Unidad didáctica 3071

Contenidos

Índice de ejemplos

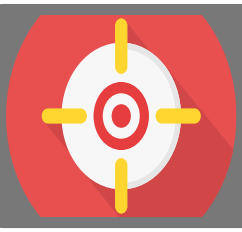
Índice de figuras

Capítulo V.

Subprocesos

Objetivo general

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



Objetivos específicos

Conocer las características de los subprocesos.

Identificar los tipos de subprocesos que se implementan en los algoritmos.

Objetivos específicos

Identificar los tipos de pasos de parámetros.

Introducción

En esta sección del libro se habla de los subprocesos y de 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 todo 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 concepto se apoya en ejemplos que evidencian la implementación de los subprocesos en PSeInt y 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.

Este capítulo se enfoca en la implementación de subprocesos en algoritmos en pseudocódigo y diagramas de flujo. Lo primero que se debe tener claro es el concepto de subprocesos y su esquema de trabajo en los algoritmos.

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

Finalmente, se analizará 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.

1.1 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 se está dividiendo el algoritmo principal en partes, pero 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:

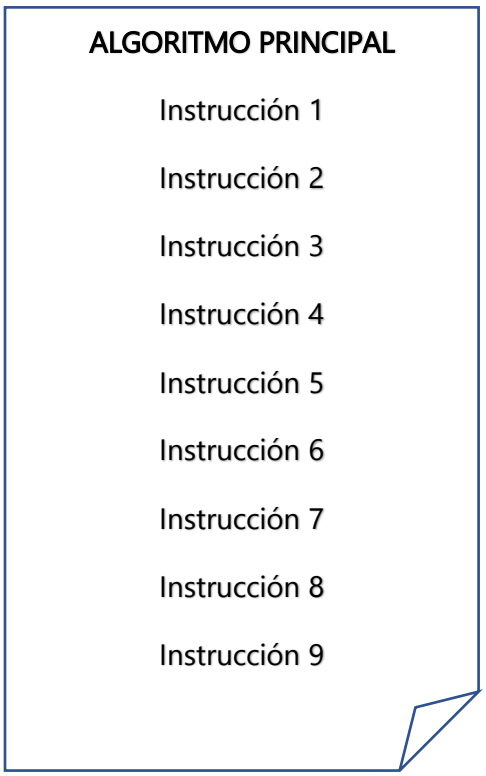


Figura 1 Esquema de un algoritmo principal

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:

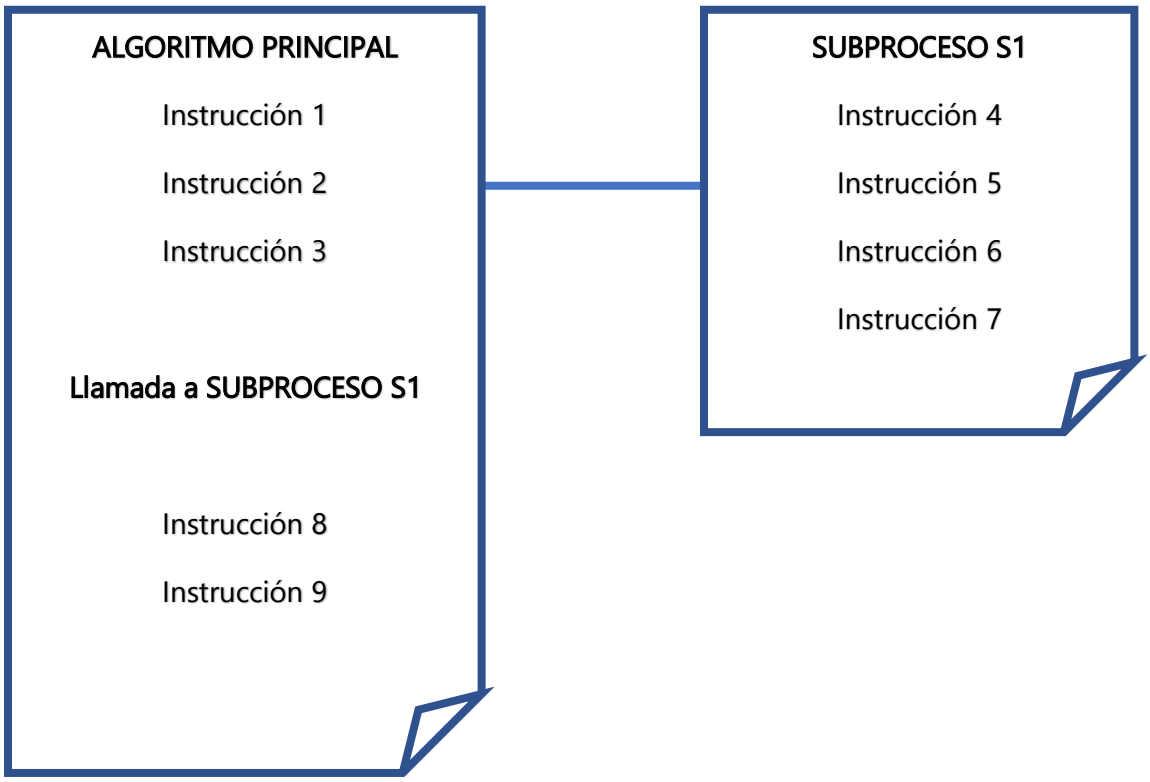


Figura 2 Esquema de una algoritmo con un subproceso

En la representación gráfica podemos notar como las instrucciones **4, 5, 6, y 7** no forman parte del algoritmo principal, sino que son parte del **Subproceso llamado 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), la misma 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**, es fundamental aclarar 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.

