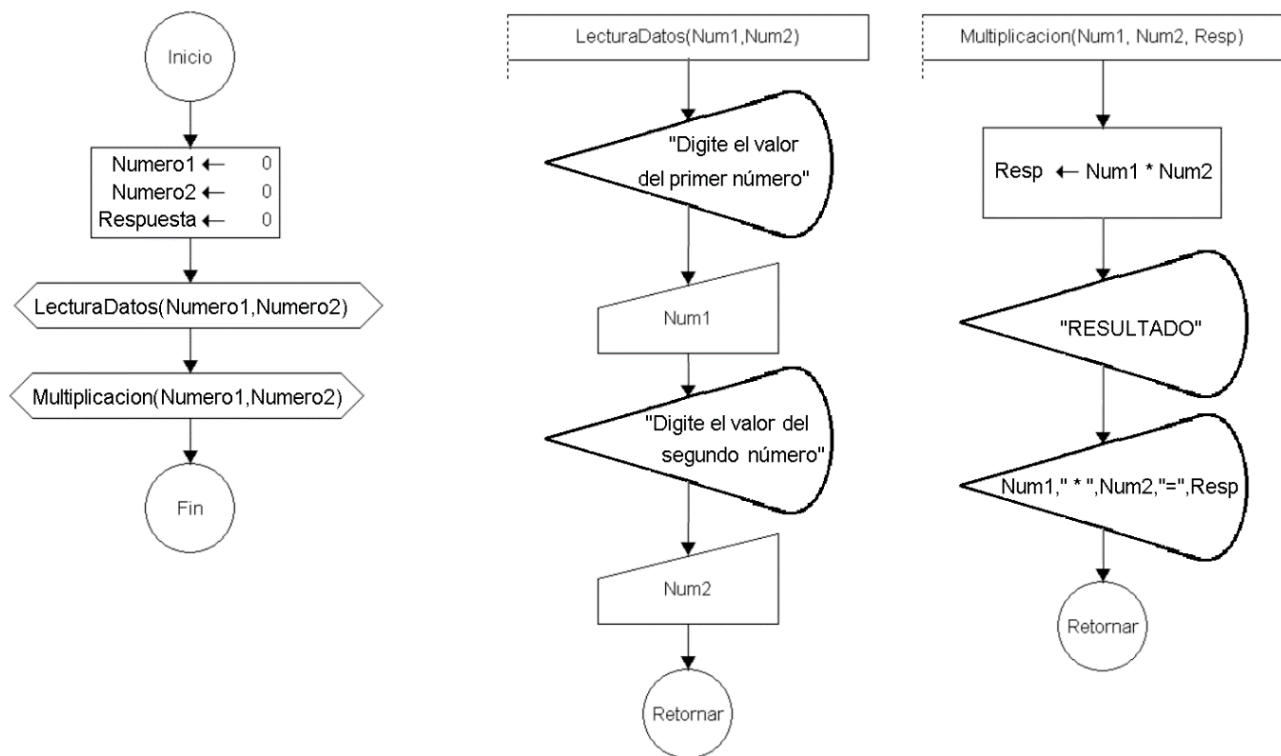


FIGURA 5.29. Diagrama en DFD usando procedimientos para leer datos y multiplicarlos

### 5.6.3 Funciones en PSeInt

Las funciones en PSeInt se deben utilizar como parte de una expresión; puede ser en una asignación de valores o en una comparación. Esto se debe a que las funciones retornan un valor y este valor hay que almacenarlo o emplearlo directamente en una instrucción.

Una función tiene la siguiente estructura (figura 5.30):

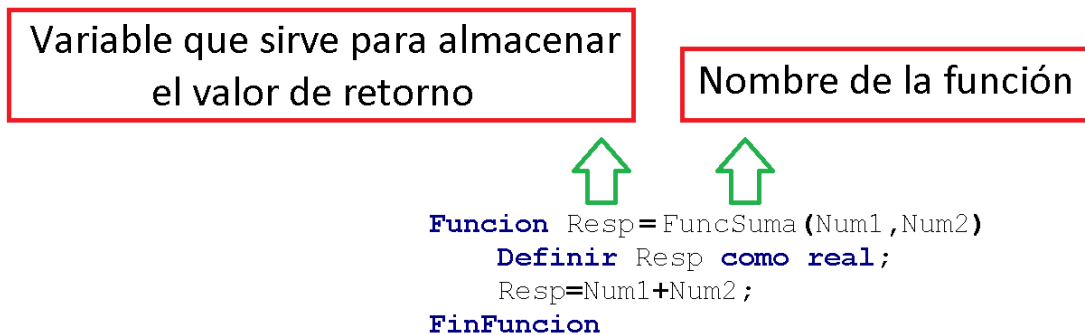


FIGURA 5.30. Detalle de la estructura de una función en PSeInt

Ahora veremos la implementación de esta función dentro de un algoritmo:

```
1  Proceso PrincipalSuma
2    Definir Numero1, Numero2 Como Real;
3    Numero1=0;
4    Numero2=0;
5
6    Escribir "Digite el primer numero.";
7    Leer Numero1;
8    Escribir "Digite el segundo numero.";
9    Leer Numero2;
10   Escribir "El resultado es: " ,FuncSuma (Numero1, Numero2);
11  FinProceso
12
13  Funcion Resp=FuncSuma (Num1,Num2)
14    Definir Resp como real;
15    Resp=Num1+Num2;
16  FinFuncion
```

FIGURA 5.31. Pseudocódigo en PSeInt que usa una función para sumar dos números

Podemos notar que, en el proceso principal, se declaran, solicitan y leen las variables `Numero1` y `Numero2`. Posteriormente, se muestra el resultado de

la suma; pero en esa misma instrucción (línea 10), se invoca a la función `FuncSuma` y se le envían las variables `Numero1` y `Numero2`.

Antes de completar la instrucción `Escribir`, se ejecuta la función. En ella, se reciben `Por Valor` los parámetros `Num1` y `Num2`. También, se declara la variable `Resp` (línea 14), la cual se utiliza para almacenar el resultado de la suma de `Num1` más `Num2` (línea 15). Después, esa misma variable (`Resp`) se emplea para retornar el valor al algoritmo principal. Esto se hace en el encabezado de la función (línea 13).

La salida en pantalla de este algoritmo es la siguiente:

```
*** Ejecución Iniciada. ***
Digite el primer numero.
> 38
Digite el segundo numero.
> 42
El resultado es: 80
*** Ejecución Finalizada. ***
```

**FIGURA 5.32.** Salida de pantalla del pseudocódigo que suma dos números con una función

#### 5.6.4 Funciones en DFD

En DFD, los subprocesos no se diferencian entre procedimientos y funciones. Esto obedece a que se trabaja la modificación de valores de variables, ya sea solo dentro de un subproceso o en todo el algoritmo.

En un diagrama de flujo, siempre se debe invocar a un subproceso con el símbolo respectivo de manera individual, es decir, no forma parte de una expresión (como lo vimos en `PSeInt`).

Veamos cómo quedaría un diagrama de flujo del ejemplo anterior (figura 5.31) en `PSeInt`:

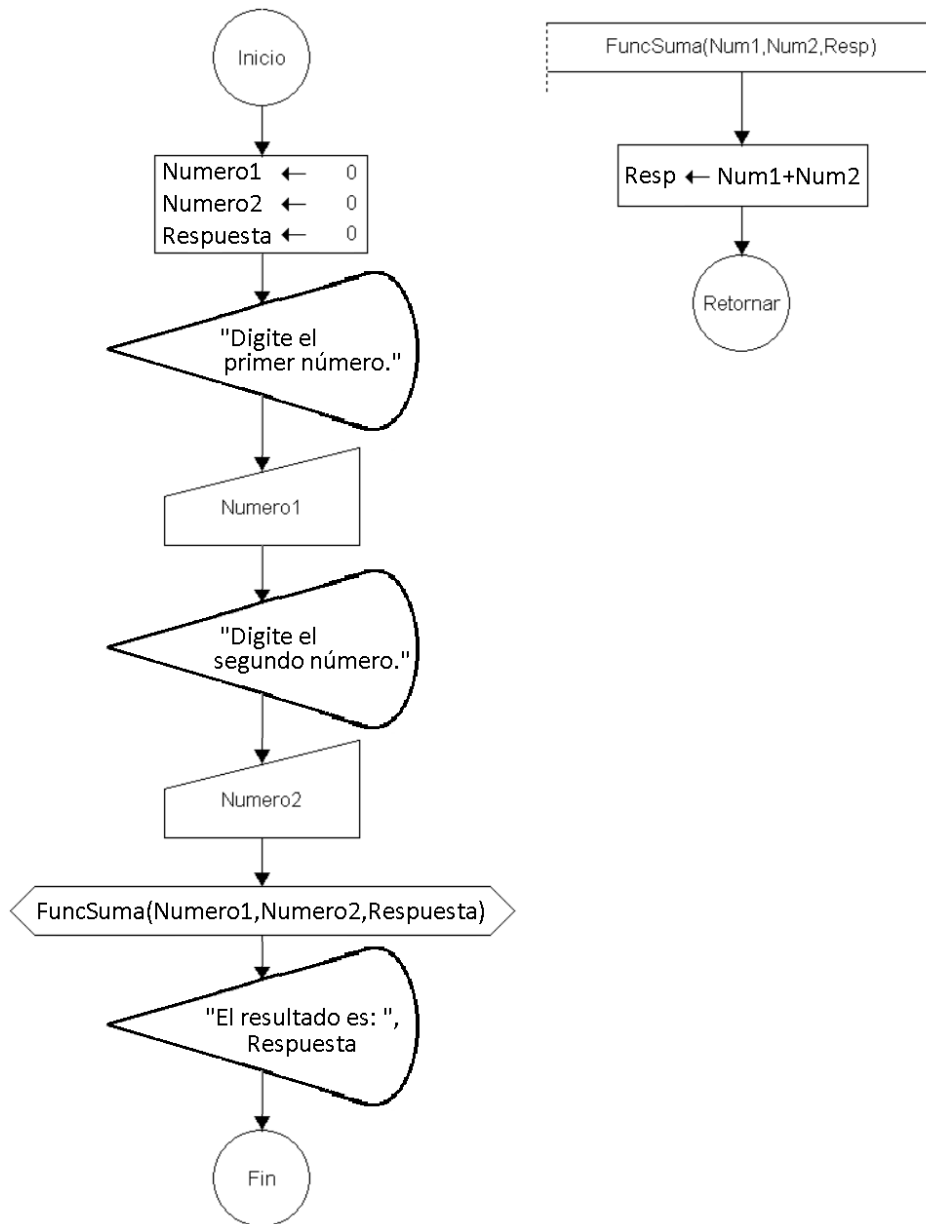


FIGURA 5.33. Diagrama en DFD que suma dos números usando el equivalente a una función

## 5.7 Ejemplos de algoritmos con subprocesos

A continuación, se desarrollan algunos ejemplos de algoritmos que emplean subprocesos. Primero, se ofrece una explicación del funcionamiento del algoritmo; luego, se muestra su versión sin subprocesos; y, posteriormente, el equivalente empleando subprocesos.

### 5.7.1 Ejemplo 1. Algoritmo para dividir dos números

**Descripción:** se toman dos números enteros y se dividen entre sí; pero antes, se valida que el divisor no sea cero. Luego de calcular la división, se evalúa si el cociente (resultado de la división) es un número múltiplo de 3 y de 5".

#### *Algoritmo original*

```
1  Proceso DivisionMultiplos
2      //declaración de variables
3      Definir dividendo, divisor, cociente Como Entero;
4      //inicialización de las variables
5      dividendo =0;
6      divisor=0;
7      cociente =0;
8
9      Escribir "Digite el dividendo";
10     Leer dividendo;
11     Escribir "Digite el divisor";
12     Leer divisor;
13
14     //Consulta si el divisor es cero
15     Si divisor=0 entonces
16     |     Escribir "Error, el divisor NO puede ser cero.";
17     FinSi
18
19     //Realiza la división si el divisor es diferente a cero
20     Si Divisor!=0 entonces
21     |     cociente= dividendo/divisor;
22     |     Escribir "Cociente es igual a: ",cociente;
23     FinSi
24
25     //Evaluación si el número es múltiplo de 3 y 5
26     Si ((cociente Mod 3=0) Y (cociente Mod 5=0) Y (divisor!=0)) Entonces
27     |     Escribir "El cociente ",cociente," es múltiplo de 3 y 5 a la vez.";
28     FinSi
29 FinProceso
```

FIGURA 5.34. Pseudocódigo con división de números y verificación de múltiplos de 3 y 5



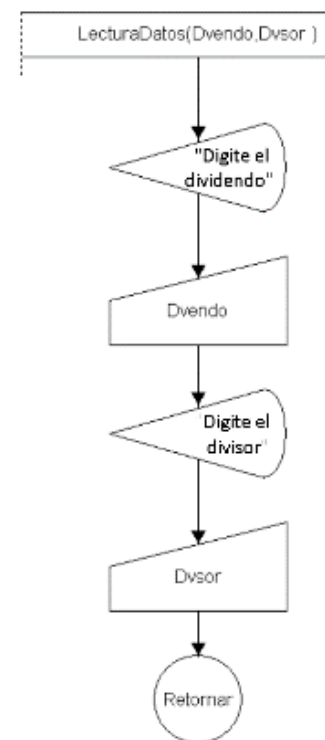
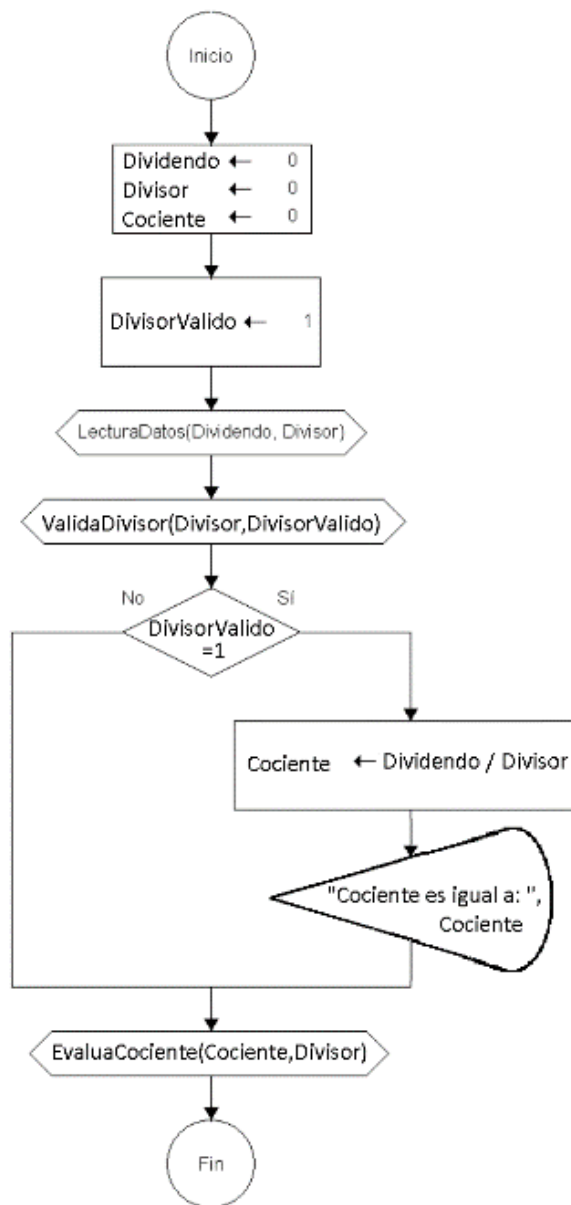
## Algoritmo con subprocesos