1.1.1 Subprocesos en PSeInt

A continuación, se presenta una comparación de dos códigos en PSeInt, uno no emplea subprocesos y otro si lo utiliza, 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 3 Pseudocódigo de división de números sin subprocesos

Ahora se muestra el mismo algoritmo, pero empleando 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 4 Pseudocódigo de división de números con subprocesos

Notemos que en las **líneas 11 y 12** es donde 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 se declara en el algoritmo principal (que es el único), pero en el segundo ejemplo 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.

Nota: En PSeInt tanto el algoritmo principal como los subprocesos se visualizan en la misma pantalla, además, el botón para insertar un subproceso es el siguiente:

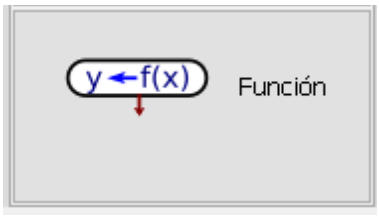


Figura 5 Botón para crear un subproceso en PSeInt

Al presionarlo se generará un subproceso llamado de forma genérica Función, pero se recomienda cambiarla por SubProceso o por SubAlgoritmo:

```
Funcion Nombre ()

FinFuncion


SubAlgoritmo Nombre ()

FinSubAlgoritmo


SubProceso Nombre ()

FinSubProceso
```

Figura 6 Ejemplos de nombres de subprocesos en PSeInt

1.1.2 Subprocesos en DFD

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 7 Botón para crear un subproceso en DFD

Esto abrirá una nueva ventana con el inicio del diagrama para el subproceso, aquí se trabajará la solución lógica para el subproceso, este primer diagrama luce de la siguiente manera:

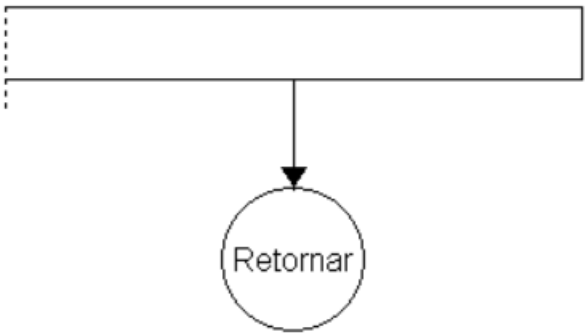


Figura 8 Estructura de un subproceso en DFD

Al presionar 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:

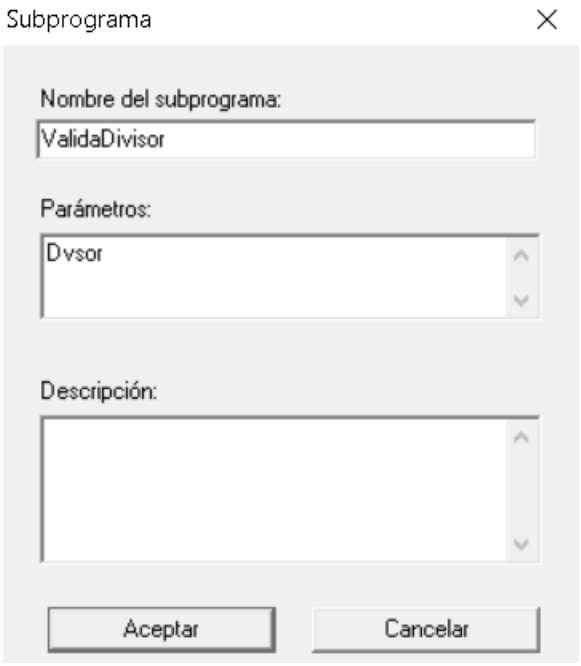


Figura 9 Ventana de datos de un subproceso en DFD

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

Figura 10 Botones para navegar entre subprocesos y diagrama principal en DFD

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

Figura 11 Botón para agregar una llamada a un subproceso en DFD

Al oprimir el botón anterior, se insertará en el diagrama el siguiente símbolo:

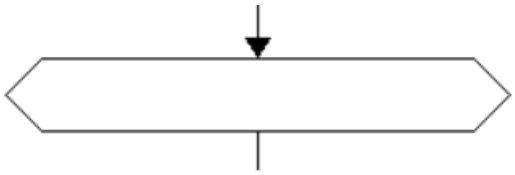


Figura 12 Símbolo de subproceso en DFD

Si presionamos dos veces (doble clic) el símbolo de subprograma, se desplegará una ventana donde debemos colocar el nombre del subproceso y los parámetros o argumentos que le enviaremos, la misma luce de la siguiente forma:

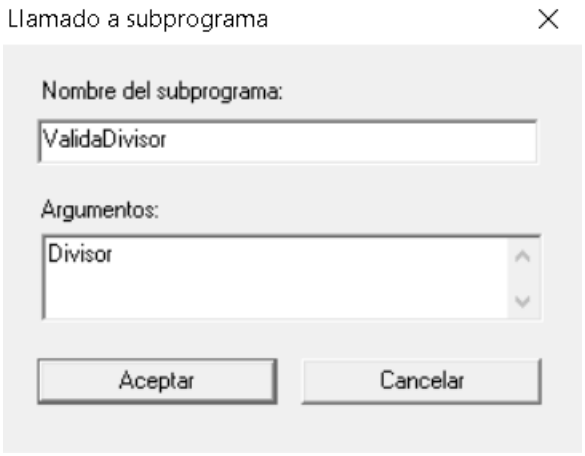


Figura 13 Ventana de datos de llamada a subproceso en DFD

Teniendo todos los aspectos anteriores claros, se presentan a continuación los diagramas de flujo del ejemplo de la división, primero el algoritmo principal y posteriormente los dos subprocesos:

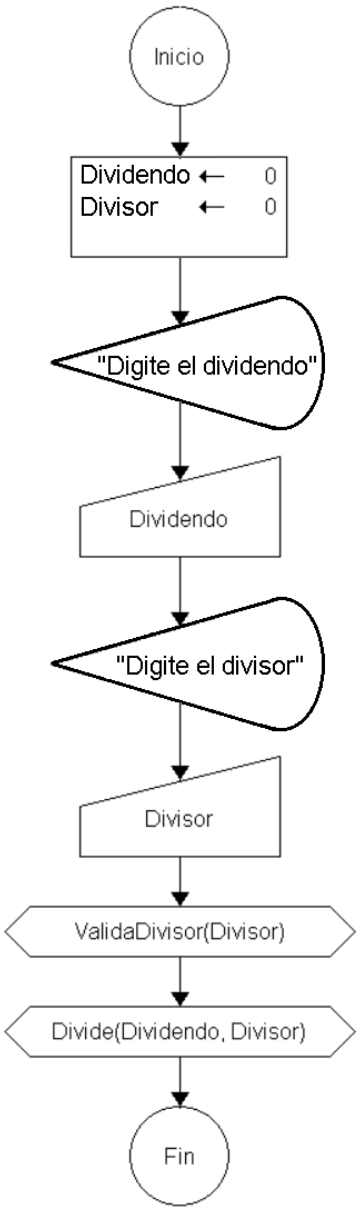


Figura 14 Diagrama de división de números con subprocesos en DFD

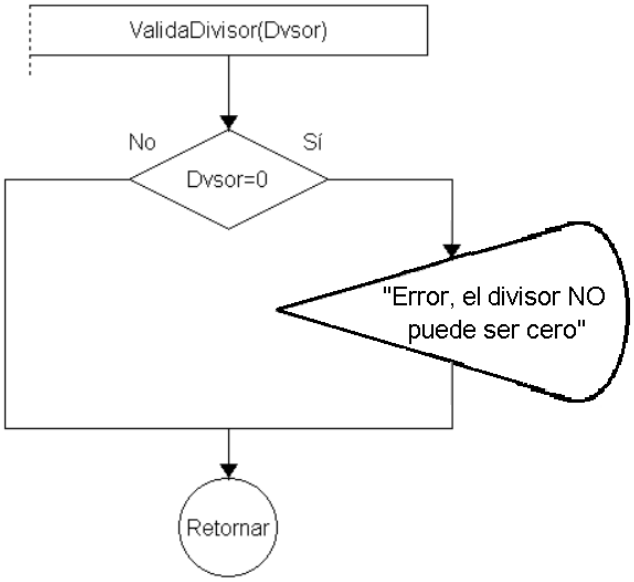


Figura 15 Detalle del subproceso en DFD de validación de divisor

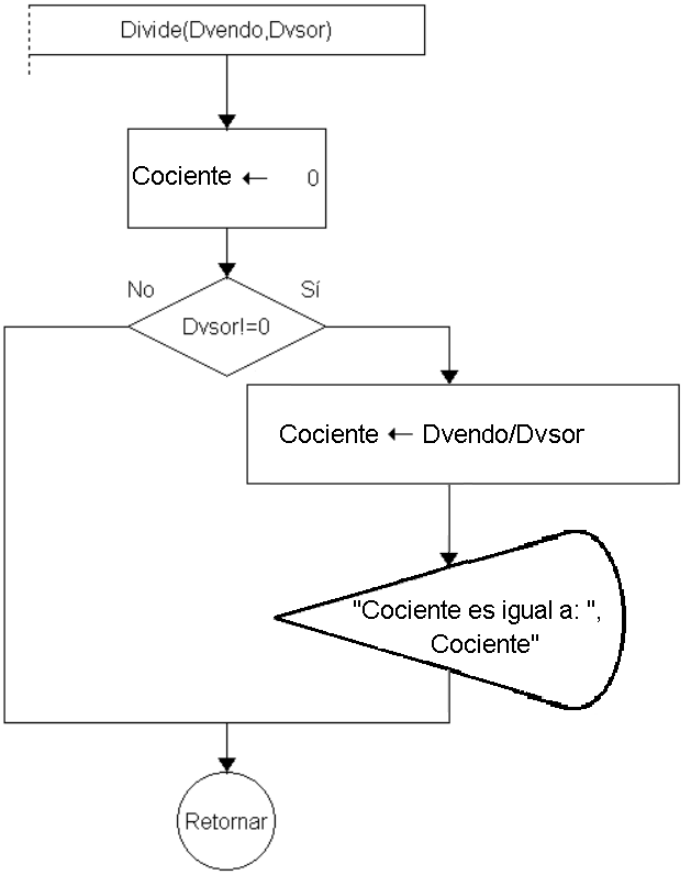


Figura 16 Detalle del subproceso en DFD que divide dos números

1.2 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.