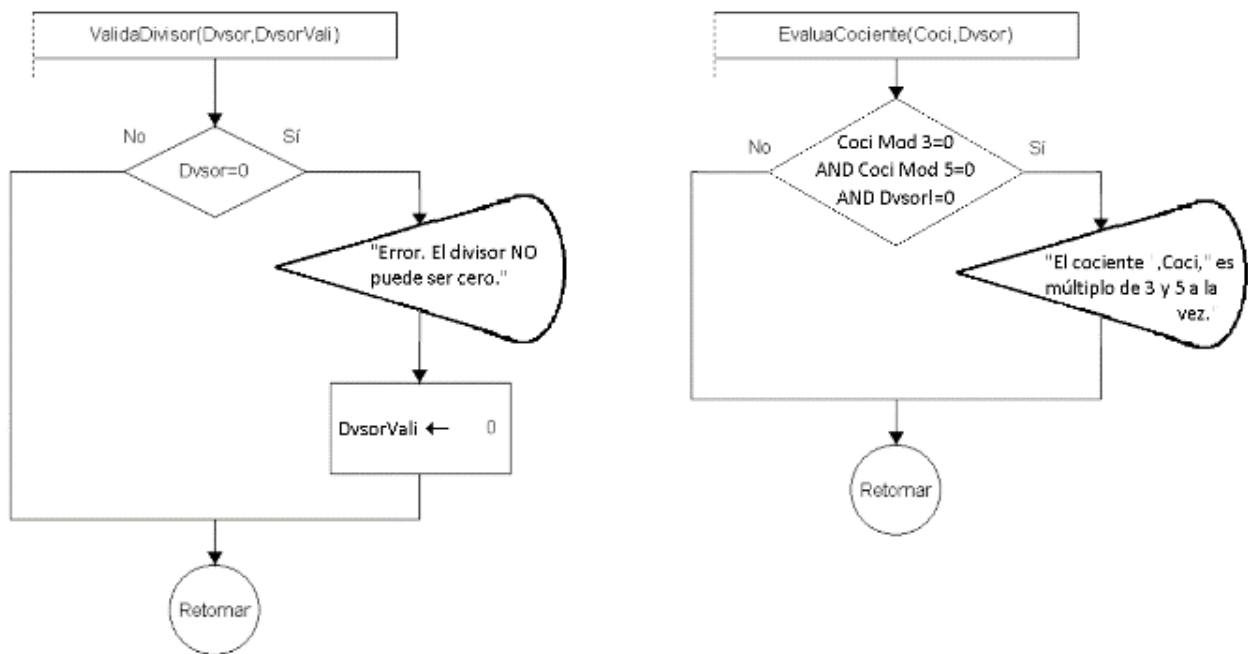
Modificaremos el algoritmo anterior para que lea, en un procedimiento, los datos; valide el dividendo mediante una función que retorna un valor booleano (verdadero o falso); y, finalmente, evalúe, en un procedimiento, el cociente de la división.

```
1  Algoritmo Division
2      Definir Dividendo, Divisor, Cociente Como Entero;
3      Dividendo=0;
4      Divisor=0;
5      Cociente=0;
6
7      LecturaDatos (Dividendo,Divisor) ;
8
9      Si ValidaDivisor (Divisor)=Verdadero entonces
10         Cociente= Dividendo/Divisor;
11         Escribir "Cociente es igual a: ",Cociente;
12     SiNo
13         Escribir "Error, el divisor NO puede ser cero.";
14     FinSi
15
16     EvaluaCociente (Cociente,Divisor) ;
17
18 FinAlgoritmo
19
20 SubProceso LecturaDatos (Dvendo Por Referencia,Dvsor Por Referencia)
21     Escribir "Digite el dividendo.";
22     Leer Dvendo;
23     Escribir "Digite el divisor.";
24     Leer Dvsor;
25 FinSubProceso
26
27 SubProceso DivisorValido=ValidaDivisor (Dvsor)
28     Definir DivisorValido Como Logico;
29     Divisorvalido=Verdadero;
30     Si Dvsor=0 entonces
31         Divisorvalido=Falso;
32     FinSi
33 FinSubProceso
34
35 SubProceso EvaluaCociente (Coci,Dvsor)
36     Si ((Coci Mod 3=0) Y (Coci Mod 5=0) Y (Dvsor!=0)) Entonces
37         Escribir "El cociente ",Coci," es múltiplo de 3 y 5 a la vez.";
38     FinSi
39 FinSubProceso
```

FIGURA 5.35. Pseudocódigo con división de números y verificación de múltiplos de 3 y 5 con subprocesos

Veamos el diagrama de flujo equivalente al pseudocódigo con subprocesos:





**FIGURA 5.36.** Diagrama con división de números y verificación de múltiplos de 3 y 5 con subprocesos



### Para tomar en cuenta

En DFD no se dispone de tipos de datos booleanos (Verdadero o Falso) por lo que un equivalente es el uso de la variable DivisorValido con 0 y 1.

### 5.7.2 Ejemplo 2. Venta de entradas al cine

**Descripción:** se recibe una opción por teclado, donde se establece si la persona es menor o mayor de edad. Si es menor de edad, solo pueden venderse entradas a la sala regular. Si es mayor de edad, se pueden elegir entradas a la sala regular o a la VIP. El precio de la sala regular es de \$ 2 500,00; y, el de la VIP, de \$5 500,00.

#### *Algoritmo original*

```
1  Proceso VentaEntradasCine
2      Definir tipoCliente, TipoSala Como Caracter;
3      Definir salaReg, salaVIP, cantEntradas, tot Como Entero;
4      salaReg=2500;
5      salaVIP=5500;
6      Escribir 'Tipo de cliente (m=menor edad, M=Mayor Edad)'
7      Leer tipoCliente;
8      Si tipoCliente='m' Entonces
9          Escribir '¿Cuántas entradas desea?'
10         Leer cantEntradas;
11         tot=cantEntradas*salaReg;
12         Escribir 'El total a pagar es: ', tot;
13     Sino //Se asume que es mayor de edad porque ya se evaluó
14         //en la condición anterior a los menores de edad
15         Escribir '¿Tipo de sala a la que desea asistir?';
16         Leer TipoSala;
17         Si TipoSala='R' Entonces
18             Escribir '¿Cuántas entradas desea?'
19             Leer cantEntradas;
20             tot=cantEntradas*salaReg;
21             Escribir 'El total a pagar es: ', tot;
22         Sino //Sino es a sala regular, entonces es la sala VIP
23             Escribir '¿Cuántas entradas desea?'
24             Leer cantEntradas;
25             tot=cantEntradas*salaVIP;
26             Escribir 'El total a pagar es: ', tot;
27         Fin Si
28     Fin Si
29 FinProceso
```

FIGURA 5.37. Pseudocódigo del cálculo de costo de entradas al cine sin subprocesos

### Algoritmo con subprocesos

```
1  Algoritmo VentaEntradasCine
2      Definir tipoCliente, tipoSala Como Caracter;
3      Definir salaReg, salaVIP, cantEntradas Como Entero;
4      salaReg=2500;
5      salaVIP=5500;
6      tipoSala="R";
7      Escribir "Tipo de cliente [m=menor de edad, M= mayor de edad]";
8      Leer tipoCliente;
9      LeerEntradas(cantEntradas);
10     Si tipoCliente="M" Entonces
11         Escribir "¿Tipo de sala a la que desea asistir? R=Regular V=VIP.";
12         Leer tipoSala;
13     FinSi
```

FIGURA 5.38. Pseudocódigo cálculo de costo entradas al cine con subprocesos

En la versión con subprocesos, se usó un procedimiento para captar la cantidad de entradas y una función para calcular el costo de la compra.

El procedimiento de lectura de entradas, por una parte, utiliza paso de parámetros *Por Referencia* para modificar el valor original de la variable `cantEntradas`. Por otra parte, en la función de cálculo de costo, se declara la variable `total` para retornarla al algoritmo de origen. Nótese que esta función forma parte de la instrucción `Escribir` de la línea 14.

#### 5.7.3 Ejemplo 3. Cálculo del índice de masa corporal (IMC)

**Descripción:** el diagrama de flujo recibe por teclado el peso en kilos y la estatura en centímetros y calcula el IMC, el cual se logra al aplicar la siguiente fórmula:

$$IMC = \frac{Peso}{Estatura\ en\ metros^2}$$

Obsérvese que la estatura en la fórmula está en metros, por ello, se debe pasar el dato introducido por el usuario a esa unidad, es decir, se divide la

estatura original entre 100. Luego del cálculo, se debe mostrar un mensaje del estado según la tabla presentada a continuación:

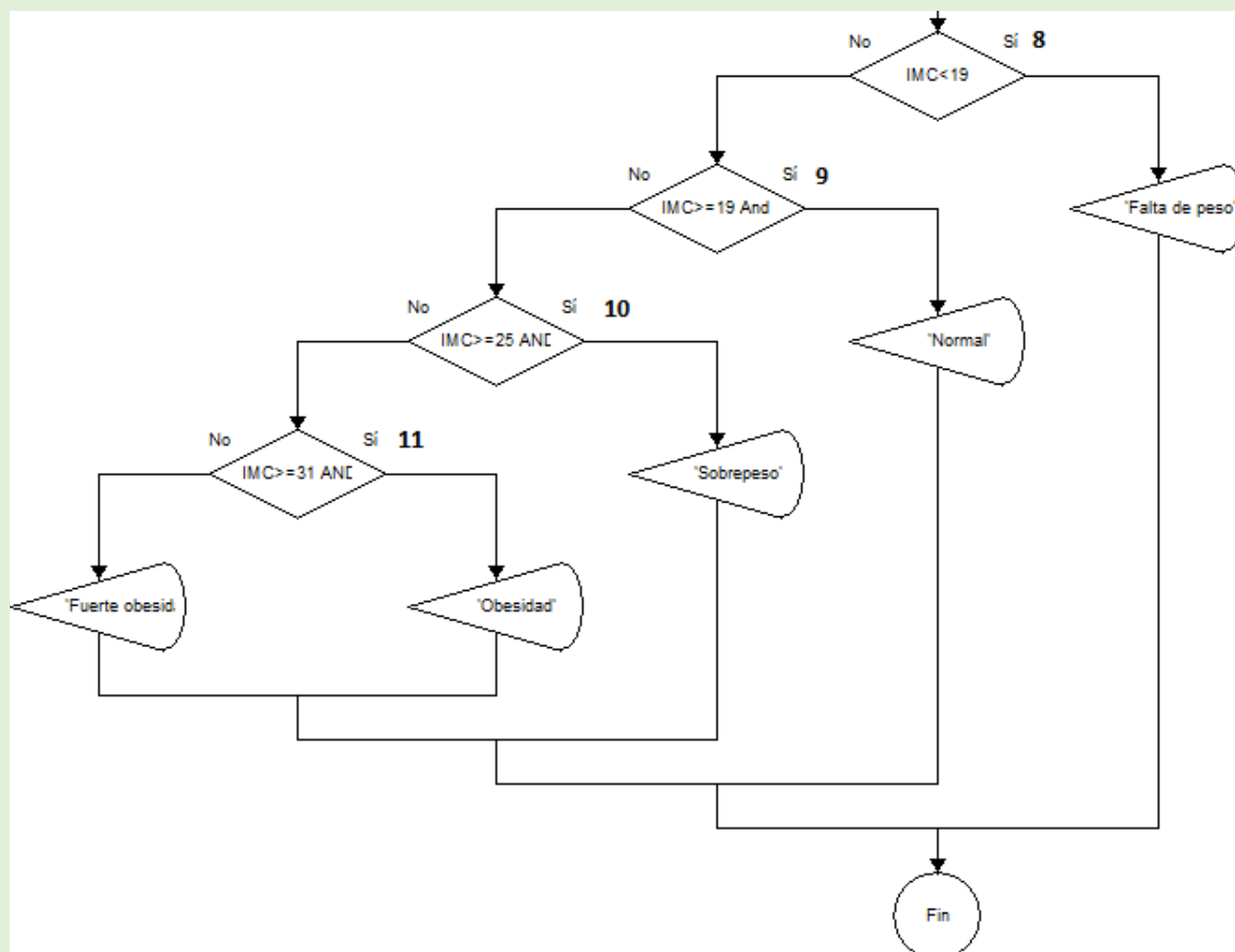
**Cuadro 5.1. Índice de masa corporal**

Mensaje	IMC
Falta de peso	por debajo de 19
Normal	19-24
Sobrepeso	25-30
Obesidad	31-40
Fuerte obesidad	mayor de 40

## Algoritmo original

Diagrama de flujo	Algoritmo en lenguaje natural
<pre> graph TD     1((Inicio)) --&gt; 2[IMC ← 0 Peso ← 0 Estatura ← 0]     2 --&gt; 3[/Peso/]     3 --&gt; 4[/Peso/]     4 --&gt; 5[/Estatura/]     5 --&gt; 6[/Estatura/]     6 --&gt; 7[IMC ← Peso/Es]     7 --&gt; 8((1)) </pre>	<ol style="list-style-type: none"> <li>1. Inicio</li> <li>2. Declarar las variables que se requieren en el problema: la estatura, el peso y el IMC.</li> <li>3. Solicitar al usuario que indique el peso.</li> <li>4. Leer, desde teclado, el peso en kilogramos digitado por el usuario.</li> <li>5. Mensaje al usuario para que digite la estatura en centímetros.</li> <li>6. Leer, desde teclado, la estatura en centímetros digitado por el usuario.</li> <li>7. Cálculo del índice de masa corporal, según la fórmula: <math display="block">IMC = \frac{Peso}{Estatura \text{ en metros}^2}</math> </li> </ol> <p style="text-align: right;"><i>Continúa...</i></p>

...continuación



#### Decisión

Condición:

IMC < 19

8. Si el IMC es menor a 19, envía el mensaje de Falta de peso.

#### Decisión

Condición:

IMC >= 19 And IMC <= 24

9. Si el IMC está entre 19 y 24, envía el mensaje de Normal.

Continúa...