1.3 Llamada o invocación de un subproceso

La manera de llamar a un subproceso es **escribir su nombre en el algoritmo de origen**, ya sea el 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.

1.4 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 **SubProc (Numero1, Numero2, Numero3);** se deben recibir en ese mismo orden, así: **SubProc(Num1, Num2, Num3).**

Un detalle importante 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 de llaman diferente para diferenciarlos de los originales, y dentro del subproceso se utilizarán Num1, Num2 y Num3 en las instrucciones; analicemos el siguiente caso:

```
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 17 Nombres de parámetros en un algoritmo principal y un subproceso 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 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.

1.4.1 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**.

1.4.1.1 Parámetros por valor en PSeInt

En **PSeInt** para indicar que un parámetro se pasa por valor se debe digitar la instrucción **Por Valor** a la derecha de cada variable que se recibe en el subproceso, pero también; **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 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 **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 Ana y 10** respectivamente. Ahora, en el subproceso se reciben ambos parámetros **por valor**, es decir como copias de las variables que se enviaron desde el algoritmo principal.

Dentro del subproceso, se asignan nuevos valores a las variables, **NumUsu** pasa a tener **44** y **Nomb** tendrá **Elena**, luego se muestran estos nuevos valores y se regresa el flujo de datos al algoritmo principal, este cambio en las variables solo rige para el subproceso. En las líneas 16 y 17 se muestran los valores de **NumeroUsuario y Nombre** , los cuáles 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 19 Salida de pantalla del tratamiento de paso de parámetros Por Valor en PSeInt