...continuación

<p><b>Decisión</b> X</p> <p>Condición:</p> <p>IMC &gt;= 25 AND IMC &lt;= 30</p>	<p>10. Si el IMC está entre 25 y 30, envía el mensaje de Sobrepeso.</p>
<p><b>Decisión</b> X</p> <p>Condición:</p> <p>IMC &gt;= 31 AND IMC &lt;= 40</p>	<p>11. Si el IMC está entre 31 y 40, envía el mensaje de Obesidad; pero si es mayor a 41, envía el mensaje de Fuerte obesidad.</p>

FIGURA 5.39. Salida en pantalla del diagrama de flujo para determinar el IMC

### Algoritmo con subprocesos

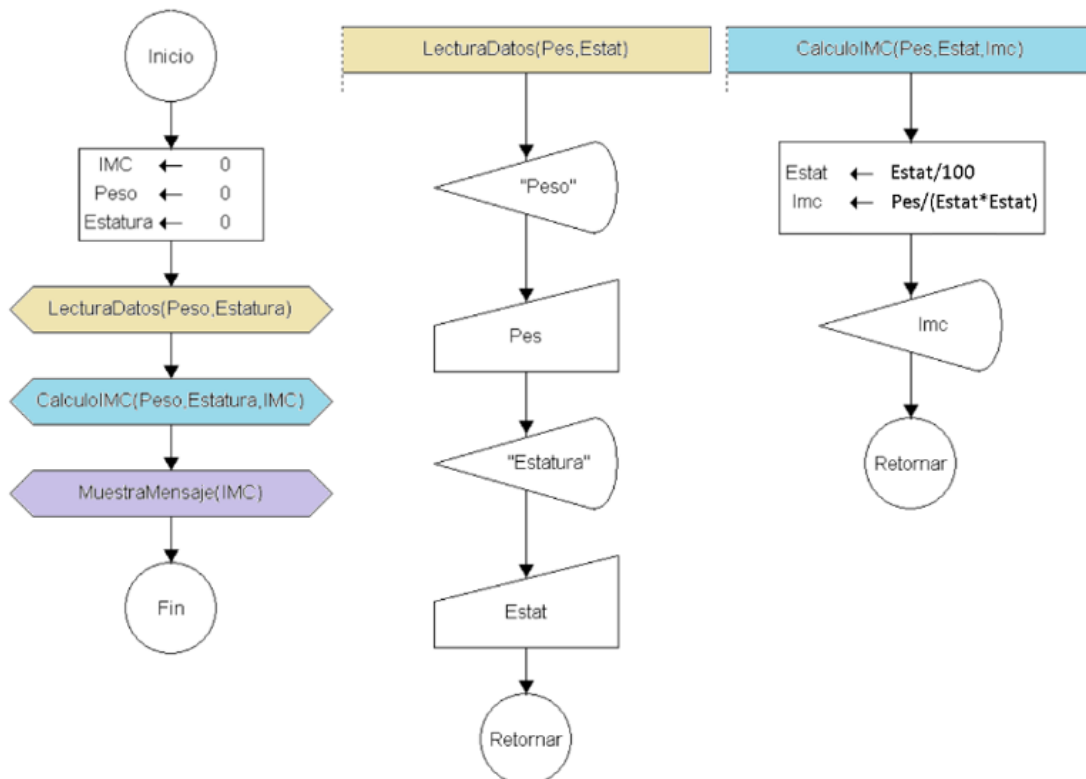


FIGURA 5.40. Primera parte diagrama de flujo con subprocesos para determinar el IMC

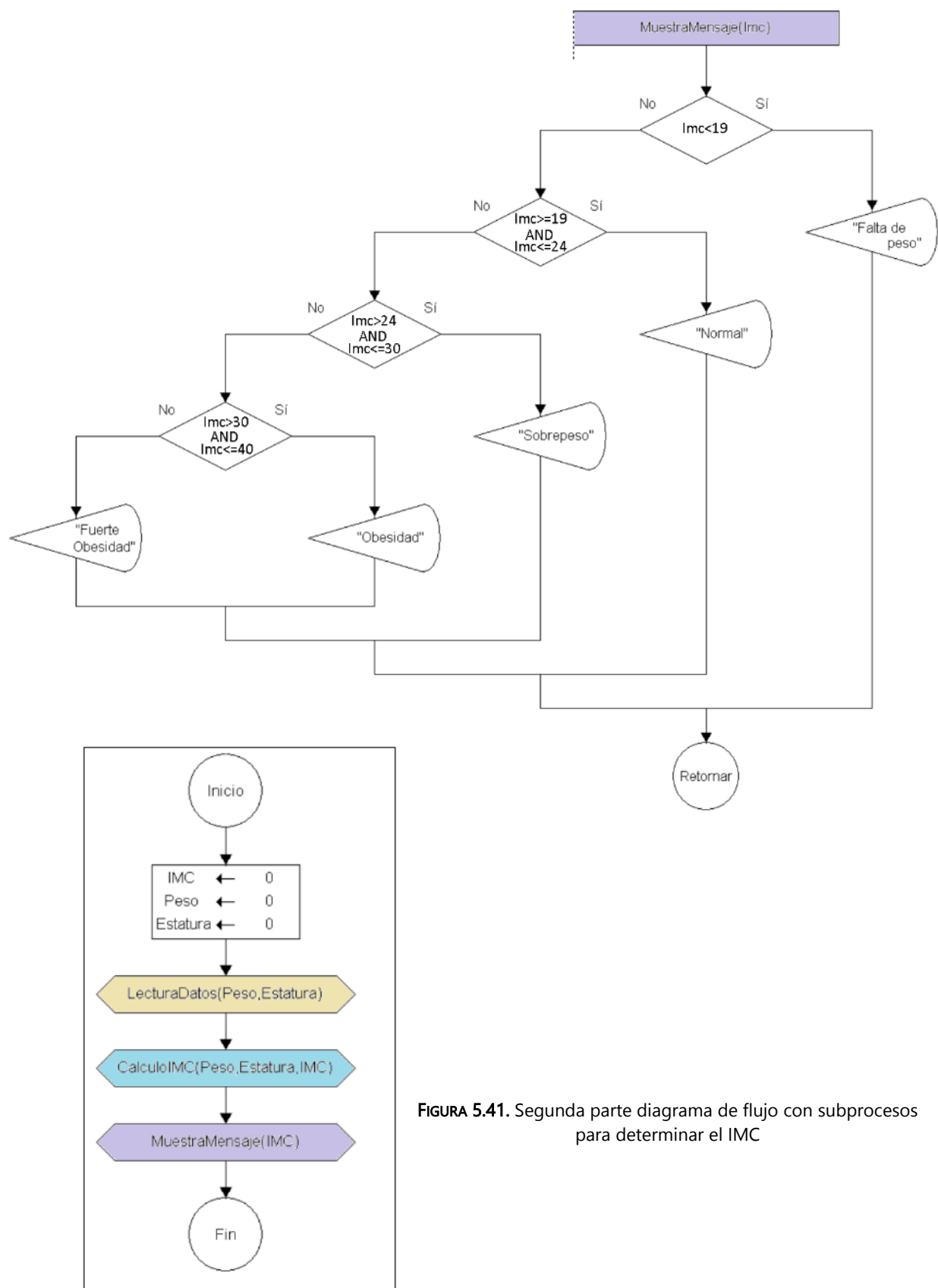


FIGURA 5.41. Segunda parte diagrama de flujo con subprocesos para determinar el IMC

En la modificación, se utilizaron tres subprocesos (para mejor identificación se pintaron en un editor de imágenes). En el primer subproceso `LecturaDatos` (figura 5.40), se leen los datos digitados por el usuario para las variables `Peso` y `Estatura`. Notemos que esas variables están declaradas en el principal, pero cambian su valor en todo el algoritmo al digitarles nuevos datos en el subproceso.

El segundo subproceso es `CalculoIMC` (figura 5.40), en el que se calcula el índice de masa corporal. Recibe los parámetros `Peso` (`Pes`), `Estatura` (`Estat`) e `IMC` (`imc`), realiza el cálculo deseado y almacena el dato en la variable `imc`. Esta modificación también aplica para el algoritmo principal.

Por último, tenemos el subproceso `MuestraMensaje`, el cual recibe el parámetro `IMC` (`imc`) y, de acuerdo con una estructura de decisiones (figura 5.41), muestra un determinado mensaje al usuario.

#### 5.7.4 Ejemplo 4. Elevar un número a un exponente con el ciclo `while`

**Funcionamiento:** el pseudocódigo recibe por teclado una base y un exponente al que se elevará la base. Al final, debe emitir un mensaje con el resultado. Por ejemplo:  $4^3$  implica que 4 se multiplica por sí mismo la cantidad de veces que indique el exponente; en este caso:  $4 \times 4 \times 4$ .

### *Algoritmo original*

```
1  Proceso CicloExponente
2      Definir base, expo, contador, tot Como Entero;
3      base=1;
4      expo=1;
5      contador=1;
6      tot=1;
7      Escribir 'Escriba la base y el exponente';
8      Leer base, expo;
9      Mientras contador<=expo Hacer
10         tot=tot*base;
11         contador=contador+1;
12      Fin Mientras
13      Escribir 'El número ',base,' elevado a ',expo, ' es ',tot;
14  FinProceso
```

FIGURA 5.42. Pseudocódigo sin subprocesos para calcular potencias

### *Algoritmo con subprocesos*

```
1  Algoritmo CicloExponente
2      Definir base,exponente,potencia Como Entero;
3      base=1;
4      exponente=1;
5      potencia=1;
6
7      IngresoValores(base,exponente);
8      potencia=CalculaPotencia(base,exponente);
9
10     Escribir "El número ",base," elevado a ",exponente," es ",potencia;
11
12  FinAlgoritmo
13
14  SubProceso IngresoValores(bas Por Referencia,expo Por Referencia)
15      Escribir "Escriba la base y el exponente.";
16      Leer bas,expo;
17  FinSubProceso
18
19  SubProceso tot=CalculaPotencia(bas Por Valor,expo Por Valor)
20      Definir tot,contador Como Entero;
21      tot=1;
22      contador=1;
23      Mientras contador<=expo Hacer
24         tot=tot*bas;
25         contador=contador+1;
26      FinMientras
27  FinSubProceso
```

FIGURA 5.43. Pseudocódigo con subprocesos para calcular potencias

En este algoritmo, volvemos a utilizar un procedimiento (líneas 7, 14, 15, 16 y 17) para la lectura de los valores que digite el usuario. Se reciben los parámetros base (bas) y exponente (expo) Por Referencia, para que, así, el valor digitado también se aplique al algoritmo principal.

Luego de la lectura, se le asigna a la variable `Potencia` el resultado de la función `CalculaPotencia` (línea 8). Al final, en la línea 10, se muestra el resultado.

Nótese que, en otros ejemplos, habíamos empleado una función como parte de la instrucción `Escribir`; sin embargo, para este caso, la utilizamos en una instrucción de asignación. Esto se hizo a fin de demostrar un uso diferente; pero si se desea, se podría haber implementado la función como parte de la instrucción `Escribir` de la línea 10. Claro, si hacemos eso, debemos eliminar la variable `potencia`, ya que no la estaríamos utilizando.

El pseudocódigo quedaría de la siguiente forma:

```
1  Algoritmo CicloExponente
2      Definir base,exponente,potencia Como Entero;
3      base=1;
4      exponente=1;
5
6      IngresoValores(base,exponente) ;
7
8      Escribir "Respuesta: ", base, " ^ ", exponente," = ",Sin Saltar;
9      Escribir CalculaPotencia(base,exponente) ;
10
11 FinAlgoritmo
12
13 SubProceso IngresoValores(bas Por Referencia,expo Por Referencia)
14     Escribir "Escriba la base y el exponente.";
15     Leer bas,expo;
16 FinSubProceso
17
18 SubProceso tot=CalculaPotencia(bas Por Valor,expo Por Valor)
19     Definir tot,contador Como Entero;
20     tot=1;
21     contador=1;
```

```

22      Mientras contador<=expo Hacer
23      |      tot=tot*bas;
24      |      contador=contador+1;
25      FinMientras
26  FinSubProceso

```

FIGURA 5.44. Pseudocódigo con subprocesos modificado para calcular potencias

### 5.7.5 Ejemplo 5. Calculadora geométrica

**Funcionamiento:** se le solicita al usuario que elija una opción de un menú; y, de acuerdo con la opción seleccionada, se calcula el área de un cuadrado, un rectángulo, un triángulo o un círculo. Si el usuario elige la opción 5, se sale del algoritmo. Además, se valida la opción del menú que digite el usuario y los valores para el cálculo de áreas.

En este ejemplo, visualizaremos el diagrama de flujo implementado con subprocesos. La figura 5.45 (página 612) muestra el diagrama principal, en el cual vemos como se declaran las variables `OpcionMenu`, `TipoDato` y `Area`; la primera es para almacenar la opción del menú que digite el usuario; la segunda, para ayudar a validar la opción del menú en un subproceso llamado `ValidaDato`; y la tercera, para almacenar el valor del área calculada, debido a que es una variable afín a cualquiera de las figuras geométricas.

Este diagrama se compone por un ciclo `Mientras`, el cual se repite mientras la opción del menú digitada sea diferente a 5. Dentro de este ciclo, se muestra el menú y se lee la opción digitada por el usuario, mediante el subproceso `MuestraMenu`; luego, se evalúa el valor de la variable `OpcionMenu` en el subproceso `ValidaDato`.

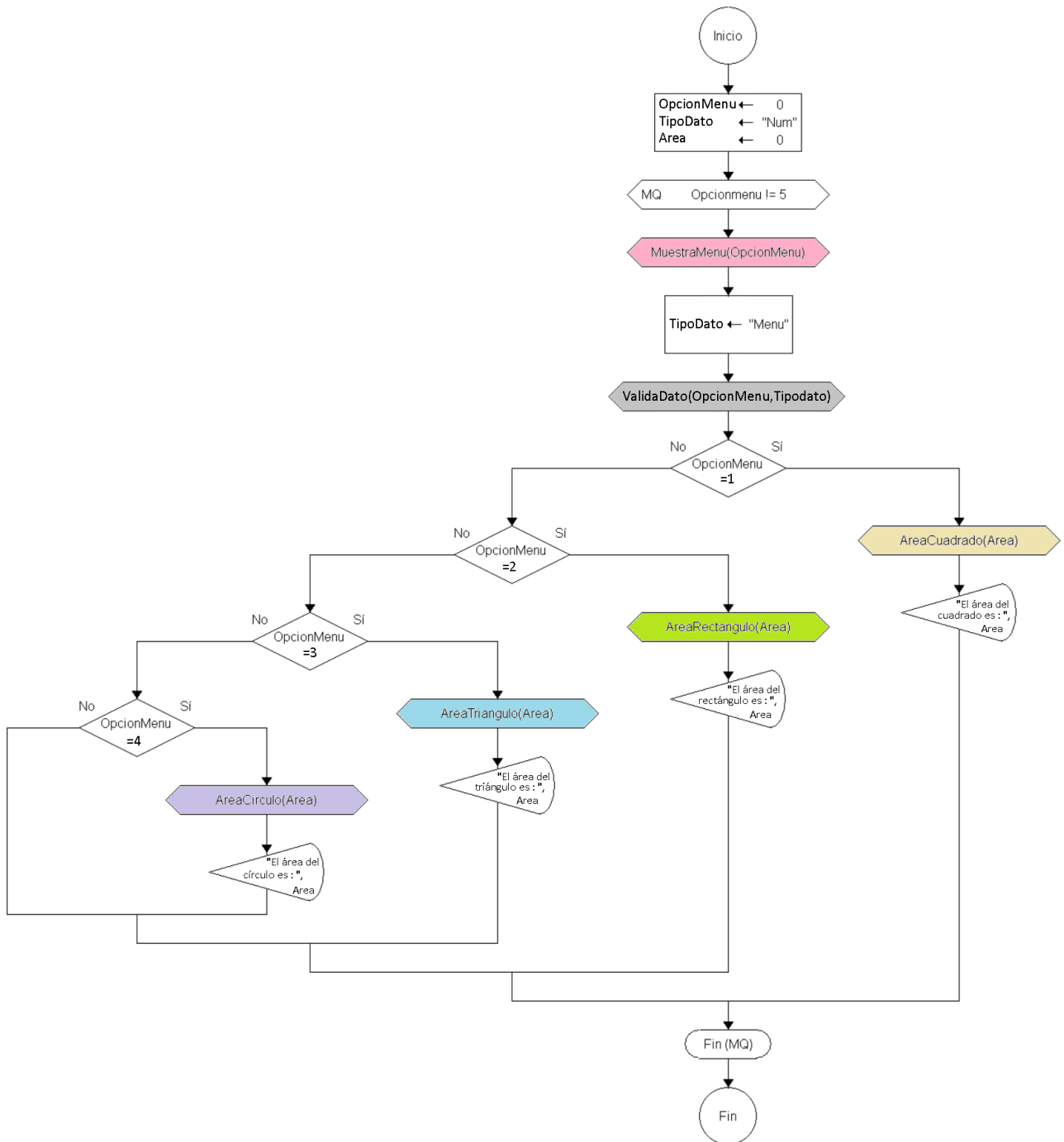


FIGURA 5.45. Diagrama principal de la calculadora geométrica

Posterior a la validación, se presentan cuatro *Si* anidados, en los que se llama a los respectivos subprocesos para el cálculo de áreas. En la figura 5.46, se muestra el detalle de los subprocesos *MuestraMenu* y *ValidaDato*:

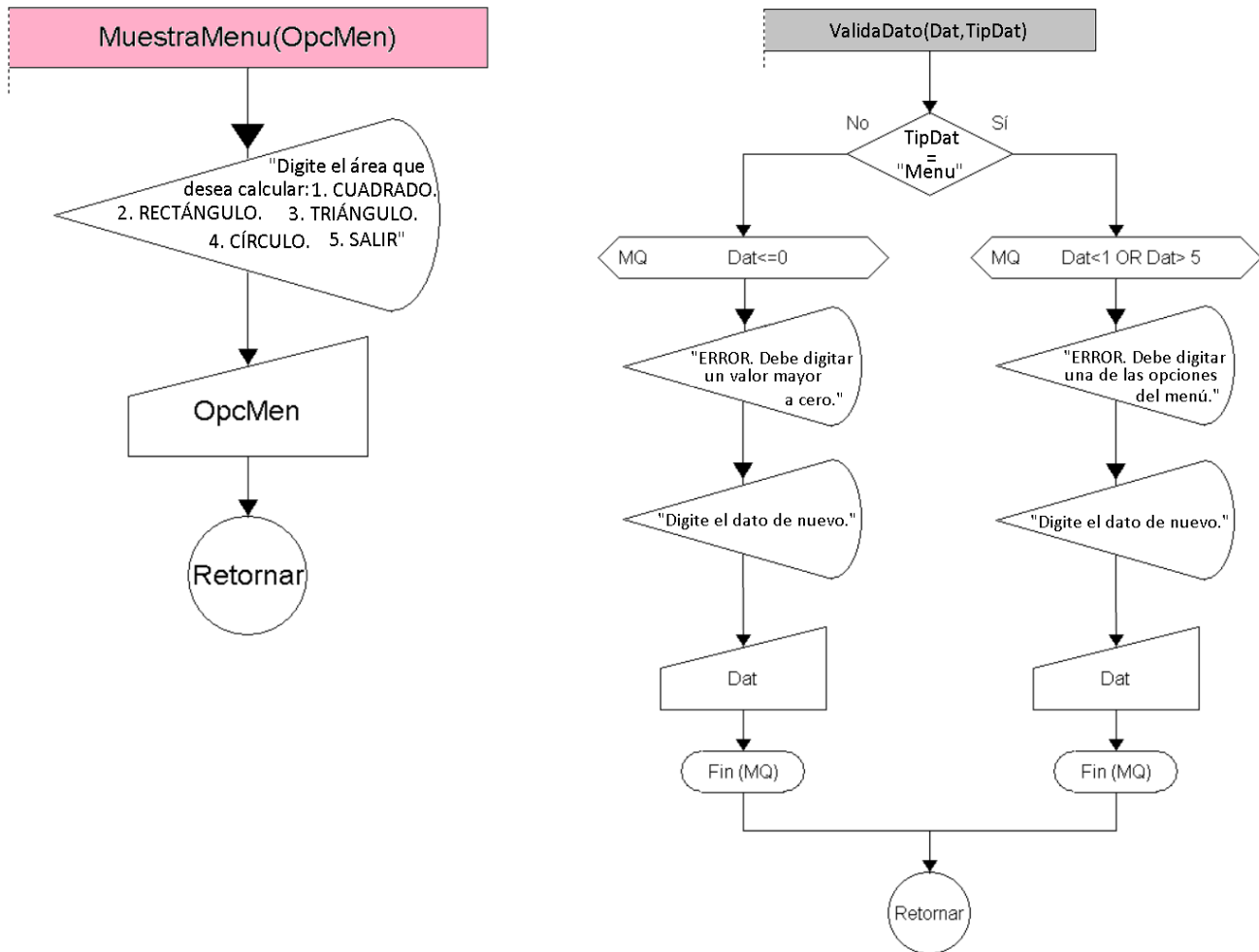


FIGURA 5.46. Diagrama de los subprocesos MuestraMenu y ValidaDato de la calculadora geométrica

El subproceso ValidaDato también se emplea para validar que los valores para el cálculo de áreas sean mayor a cero; por lo tanto, este subproceso evalúa tanto la opción del menú como los valores de los lados, altura, base o radio. Para diferenciar si es una opción del menú o si es un valor para las áreas, utiliza un Si con TipDat.

Como se mencionó, en los Si anidados, se hace el llamado a los subprocesos para el cálculo de las áreas, a saber: AreaCuadrado, AreaRectangulo, AreaTriangulo y AreaCirculo. En cada uno de ellos, se envía la variable Area. Luego del cálculo solicitado, se regresa al principal y se muestra el valor del área respectivo.



La figura 5.47 muestra los diagramas de los subprocesos:

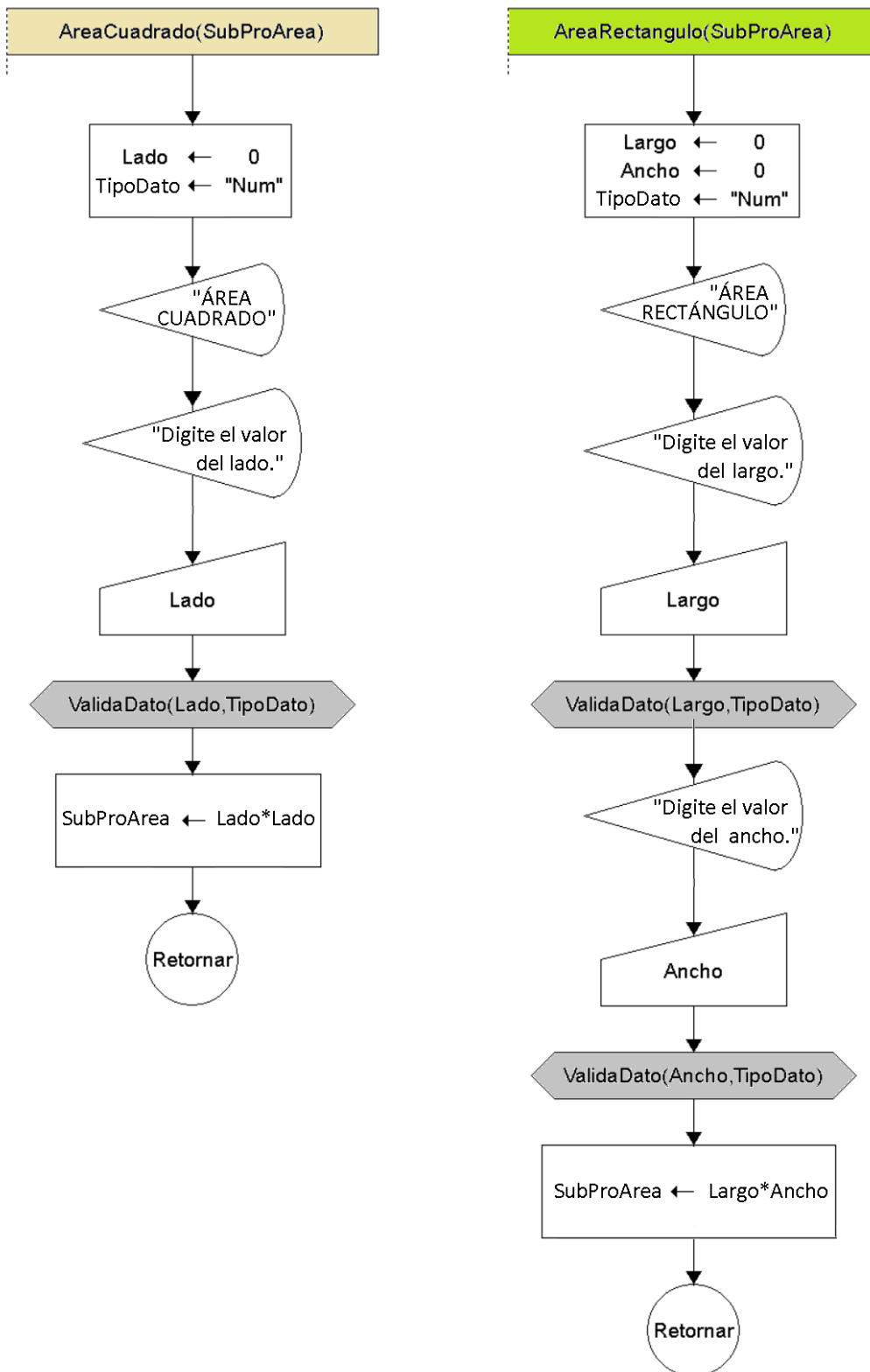


FIGURA 5.47. Diagramas de los subprocesos AreaCuadrado y AreaRectangulo de la calculadora geométrica