1.4.1.2 Parámetros por valor en DFD

En **DFD**, al momento de 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:

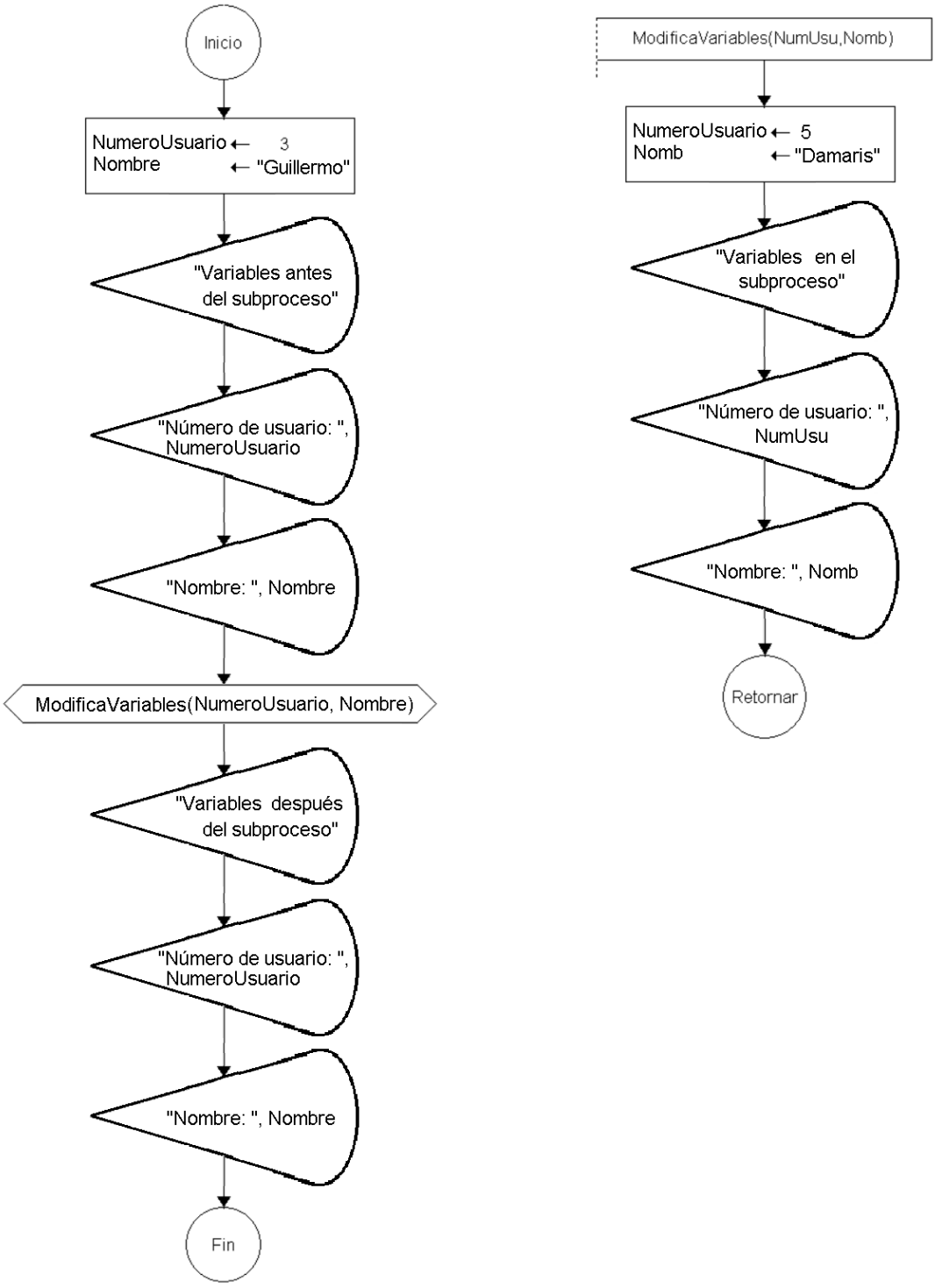


Figura 20 Paso de parámetros Por Valor en DFD

Vemos que las variables **NumeroUsuario y Nombre** tienen los valores **3 y "Guillermo"** respectivamente, luego se da la muestra de esos valores antes de la llamada al subproceso **ModificaVariables**.

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

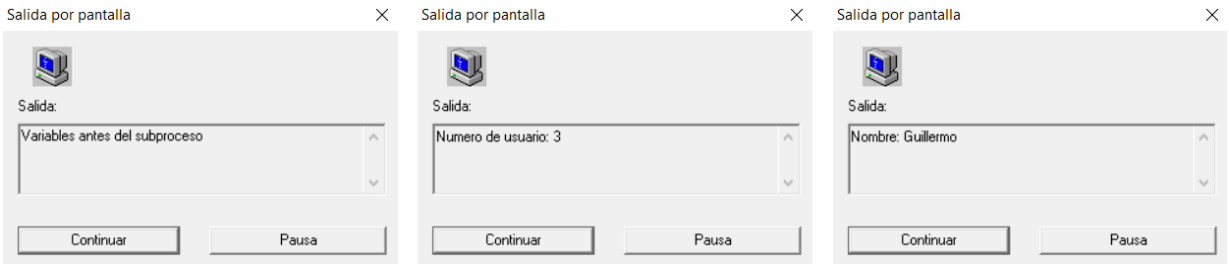


Figura 21 Salida de pantalla de parámetros Por Valor antes del subproceso en DFD

En la llamada al subproceso 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:

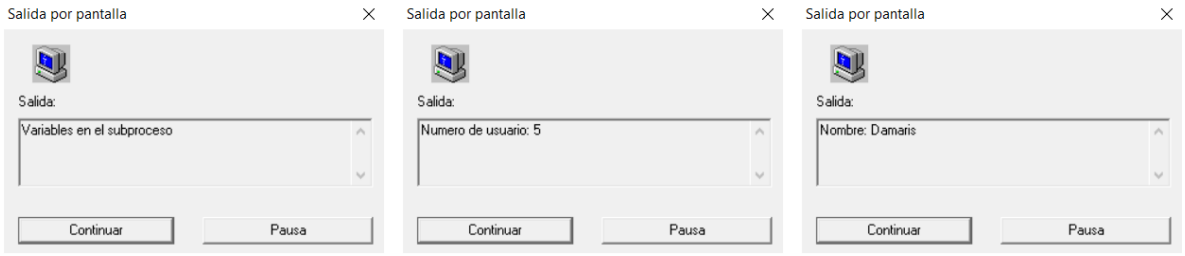


Figura 22 Salida de pantalla de parámetros Por Valor en el subproceso en DFD

Una vez finalizado el subproceso ModificaVariables, nos regresamos al algoritmo principal y se muestran de nuevo los valores de las variables NumeroUsuario y Nombre, pero con los nuevos datos, es decir ya no tienen los inciales (3 y "Guillermo") sino que poseen **5 y "Damaris"**, esto podemos verlo en las siguientes salidas:

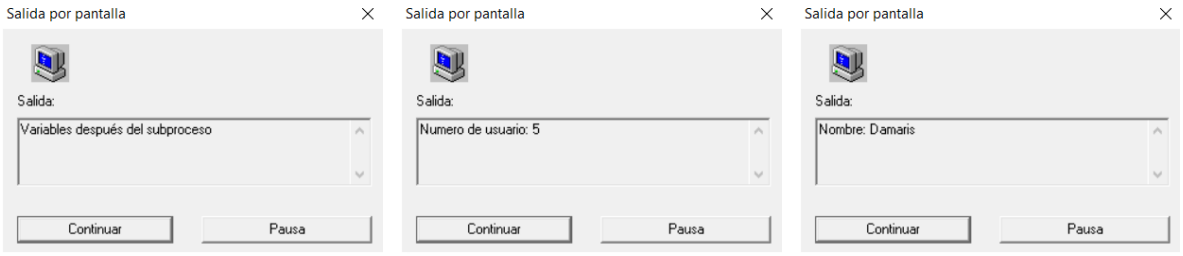


Figura 23 Salida de pantalla de parámetros Por Valor luego del subproceso en DFD

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.

1.4.2 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 regirá también en el algoritmo de origen.