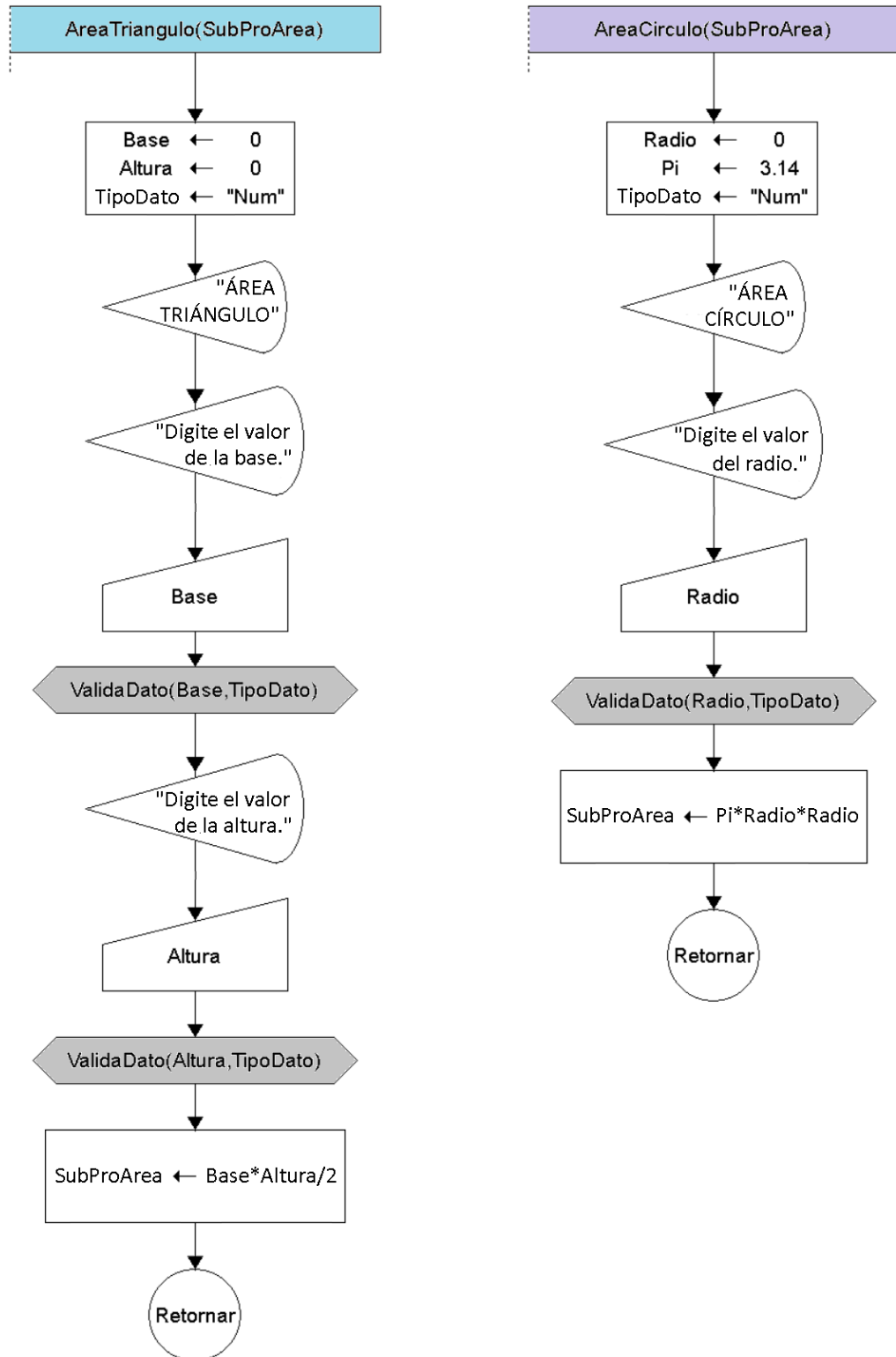


FIGURA 5.48. Diagrama se los subprocesos AreaTriangulo y AreaCirculo de la calculadora geométrica

En cada subproceso, se declaran las variables necesarias para el cálculo del área y se calcula su valor. Este se almacena en la variable `SubProArea` que es la misma `Area` que se envía en la llamada del diagrama principal. Además, se valida el valor que digita el usuario para cada solicitud por medio del subproceso `ValidaDato`. Nótese que, en esta validación, se llama al subproceso `ValidaDato` desde los subprocesos de cálculo de áreas, por ende, no solo desde el principal se pueden invocar subprocesos.

### 5.7.6 Ejemplo 6. Modificación de matriz

**Funcionamiento:** el pseudocódigo presenta un menú donde se pueden hacer diversas acciones sobre una matriz de  $3 \times 4$ , la cual se llena con números aleatorios del 10 al 99. El algoritmo principal es el siguiente:

```

1  Proceso ModificaMatrizCSubProc
2
3      //Declaraciones de variables y matriz
4      Definir Matriz Como Entero;
5      Definir OpcionMenu Como Caracter;
6      Definir MatrizLlena Como Logico;
7
8      //Dimensionamiento de la matriz
9      Dimension Matriz(3,4);
10
11     //Inicializaciones de variables
12     MatrizLlena=Falso;
13     OpcionMenu="X";
14
15     Repetir
16         MuestraMenu(OpcionMenu);
17         Segun OpcionMenu Hacer
18             "1":
19                 LlenaMatriz(Matriz,MatrizLlena);
20                 Mensaje();
21             "2":
22                 MuestraMatriz(Matriz,MatrizLlena);
23                 Mensaje();
24             "3":
25                 ModificaMatriz(Matriz,MatrizLlena);
26                 Mensaje();

```

```

27         "S", "s":
28         |     Salida();
29         De Otro Modo:
30         |     Limpiar Pantalla;
31         |     Escribir "ERROR. Opción inválida.";
32         |     Mensaje();
33     FinSegun
34 Hasta Que OpcionMenu="S" O OpcionMenu="s"
35
36 FinProceso

```

FIGURA 5.49. Pseudocódigo del algoritmo principal para modificar una matriz

Como podemos apreciar en el pseudocódigo del algoritmo principal, consta de una estructura Segun, desde la que se invocan diversos subprocesos.

El primer subproceso en invocarse es MuestraMenu, al que se le envía la variable OpcionMenu. Este es el detalle de ese subproceso (figura 5.50):

```

38 SubProceso MuestraMenu(OpcMen Por Referencia)
39     Limpiar Pantalla;
40     Escribir "Menú.";
41     Escribir "Digite una de las siguientes opciones:";
42     Escribir "1. Llenar matriz.";
43     Escribir "2. Mostrar matriz.";
44     Escribir "3. Modificar matriz [Copia].";
45     Escribir "S. Salir";
46     Escribir "OPCIÓN--->", Sin Saltar;
47     Leer OpcMen;
48 FinSubProceso

```

FIGURA 5.50. Pseudocódigo del subproceso MuestraMenu algoritmo para modificar una matriz

Vemos que este procedimiento recibe el parámetro OpcMen Por Referencia, ya que será la opción del menú que digite el usuario y debe modificarse, también, en el algoritmo de origen.

Ahora, analizaremos los subprocesos de acuerdo al orden del menú. Empecemos con LlenaMatriz. En su llamada, desde el pseudocódigo principal, se envían los parámetros Matriz y MatrizLlena. Esta segunda variable es de tipo lógico (Verdadero o Falso) y la usaremos para evitar que se

procesen las opciones 2 y 3 del menú si la matriz no está llena. Lógicamente, no se podría mostrar ni modificar una matriz que aún no tiene elementos.

La figura 5.51 muestra el pseudocódigo del subproceso:

```
67 SubProceso LlenaMatriz(Mat Por Referencia, MatLlena Por Referencia)
68     //Llenado de la matriz
69     Definir Fila,Columna Como Entero;
70     Fila=0;
71     Columna=0;
72     Para Fila<-0 Hasta 2 Con Paso 1 Hacer
73     |     Para Columna<-0 Hasta 3 Con Paso 1 Hacer
74     | |     Mat (Fila,Columna)=azar(90)+10;
75     |     FinPara
76     FinPara
77     MatLlena=Verdadero;
78     MuestraMatriz(Mat,MatLlena);
79     Escribir "Llenado de matriz completo.";
80 FinSubProceso
```

FIGURA 5.51. Pseudocódigo del subproceso LlenaMatriz algoritmo para modificar una matriz

En este procedimiento, como su nombre lo indica, llena la matriz, la cual se recibe Por Referencia, aunque recordemos que, en PseInt, todos los arreglos se manejan Por Referencia, ya sea que se indique o no.

Una vez llena la matriz, se cambia el valor de MatLlena a Verdadero (estaba inicializada en Falso) y se indica que la matriz ya tiene elementos. Finalmente, se muestra el contenido de esta; para ello, se invoca al subproceso MuestraMatriz.

MuestraMatriz es la segunda opción en el menú y su detalle es el siguiente (figura 5.52):

```

82 Subproceso MuestraMatriz(Mat,MatLlena Por Valor)
83     //Muestra de la matriz
84     Definir Fila,Columna Como Entero;
85     Fila=0;
86     Columna=0;
87     //Limpiar Pantalla;
88     Si MatLlena=Verdadero Entonces
89         Limpiar Pantalla;
90         Escribir "Valores de la matriz";
91         Para Fila<-0 Hasta 2 Con Paso 1 Hacer
92             Para Columna<-0 Hasta 3 Con Paso 1 Hacer
93                 Escribir Mat(Fila,Columna)," ",Sin Saltar;
94             FinPara
95             Escribir "";
96         FinPara
97     SiNo
98         Limpiar Pantalla;
99         Escribir ";Atención!Primero debe llenar la matriz.";
100     FinSi
101 FinSubProceso

```

FIGURA 5.52. Pseudocódigo del subproceso MuestraMatriz algoritmo para modificar una matriz

Vemos que, en `MuestraMatriz`, se implementa un `Si` para determinar si la variable `MatLlena` tiene un valor de `Verdadero` o `Falso`. Si es el primer caso, se imprime en pantalla la matriz utilizando dos ciclos `Para` anidados. Pero si el valor de `MatrizLlena` es `Falso`, se despliega un mensaje de error (línea 99) y la matriz no se exhibe, ya que se asume que está vacía. Esto sucede porque el usuario eligió la opción 2 antes que la 1, lo cual es libre de hacer, pero el pseudocódigo debe contemplar ese detalle e indicar cómo debe proceder.

La tercera opción del menú se relaciona con el subproceso `ModificaMatriz` que se despliega en la figura 5.53:

```

103 SubProceso ModificaMatriz(Mat,MatLlena)
104     Definir Fila,Columna Como Entero;
105     Fila=0;
106     Columna=0;
107     Si MatLlena=Verdadero Entonces
108         Para Fila=0 Hasta 2 Con Paso 1 Hacer
109             Para Columna=0 Hasta 3 Con Paso 1 Hacer
110                 Mat (Fila,Columna)=Mat (Fila,Columna)+1;
111             FinPara
112         FinPara
113         MuestraMatriz (Mat,MatLlena) ;
114         Escribir "Modificación de matriz completa.";
115     SiNo
116         Limpiar Pantalla;
117         Escribir "¡Atención!Primero debe llenar la matriz.";
118     FinSi
119
120 FinSubProceso

```

FIGURA 5.53. Pseudocódigo del subproceso ModificaMatriz algoritmo para modificar una matriz

En `ModificaMatriz`, también se valida si la matriz está llena o no (línea 107). Luego de verificar que el arreglo contiene elementos, se ejecuta la modificación, la cual consiste en sumar una unidad al valor de cada celda.

Podemos notar que, en todos los subprocesos de la matriz, se recibe el arreglo (`Mat`) y el parámetro `MatLlena`, pero solo en `LlenaMatriz` se recibe `MatLlena` Por Referencia, pues ahí se modifica de Falso a Verdadero. En los otros dos subprocesos se utiliza Por Valor, ya que solo se emplea para corroborar su estado y no es necesario cambiar el dato que tiene originalmente.

La cuarta opción del menú es el procedimiento `Salida`, el cual despliega una animación antes de salir del algoritmo. Su pseudocódigo se muestra en la figura 5.54:

```

50 SubProceso Salida()
51     Definir Contador Como Entero;
52     Contador=5;
53     Mientras Contador>=0 Hacer
54         Limpiar Pantalla;
55         Escribir "Saliendo en...",Sin Saltar;
56         Escribir Contador,Sin Saltar;
57         Contador=Contador-1;
58         Esperar 1 segundos;
59     FinMientras
60 FinSubProceso

```

FIGURA 5.54. Pseudocódigo del subproceso Salida algoritmo para modificar una matriz

En el algoritmo, se emplea otro subprograma llamado `Mensaje`, que muestra, en pantalla, un mensaje y espera a que el usuario presione alguna tecla. Su detalle es el siguiente (figura 5.55):

```

62 SubProceso Mensaje()
63     Escribir "Presione cualquier tecla para volver al menú.";
64     Esperar Tecla;
65 FinSubProceso

```

FIGURA 5.55. Pseudocódigo del subproceso Mensaje algoritmo para modificar una matriz





## 5.8 Ejercicios de autoevaluación

A continuación, se presentan una serie de ejercicios que le servirán para evaluar su aprendizaje de los temas vistos en este capítulo. Le recomendamos responderlos de acuerdo al conocimiento adquirido y solo después revisar el apartado “Respuesta a los ejercicios de autoevaluación”, a fin de contrastar sus respuestas con las propuestas.

1. En este paso de parámetro se hace una copia de la variable original y su cambio de valor solo rige para el subproceso y no para el algoritmo de origen: \_\_\_\_\_.
2. Tipo de paso de parámetro donde, si en el subproceso se cambia el valor de la variable, afecta también en el principal: \_\_\_\_\_.
3. Elabore un pseudocódigo en PSeInt que lea un nombre digitado por el usuario. Luego, muestre el saludo “Hola” y el nombre digitado, por ejemplo: “Hola Guillermo”. El nombre debe leerlo en el algoritmo principal. Después, debe enviarlo por referencia a un procedimiento (MuestraSaludo) que imprima el mensaje final.
4. Desarrolle un algoritmo en PSeInt que lea el valor para dos variables, una de tipo `Real` y otra de tipo `Caracter`. Luego, muestre el valor de cada una de ellas por pantalla. Posteriormente, envíe las dos variables a un procedimiento llamado `ModificaVariables`, una debe ser un parámetro `Por valor` y la otra `Por referencia`. En ese procedimiento, modifique el valor de ambas variables y presente los nuevos datos por pantalla.  
Finalmente, en el algoritmo principal, muestre de nuevo los valores de las variables. Observe los cambios que se dieron entre los valores

de la variable enviada `Por valor` y la remitida `Por referencia`.  
¿Cuál de las dos variables cambió su valor en el algoritmo principal y cuál regresó a su valor inicial?

5. Elabore un diagrama de flujo en `DFD` donde se lean los valores de dos variables. Luego, en un subproceso, se restan esos valores entre sí y se asignan a otra variable.  
En otro subproceso, se debe leer la variable donde se almacenó el resultado de la resta e indicar si ese número es positivo o negativo junto con el valor de la variable.
6. Elabore un pseudocódigo en `PSeInt` donde se evalúe, mediante una función, si una variable es múltiplo de 3 o no. Debe leer el valor de la variable en el programa principal y, luego, evaluarla en una función que se vincule a una estructura `Si`. La función debe retornar un valor booleano, es decir, `Verdadero` o `Falso`.
7. Cree un pseudocódigo en `PSeInt` que realice la conversión de colones a dólares, solicitándole al usuario la cantidad de colones y el tipo de cambio. La solicitud de datos debe hacerla en un procedimiento llamado `SolicitudDatos` y el cálculo del tipo de cambio debe hacerlo en una función (`CalculoDolares`) que tome el monto en colones digitado en `SolicitudDatos`. Utilice la función en una instrucción `Escribir`.
8. Modifique el ejercicio anterior; pero en este caso, implemente la función `SolicitudDatos` en una instrucción de asignación.
9. Elabore un diagrama de flujo en `DFD` que solicite el nombre de una persona y envíe un saludo: "Hola," más el `Nombre`. El nombre debe aceptarlo en un subproceso y la muestra del mensaje en otro.
10. Realice un diagrama de flujo de datos en `DFD` que calcule el salario semanal de un trabajador, considerando los siguientes rebajos al salario bruto por cargas sociales:

- Caja Costarricense del Seguro Social (CCSS) 8 %.
- Póliza de seguro ₡10 000,00.
- Impuesto de renta 13 %.

Además, considere que el salario por hora del trabajador es de ₡2000; que las horas extra se pagan doble; y que la jornada es de 40 horas semanales. Elabore los siguientes subprocesos:

- `LecturaDatos`: lee los datos del trabajador y el de sus horas laboradas.
- `CalculoSalarioBruto`: debe calcular el salario bruto.
- `CalculoDeducciones`: calcula las deducciones por cargas sociales.
- `CalculoSalarioNeto`: calcula el salario neto, que es el salario bruto menos las deducciones de cargas sociales.
- `MuestraInforme`: presenta, por pantalla, el informe de pago de salario con los siguientes datos: nombre del trabajador, horas extra trabajadas, monto por horas extra trabajadas, salario bruto, monto por deducciones y salario neto.

Al final de la muestra del informe de pago, debe consultarle al usuario si desea procesar otro empleado. Si la respuesta es afirmativa, solicite todos los datos nuevamente; de lo contrario, sale del algoritmo.

11. Elabore un pseudocódigo en `PSeInt` que llene un vector de 100 posiciones con números aleatorios entre 100 y 500. Una vez lleno, solicite al usuario un número por buscar en el vector. Valide que el número digitado por el usuario esté entre 100 y 500.

Luego de la búsqueda, el algoritmo le informa al usuario si el número buscado está o no en el vector; y, de estarlo, debe indicar cuántas veces se encuentra. Posteriormente, le consultará al usuario si desea hacer otra búsqueda y, de ser así, solicita de nuevo el número por buscar. De lo contrario, muestra el vector y sale del algoritmo. Debe implementar subprocesos para todas las acciones del vector.

12. Desarrolle un pseudocódigo en `PSeInt` que calcule el promedio de una matriz y encuentre el número mayor y el menor que hay en esta. Trabaje con una matriz de  $10 \times 10$ , pero el usuario podrá elegir sus dimensiones (solo elija una porque el arreglo es cuadrado). Valide que el número digitado por el usuario esté entre 1 y 10.

La matriz se debe llenar con números aleatorios del 100 al 500. Al final, el algoritmo debe mostrar la matriz de forma cuadrada y un informe con los siguientes datos:

- La sumatoria de toda la matriz
- El promedio de toda la matriz
- El número mayor.
- El número menor

Considere que debe emplear subprocesos para todas las acciones que involucren a la matriz.

13. Elabore un pseudocódigo en `PSeInt` que intercambie valores de dos celdas de un arreglo. El vector es de 10 posiciones y se llenará de nombres. Luego, se le solicita al usuario el número de dos celdas y se intercambia el contenido de estas entre sí. Debe hacer un subproceso para que el usuario llene el vector, otro para solicitar los números de las celdas, un tercero para hacer el intercambio, y el último para mostrar el arreglo. Primero, debe solicitar los nombres; luego, imprimir el vector; después, solicitar, las dos posiciones que desea cambiar; y, finalmente, mostrar de nuevo el vector.



## 5.9 Respuesta a los ejercicios de autoevaluación

En este apartado, encontrará las respuestas de los ejercicios de autoevaluación. Recuerde que el propósito de estos ejercicios es que usted evalúe su aprendizaje; por lo tanto, se le recomienda no revisarlo hasta no haber respondido los ejercicios propuestos con el conocimiento adquirido en su estudio previo.

1. *En este paso de parámetro se hace una copia de la variable original y su cambio de valor solo rige para el subproceso y no para el algoritmo de origen: paso por valor.*
2. *Tipo de paso de parámetro donde si, en el subproceso se cambia el valor de la variable, afecta también en el principal: paso por referencia.*
3. *Pseudocódigo en PSeInt que lea un nombre digitado por el usuario. Luego, muestre el saludo "Hola" y el nombre digitado.*

```
1  Algoritmo Saludo
2      Definir Nombre Como Caracter;
3      Nombre="X";
4
5      Escribir "Digite un nombre.";
6      Leer Nombre;
7      MuestraSaludo(Nombre);
8
9  FinAlgoritmo
10
11
12 SubProceso MuestraSaludo(Nomb Por Referencia)
13     Escribir "Hola ",Nomb;
14 FinSubProceso
```

FIGURA 5.56. Pseudocódigo en PSeInt que lee un nombre digitado por el usuario; y, luego, muestra el saludo "Hola" y el nombre digitado

4. *Pseudocódigo que modifica el valor de dos variables en un procedimiento y los muestra por pantalla.*

```
1  Algoritmo ModificacionVariables
2      Definir VariableReal Como Real;
3      Definir VariableCaracter Como Caracter;
4
5      VariableReal=0;
6      VariableCaracter="X";
7      |
8      Escribir "INGRESO DE VALORES";
9      Escribir "Digite un valor para la variable de tipo real.";
10     Leer VariableReal;
11     Escribir "Digite un valor para la variable de tipo caracter.";
12     Leer VariableCaracter;
13
14     Escribir "";
15     Escribir "VALORES ORIGINALES: antes del procedimiento";
16     Escribir "Variable de tipo real: ",VariableReal;
17     Escribir "Variable de tipo caracter: ",VariableCaracter;
18
19
20     ModificaVariables(VariableReal,VariableCaracter);
21
22     Escribir "";
23     Escribir "VALORES FINALES: luego del procedimiento";
24     Escribir "Variable de tipo real: ",VariableReal;
25     Escribir "Variable de tipo caracter: ",VariableCaracter;
26
27 FinAlgoritmo
```

FIGURA 5.57. Procedimiento que muestra el comportamiento de dos variables al ser modificadas en un subproceso. Una de ellas enviada como parámetro Por Valor; y la otra, Por Referencia

Por una parte, la variable que cambió su valor al regresar al algoritmo principal es la que se pasó por referencia. Por otra parte, la que regresó a su valor inicial fue la que se pasó por valor.

5. Diagrama de flujo que resta dos variables y su resultado se asigna a una tercera variable.

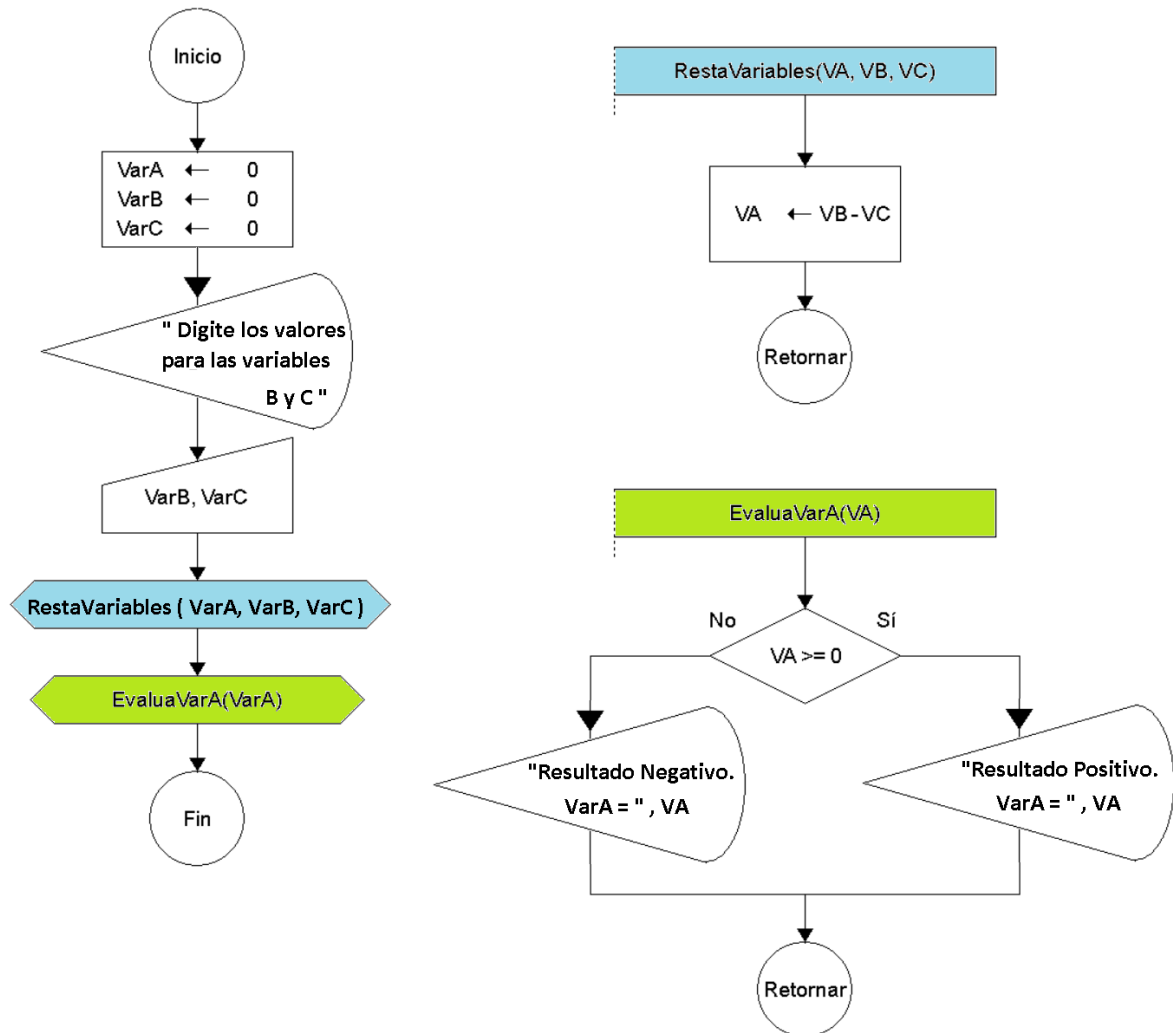


FIGURA 5.58. DFD que lee el valor de dos variables y calcula su diferencia, asignando el resultado a una tercera variable e indica si el valor es positivo o negativo

6. *Pseudocódigo en PSeInt que evalúa en una función si un número es múltiplo de 3.*

```
1  Algoritmo MultiploDe3
2      Definir Dato Como Entero;
3      Dato=0;
4
5      Escribir "Digite un valor.";
6      Leer Dato;
7      Si FuncEvaluaDato(Dato)=Verdadero Entonces
8          | Escribir "El dato ",Dato," es múltiplo de 3.";
9      SiNo
10         | Escribir "El dato ",Dato," NO es múltiplo de 3.";
11     FinSi
12 FinAlgoritmo
13
14
15 Funcion EvalDato=FuncEvaluaDato (Dat Por Referencia)
16     Definir EvalDato Como Logico;
17     EvalDato=Verdadero;
18
19     Si (Dat Mod 3 != 0) Entonces
20         | EvalDato=Falso;
21     FinSi
22 FinFuncion
```

FIGURA 5.59. Pseudocódigo en PSeInt que evalúa, a través de una función, si un número es múltiplo de 3



7. Pseudocódigo en PSeInt que convierte un monto en colones a dólares, solicitando el tipo de cambio.

```
1  Algoritmo TipoCambio
2      Definir Colon,Cambio Como Real;
3      Colon=0;
4      Cambio=0;
5      SolicitudDatos (Colon,Cambio) ;
6      Escribir "Con ",Colon," colones, puede comprar ",Sin Saltar;
7      Escribir CalculoDolares (Colon,Cambio) ," dólares.";
8  FinAlgoritmo
9
10 SubProceso SolicitudDatos (Col Por Referencia,Camb Por Referencia)
11     Escribir "Digite la cantidad de colones que desea cambiar.";
12     Leer Col;
13     Escribir "Digite el tipo de cambio.";
14     Leer Camb;
15 FinSubProceso
16
17 SubProceso Dolares=CalculoDolares (Col Por Valor,Camb Por Valor)
18     Definir Dolares Como Real;
19     Dolares=0;
20     Dolares=Col/Camb;
21 FinSubProceso
```

FIGURA 5.60. Pseudocódigo en PSeInt que solicita el tipo de cambio y convierte un monto en colones a dólares

8. *Modifique el ejercicio anterior; pero en este caso, implemente la función SolicitudDatos en una instrucción de asignación.*

```
1  Algoritmo TipoCambio
2      Definir Colon,Cambio,Dolar Como Real;
3      Colon=0;
4      Cambio=0;
5      Dolar=0;
6      SolicitudDatos(Colon,Cambio);
7      Dolar=CalculoDolares(Colon,Cambio);
8      Escribir "Con ",Colon," colones, puede comparar ",Dolar," dólares.";
9  FinAlgoritmo
10
11 SubProceso SolicitudDatos(Col Por Referencia,Camb Por Referencia)
12     Escribir "Digite la cantidad de colones que desea cambiar.";
13     Leer Col;
14     Escribir "Digite el tipo de cambio.";
15     Leer Camb;
16 FinSubProceso
17
18 SubProceso Dolares=CalculoDolares(Col Por Valor,Camb Por Valor)
19     Definir Dolares Como Real;
20     Dolares=0;
21     Dolares=Col/Camb;
22 FinSubProceso
```

FIGURA 5.61. Pseudocódigo en PSeInt que solicita el tipo de cambio y convierte un monto en colones a dólares, solicitando los datos a través de una instrucción de asignación

9. *Elabore un diagrama de flujo en DFD que solicite el nombre de una persona y envíe un saludo: "Hola," más el Nombre. El nombre debe aceptarlo en un subproceso y la muestra del mensaje en otro.*

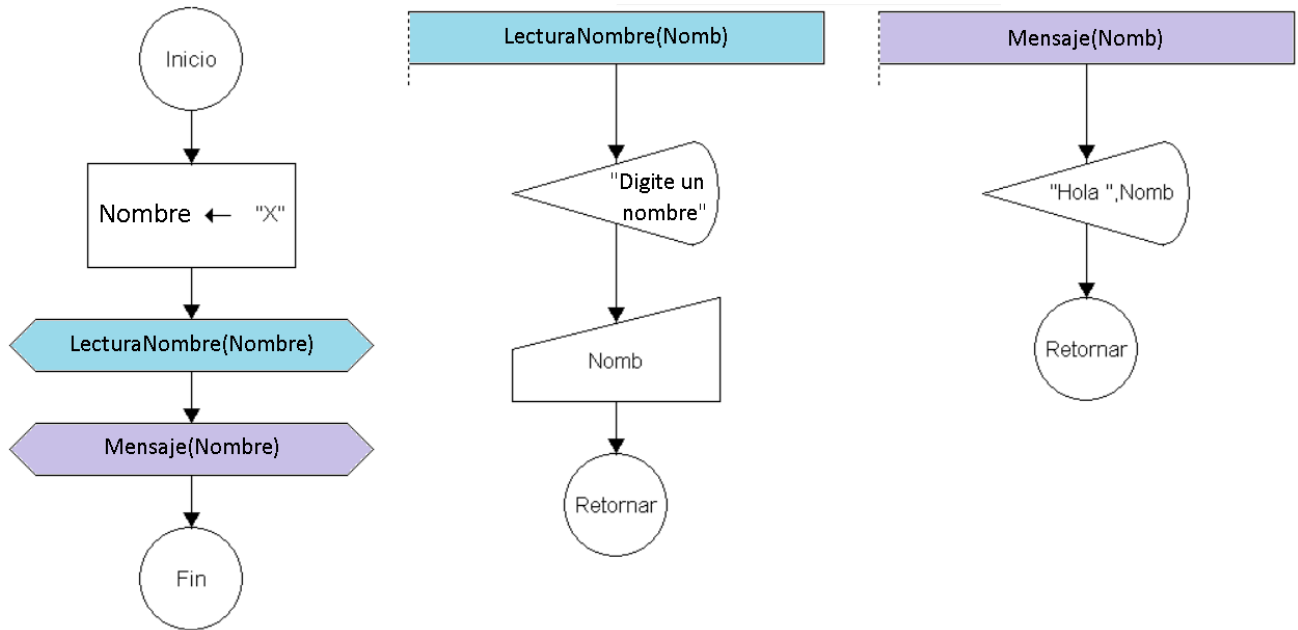


FIGURA 5.62. Diagrama de flujo en DFD que solicita el nombre de una persona y despliega un saludo

10. Diagrama de flujo del algoritmo en DFD que calcula salario con base en una serie de parámetros.

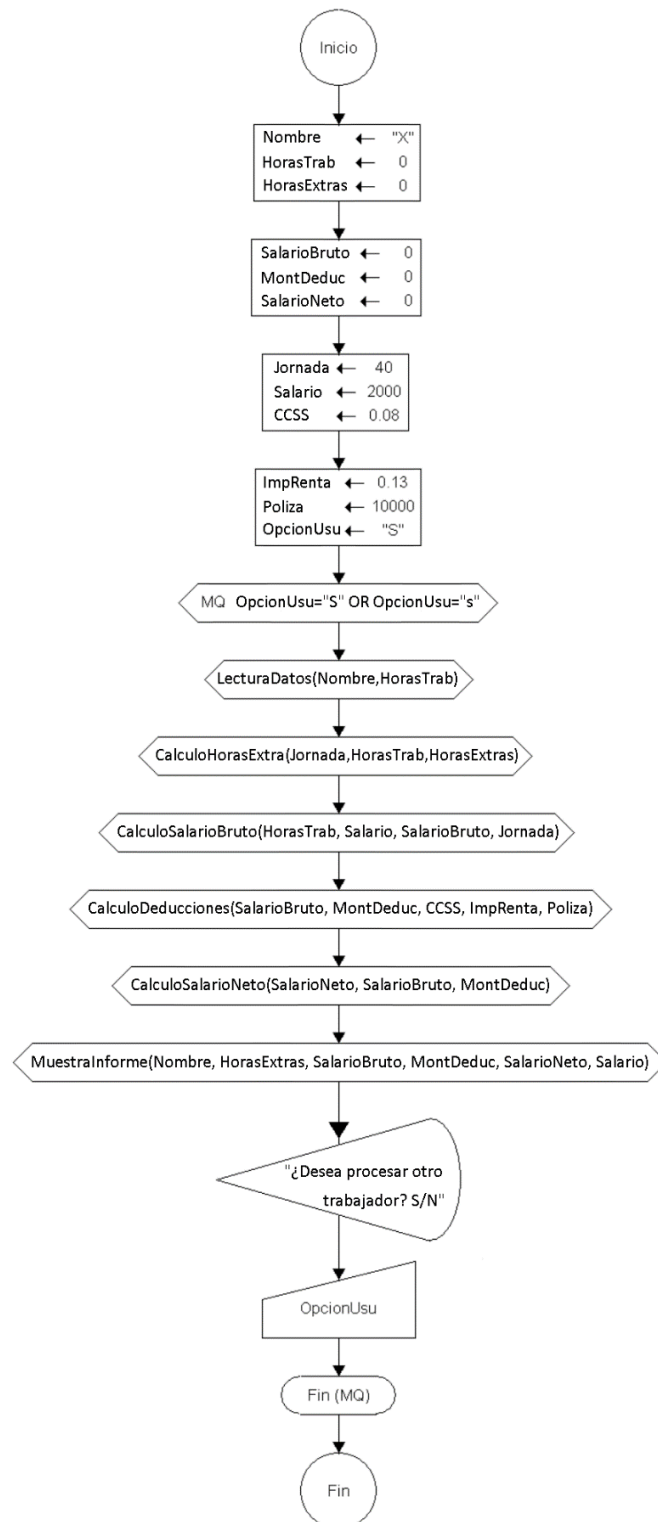
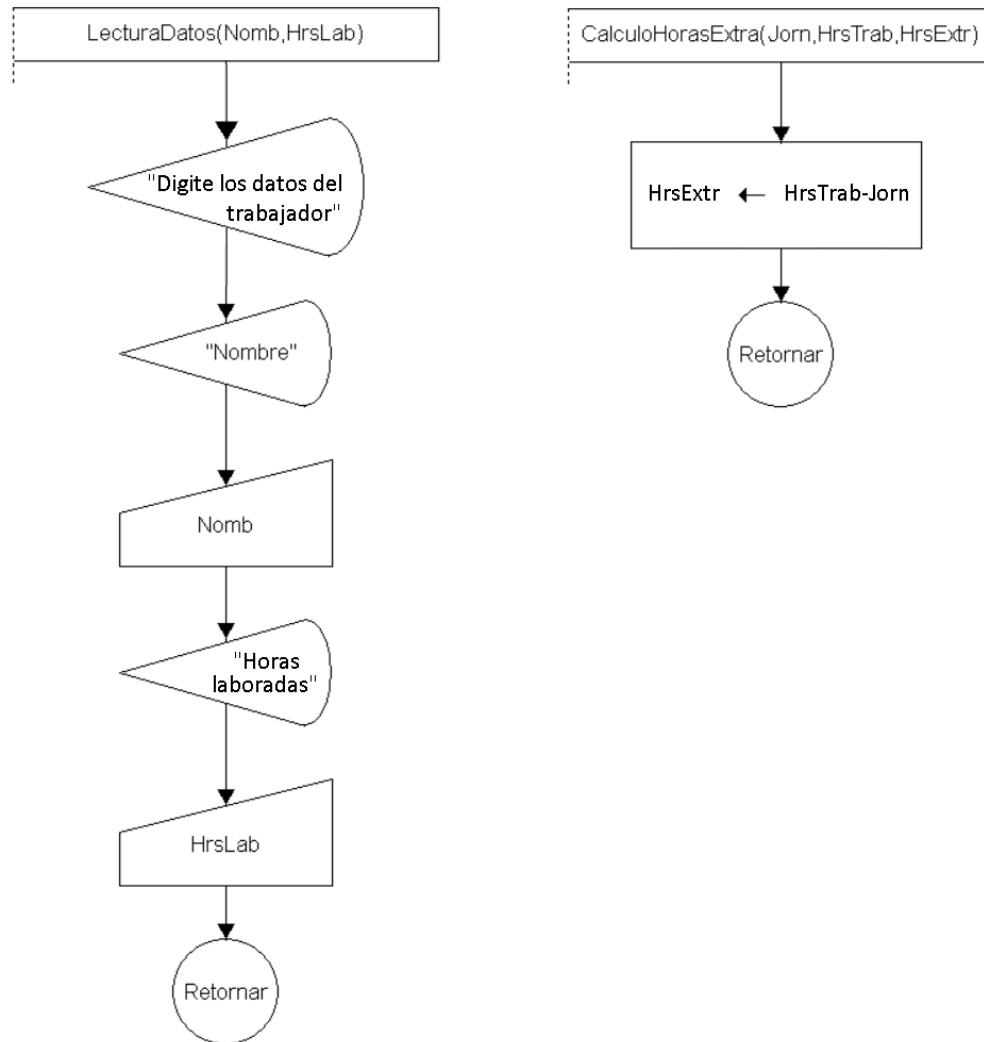
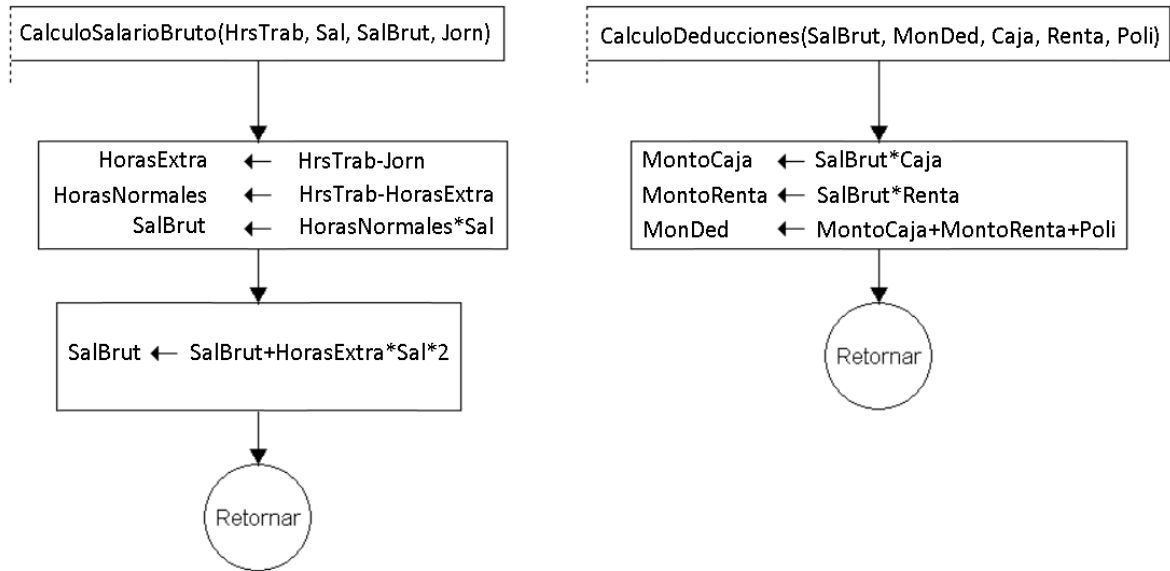


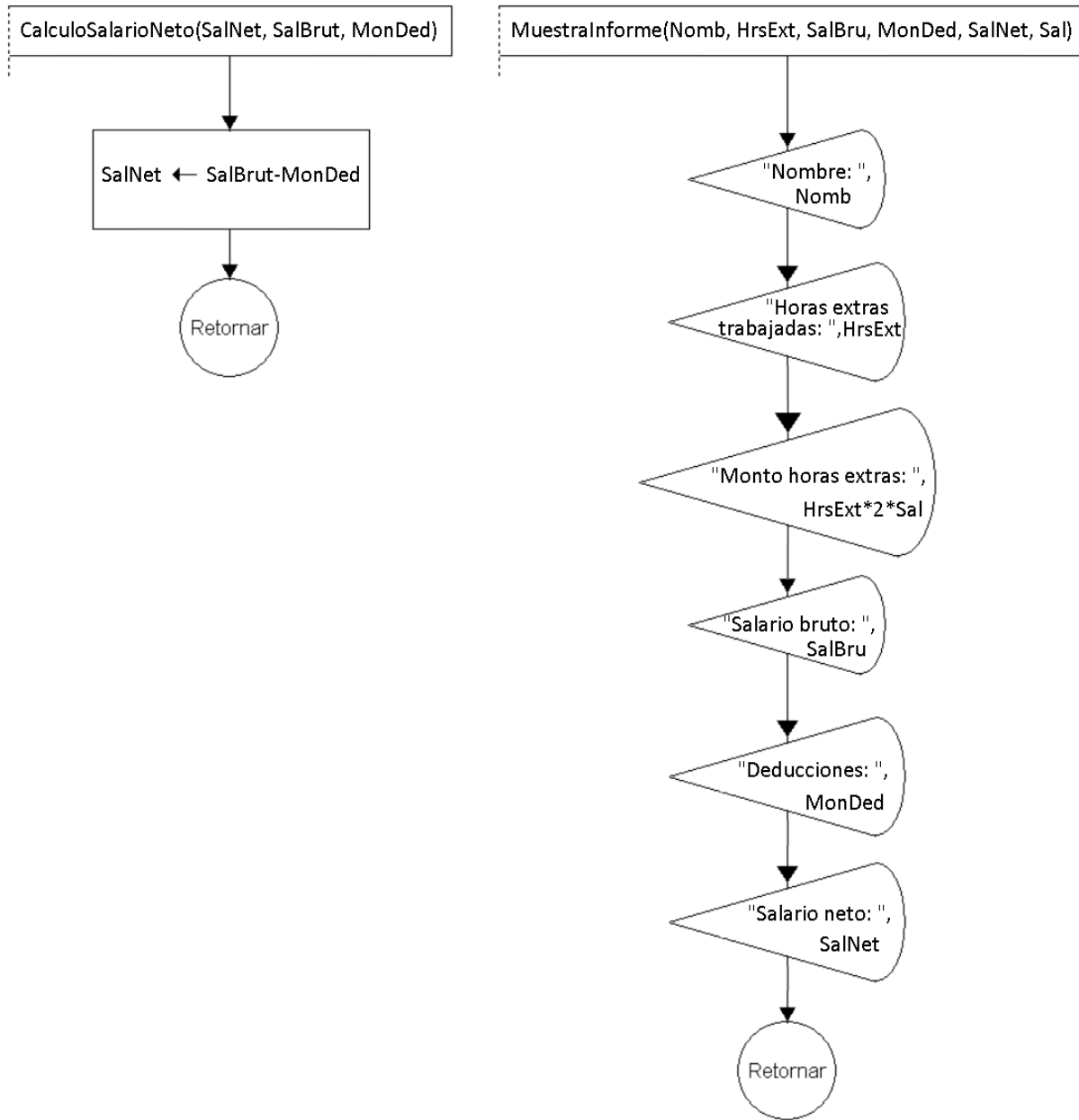
FIGURA 5.63. Diagrama de flujo principal en DFD que calcula el salario semana con base en una serie de parámetros



**FIGURA 5.64.** Diagrama de flujo de los subprocesos `LecturaDatos` y `CalculoHorasExtra` del algoritmo de cálculo de salario de la figura 5.63



**FIGURA 5.65.** Diagrama de flujo de los subprocesos *CalculoSalarioBruto* y *CalculoDeducciones* del algoritmo de cálculo de salario de la figura 5.63



**FIGURA 5.66.** Diagrama de flujo subprocesos **CalculoSalarioNeto** y **MuestraInforme** del algoritmo de cálculo de salario de la figura 5.63

11. *Pseudocódigo en PSeInt que llena un vector con número entre 100 y 500, y busca uno indicado por el usuario.*

```
1  Proceso BusquedaenVectorConSubProc
2
3      //Declaraciones de variables y vector
4      Definir Vector, NumeroBuscado, CantidadCoincidencias Como Entero;
5      Definir Opcionusuario Como Caracter;
6
7      //Dimensionamiento del arreglo
8      Dimension Vector(100);
9
10     //Inicializaciones de variables
11     NumeroBuscado=0;
12     CantidadCoincidencias=0;
13     Opcionusuario="S";
14
15     LlenaVector(Vector);
16     Mientras Opcionusuario="S" o Opcionusuario="s" Hacer
17
18         //Solicitud, lectura y validación del número a buscar
19         SolicitaNumero(NumeroBuscado);
20
21         //Búsqueda del número en el vector
22         CantidadCoincidencias=Busqueda(Vector,NumeroBuscado);
23
24
25         //Muestra de resultados
26         MuestraResultados(CantidadCoincidencias,NumeroBuscado);
27
28         Escribir "¿Desea hacer otra búsqueda? S/N";
29         Leer Opcionusuario;
30     FinMientras
31     MuestraVector(Vector);
32     Escribir "Presione cualquier tecla para salir.";
33     Esperar Tecla;
34
35 FinProceso
```

FIGURA 5.67. Pseudocódigo principal en PSeInt del algoritmo búsqueda en un vector con números entre 100 y 500, uno indicado por el usuario



```

37 SubProceso LlenaVector(Vec)
38     Definir Celda Como Entero;
39     Celda=0;
40     Para Celda<-0 Hasta 99 Con Paso 1 Hacer
41     |     Vec(Celda)=azar(401)+100;
42     FinPara
43 FinSubProceso
44
45 SubProceso MuestraVector(Vec)
46     Definir Celda Como Entero;
47     Celda=0;
48     Escribir "***** VECTOR *****";
49     Para Celda<-0 Hasta 99 Con Paso 1 Hacer
50     |     Escribir Vec(Celda) , " ",Sin Saltar;
51     FinPara
52 FinSubProceso
53
54 SubProceso SolicitaNumero(NumBus Por Referencia)
55     Repetir
56     |     Escribir "";
57     |     Escribir "Digite un número a buscar en el vector.";
58     |     Leer NumBus;
59     |     Si NumBus<100 O NumBus>500 Entonces
60     |     |     Escribir "Error. El número a buscar debe estar entre 100 y 500.";
61     |     FinSi
62     Hasta Que NumBus>=100 Y NumBus<=500
63 FinSubProceso

```

FIGURA 5.68. Subprocesos LlenaVector, MuestraVector y SolicitaNumero del algoritmo búsqueda en un vector de la figura 5.67

```

65 SubProceso CantCoinc=Busqueda(Vec,NumBus)
66     Definir Celda,CantCoinc Como Entero;
67     Celda=0;
68     CantCoinc=0;
69     Escribir "Búsqueda del número ",NumBus," en el vector.";
70     Para Celda<-0 Hasta 99 Con Paso 1 Hacer
71         Si NumBus=Vec(Celda) Entonces
72             CantCoinc=CantCoinc+1;
73         FinSi
74     FinPara
75 FinSubProceso
76
77 SubProceso MuestraResultados(CantCoinc,NumBus)
78     Limpiar Pantalla;
79     Escribir "*****INFORME DE BÚSQUEDA*****";
80     Si CantCoinc>0 Entonces
81         Escribir "El número ",NumBus," está ",CantCoinc," veces en el vector.";
82     Sino
83         Escribir "El número ",NumBus," NO está en el vector.";
84     FinSi
85     Escribir "";
86 FinSubProceso

```

FIGURA 5.69. Subprocesos Busqueda, MuestraResultados del algoritmo búsqueda en un vector de la figura 5.67

12. Pseudocódigo en PSeInt que calcula el promedio y encuentra el mayor y menor de los números de una matriz.

```
1  Proceso PromedioMayorMenorMatriz
2
3      //Declaraciones//
4      Definir Matriz Como Entero;
5      Definir Sumatoria, NumMayor, NumMenor, TamanoMatriz Como Entero;
6      Definir Promedio Como Real;
7
8      //Dimensionamiento de la matriz//
9      Dimension Matriz(10,10);
10
11     //Inicialización de variables//
12     Promedio=0;
13     Sumatoria=0;
14     TamanoMatriz=0;
15
16     //Dimensionamiento de la matriz por el usuario//
17     DimensionaMatriz(TamanoMatriz);
18
19     //Llena la matriz//
20     LlenaMatriz(Matriz, TamanoMatriz);
21
22     //Calcula la sumatoria de la matriz//
23     Sumatoria=SumatoriaMatriz(Matriz, TamanoMatriz);
24
25     //Calcula el promedio//
26     Promedio=Sumatoria/(TamanoMatriz*TamanoMatriz);
27
28     //Determina el número mayor y menor de la matriz//
29     DeterminaMayMen(Matriz, NumMayor, NumMenor, TamanoMatriz);
30
31     //Muestra la matriz y el informe//
32     MuestraMatriz(Matriz, TamanoMatriz);
33     MuestraInforme(Sumatoria, Promedio, NumMayor, NumMenor);
34
35 FinProceso
36
37 SubProceso DimensionaMatriz(TamMat Por Referencia)
38     Escribir "Bienvenid@";
39     Repetir
40         Escribir "Digite el tamaño de la matriz.";
41         Leer TamMat;
42         Si TamMat<1 O TamMat>10 Entonces
43             Limpiar Pantalla;
44             Escribir "Error. El tamaño de la matriz debe estar entre 1 y 10.";
45         FinSi
46     Hasta Que TamMat>=1 Y TamMat<=10
47 FinSubProceso
```

FIGURA 5.70. Pseudocódigo principal de un algoritmo en PSeInt que calcula el promedio y encuentra el mayor y el menor número en una matriz

```

49 SubProceso LlenaMatriz(Mat,TamMat)
50     Definir Fila, Columna Como Entero;
51     Fila=0;
52     Columna=0;
53     Para Fila=0 Hasta TamMat-1 Con Paso 1 Hacer
54         Para Columna<-0 Hasta TamMat-1 Con Paso 1 Hacer
55             Mat[Fila,Columna]= azar(401)+100;
56         FinPara
57     FinPara
58 FinSubProceso
59
60 SubProceso Sumato=SumatoriaMatriz(Mat,TamMat)
61     Definir Fila, Columna,Sumato Como Entero;
62     Fila=0;
63     Columna=0;
64     Sumato=0;
65     Para Fila=0 Hasta TamMat-1 Con Paso 1 Hacer
66         Para Columna<-0 Hasta TamMat-1 Con Paso 1 Hacer
67             Sumato=Sumato+Mat[Fila,Columna];
68         FinPara
69     FinPara
70 FinSubProceso

```

FIGURA 5.71. Procedimiento LlenaMatriz y función SumatoriaMatriz del algoritmo de la figura 5.70

```

72 SubProceso DeterminaMayMen(Mat,NumMay Por Referencia,NumMen Por Referencia,TamMat)
73     Definir Fila, Columna Como Entero;
74     Fila=0;
75     Columna=0;
76     NumMay=Mat[0,0];
77     NumMen=Mat[0,0];
78     Para Fila=0 hasta TamMat-1 con paso 1 Hacer
79         Para Columna=0 hasta TamMat-1 con paso 1 Hacer
80             Si Mat[Fila,Columna]>NumMay Entonces
81                 NumMay<-Mat[Fila,Columna];
82             FinSi
83
84             Si Mat[Fila,Columna]<NumMen Entonces
85                 NumMen=Mat[Fila,Columna];
86             FinSi
87         FinPara
88     FinPara
89 FinSubProceso

```

FIGURA 5.72. Procedimiento DeterminaMayMen del algoritmo de la figura 5.70

```

91 SubProceso MuestraMatriz(Mat,TamMat)
92     Definir Fila, Columna Como Entero;
93     Fila=0;
94     Columna=0;
95     Para Fila=0 Hasta TamMat-1 Con Paso 1 Hacer
96         Para Columna<=0 Hasta TamMat-1 Con Paso 1 Hacer
97             Escribir Mat[Fila,Columna]," ",Sin Saltar;
98         FinPara
99         Escribir "";
100     FinPara
101 FinSubProceso
102
103 SubProceso MuestraInforme(Sumato,Prom,NumMay,NumMen)
104     Escribir "";
105     Escribir "-----";
106     Escribir "INFORME";
107     Escribir "La suma de toda la matriz es de:", Sumato;
108     Escribir "El promedio de toda la matriz es de:", Prom;
109     Escribir "El número mayor es: ", NumMay;
110     Escribir "El número menor es: ", NumMen;
111     Escribir "-----";
112     Escribir " ";
113 FinSubProceso

```

**FIGURA 5.73.** Subprocesos MuestraMatriz y MuestraInforme algoritmo de trabajos con una matriz del algoritmo de la figura 5.70

13. *Pseudocódigo en PSeInt que intercambia los valores de dos celdas de un vector.*

```
1  Algoritmo CambioEnVector
2      Definir Vector Como Caracter;
3      Definir CeldaCambio1,CeldaCambio2 Como Entero;
4      Dimension Vector(10);
5
6      CeldaCambio1=0;
7      CeldaCambio2=0;
8
9      LlenaVector(Vector);
10     MuestraVector(Vector);
11     SolicitudCeldas(CeldaCambio1,CeldaCambio2);
12     CambiaContenido(Vector,CeldaCambio1,CeldaCambio2);
13     MuestraVector(Vector);
14
15 FinAlgoritmo
```

FIGURA 5.74. Pseudocódigo en PSeInt que intercambia los valores de dos celdas de un vector lleno de nombres

```
17 SubProceso LlenaVector(Vec)
18     Definir Celda Como Entero;
19     Celda=0;
20     Escribir "Digite 10 nombres.";
21     Para Celda=0 Hasta 9 Con Paso 1 Hacer
22         Escribir Celda+1,": ",Sin Saltar;
23         Leer Vec(Celda);
24     FinPara
25 FinSubProceso
26
27 SubProceso MuestraVector(Vec)
28     Definir Celda Como Entero;
29     Celda=0;
30     Escribir "*****Vector de nombres*****";
31     Para Celda=0 Hasta 9 Con Paso 1 Hacer
32         Escribir Celda+1,". ",Vec(Celda);
33     FinPara
34 FinSubProceso
```

FIGURA 5.75. Procedimientos LlenaVector y MuestraVector del algoritmo de la figura 5.74

```

36 SubProceso SolicitudCeldas(CelCamb1 Por Referencia,CelCamb2 Por Referencia)
37     Escribir "Digite el número de las dos celdas entre las que desea intercambiar."
38     Escribir "Celda: ",Sin Saltar;
39     Leer CelCamb1;
40     CelCamb1=CelCamb1-1;
41     Escribir "Celda: ",Sin Saltar;
42     Leer CelCamb2;
43     CelCamb2=CelCamb2-1;
44 FinSubProceso
45
46 SubProceso CambiaContenido(Vec,CelCamb1,CelCamb2)
47     Definir CeldaAux Como Caracter;
48     CeldaAux="X";
49     CeldaAux=Vec(CelCamb1);
50     Vec(CelCamb1)=Vec(CelCamb2);
51     Vec(CelCamb2)=CeldaAux;
52 FinSubProceso

```

**FIGURA 5.76.** Procedimientos SolicitudCeldas y CambiaContenido del algoritmo de la figura 5.74



## 5.10 Referencias

Hernández Yáñez, Luis. (2013-2014). *Apuntes de clase de la asignatura Fundamentos de la programación*. Universidad Complutense de Madrid.

Recuperado de <https://www.fdi.ucm.es/profesor/luis/fp/fp.pdf>