1.4.2.1 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 siguiente ejemplo donde 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 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**, luego 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 de **Nomb** lo hará **solo 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 si**, esta secuencia la podemos observar en la siguiente 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 25 Salida de pantalla de tratamiento de parámetros Por referencia en PSeInt

1.4.2.2 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 modifique dentro de un subproceso cambiará también su valor en el algoritmo de origen.

1.4.3 Variables globales

Una variable global es aquella que 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.

Es importante recalcar 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 (2014) las excepciones para utilizar variables globales son las constantes globales, las cuáles 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 ser declaradas a lo interno de cada uno o bien recibidas como parámetros.

1.4.4 Variables locales

Una variable local es la que se **declara dentro del proceso principal o de un subproceso y su uso de restringe a esa parte del código**, es decir no puede ser usada 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.

1.5 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.

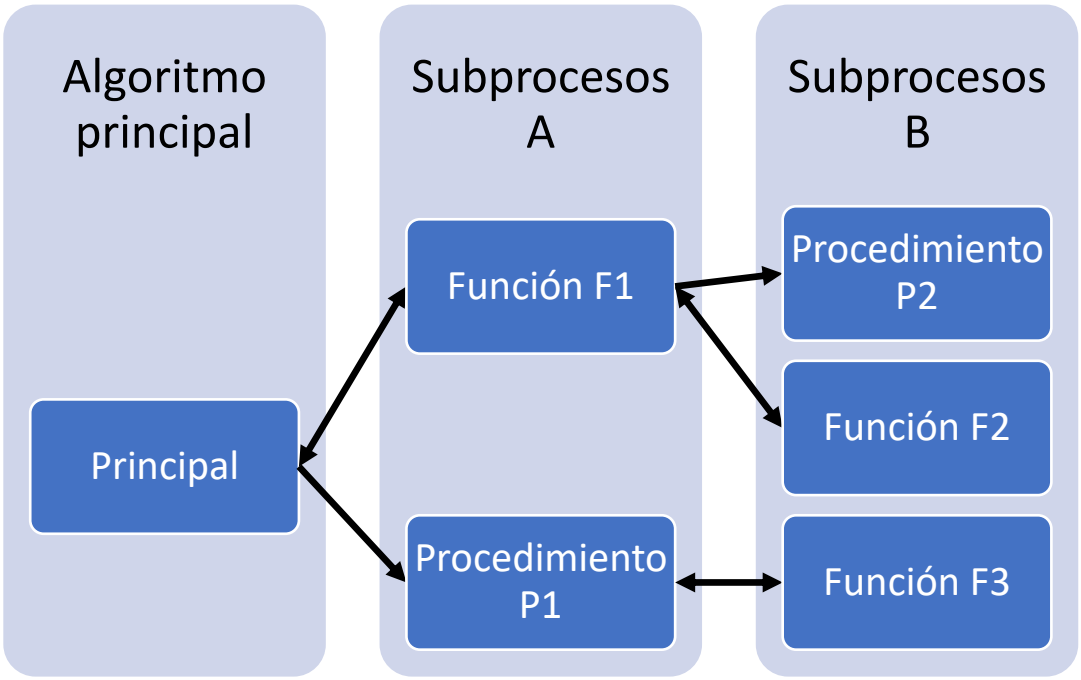


Figura 26 Esquema de retorno de datos y diferencia entre procedimientos y funciones

Como se puede observar en el gráfico, 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. Es importante aclarar que puede darse el caso de que los subprocesos, tanto procedimientos como funciones, no reciban datos, si es un procedimiento simplemente hará lo que debe hacer sin retornar valor alguno y la función hará su trabajo y siempre regresará algún valor.

1.5.1 Procedimientos en PSeInt

En PSeInt un procedimiento tiene la siguiente estructura:

```
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 27 Multiplicación de números usando un procedimiento en PSeInt

Notemos que el subproceso solo lleva el nombre y luego los parámetros que recibe entre paréntesis, en el ejemplo son parámetros **Por Valor**, recordemos que en PSeInt **si no se indica** se **asume** que son con ese tipo de paso.

En el algoritmo principal, se leen las variables **Numero1 y Numero2**, 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 lo que 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 28 Pseudocódigo en PSeInt usando procedimientos para leer datos y multiplicarlos

1.5.2 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:

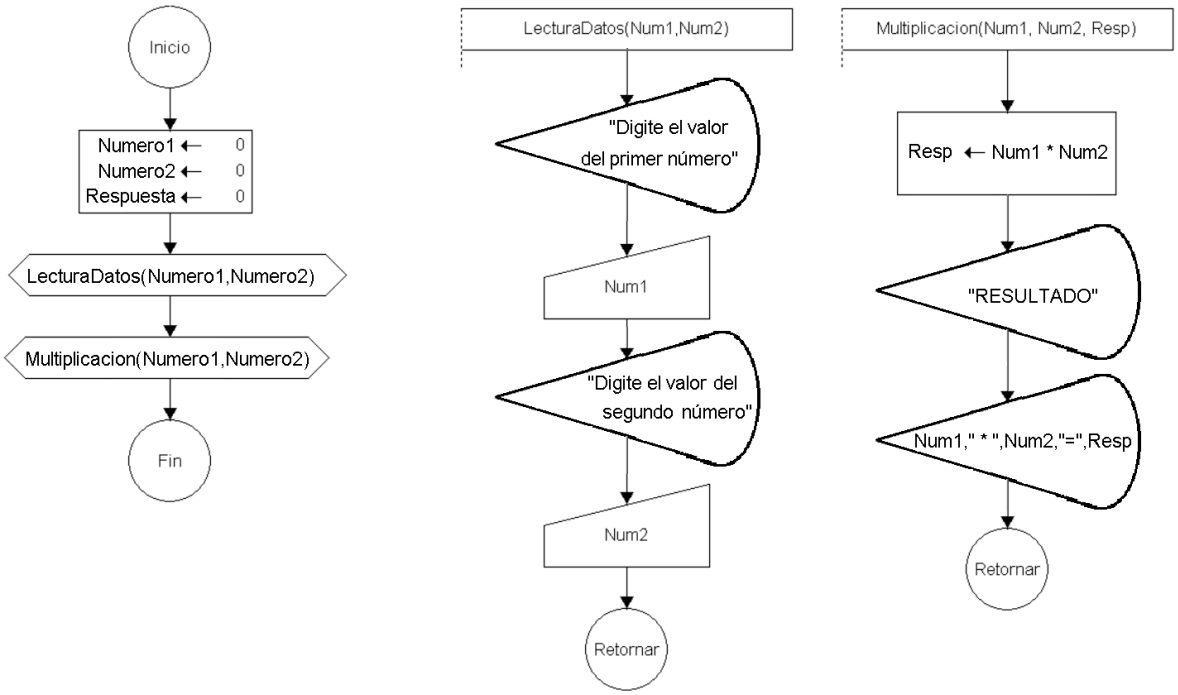


Figura 29 Diagrama en DFD usando procedimientos para leer datos y multiplicarlos

1.5.3 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 debe almacenarse o emplearse directamente en una instrucción.

Una función tiene la siguiente estructura:

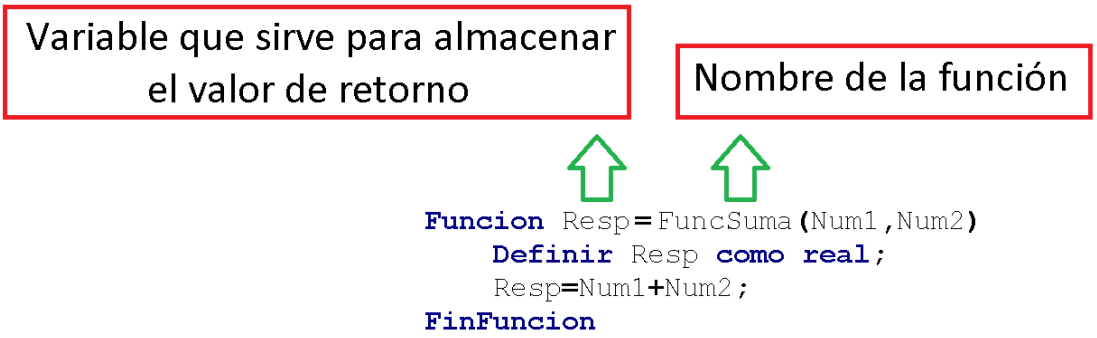


Figura 30 Detalle de la estructura de una función en PSeInt

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 31 Pseudocódigo en PSeInt que usa una función para sumar dos números

Podemos notar que en el proceso principal de 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 es utilizada 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 32 Salida de pantalla del pseudocódigo que suma de dos números con una función

1.5.4 Funciones en DFD

En DFD, los subprocesos no se diferencian entre procedimientos y funciones, esto obedece a que lo que se trabaja es la modificación de valores de variables, ya sea únicamente dentro de un subproceso o a nivel general de todo el algoritmo.

En un diagrama de flujo siempre de 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 en PSeInt:

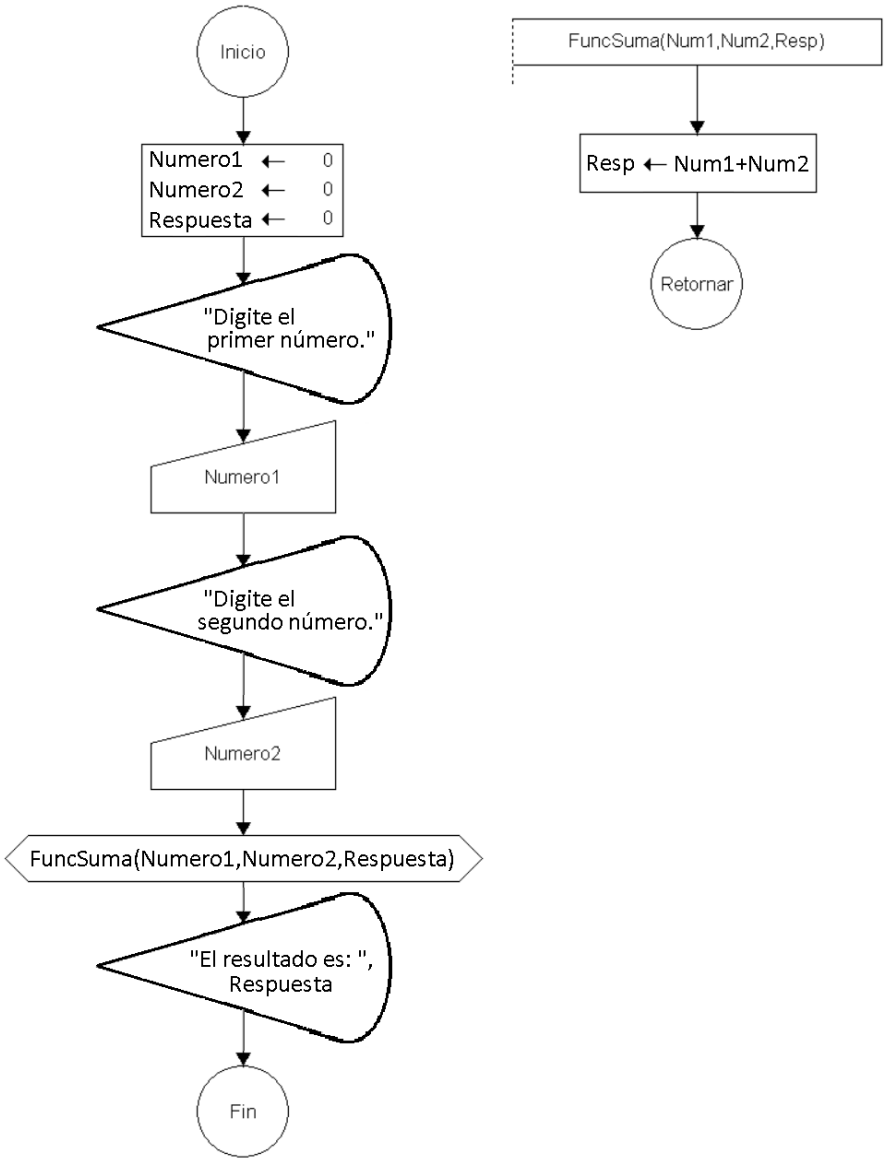


Figura 33 Diagrama en DFD que suma dos números usando el equivalente a una función

1.6 Ejemplos

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

Ejemplo 1 Algoritmo para dividir dos números.

Funcionamiento: Toma dos números enteros y los divide entre sí, pero antes valida que el divisor no sea cero, luego de calcular la división evalúa si el cociente (resultado de la división) es un número múltiplo de 3 y de 5.

Algoritmo original

```
Proceso DivisionMultiplos
    Definir dividendo, divisor, cociente Como Entero //declaración de variables
    //inicialización de los valores de las variables //
    dividendo =0;
    divisor=0;
    cociente =0;

    Escribir "Digite el dividendo"; //mensaje solicitando el dividendo
    Leer dividendo;                 //lee el valor desde teclado
    Escribir "Digite el divisor";   //mensaje solicitando el divisor
    Leer divisor;                   //lee el valor desde teclado
    Si divisor=0 entonces //Consulta si el divisor es cero
        Escribir "Error, el divisor NO puede ser cero. NO se puede realizar la división";
    FinSi
    Si Divisor!=0 entonces //realiza la operación porque el divisor es diferente a cero
        cociente= dividendo/divisor;
        Escribir "Cociente es igual a: ",cociente;
    FinSi
    //Si se dan las condiciones de loscálculos para determinar si el número es múltiplo de 3 y 5
    Si ((cociente Mod 3=0) Y (cociente Mod 5=0) Y (divisor!=0)) Entonces
        Escribir "El cociente ",cociente," es múltiplo de 3 y 5 a la vez."; //envía mensaje
    FinSi
FinProceso
```

Figura 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 evalúen en un procedimiento el cociente de la división.

```
Algoritmo Division
    Definir Dividendo, Divisor, Cociente Como Entero;
    Dividendo=0;
    Divisor=0;
    Cociente=0;

    LecturaDatos(Dividendo,Divisor);

    Si ValidaDivisor(Divisor)=Verdadero entonces
        Cociente= Dividendo/Divisor;
        Escribir "Cociente es igual a: ",Cociente;
    SiNo
        Escribir "Error, el divisor NO puede ser cero.";
    FinSi

    EvaluaCociente(Cociente,Divisor);

FinAlgoritmo

SubProceso LecturaDatos (Dvendo Por Referencia,Dvsor Por Referencia)
    Escribir "Digite el dividendo.";
    Leer Dvendo;
    Escribir "Digite el divisor.";
    Leer Dvsor;
FinSubProceso

SubProceso DivisorValido=ValidaDivisor(Dvsor)
    Definir DivisorValido Como Logico;
    Divisorvalido=Verdadero;
    Si Dvsor=0 entonces
        Divisorvalido=Falso;
    FinSi
FinSubProceso

SubProceso EvaluaCociente(Coci,Dvsor)
    Si ((Coci Mod 3=0) Y (Coci Mod 5=0) Y (Dvsor!=0)) Entonces
        Escribir "El cociente ",Coci," es múltiplo de 3 y 5 a la vez.";
    FinSi
FinSubProceso
```

Figura 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:

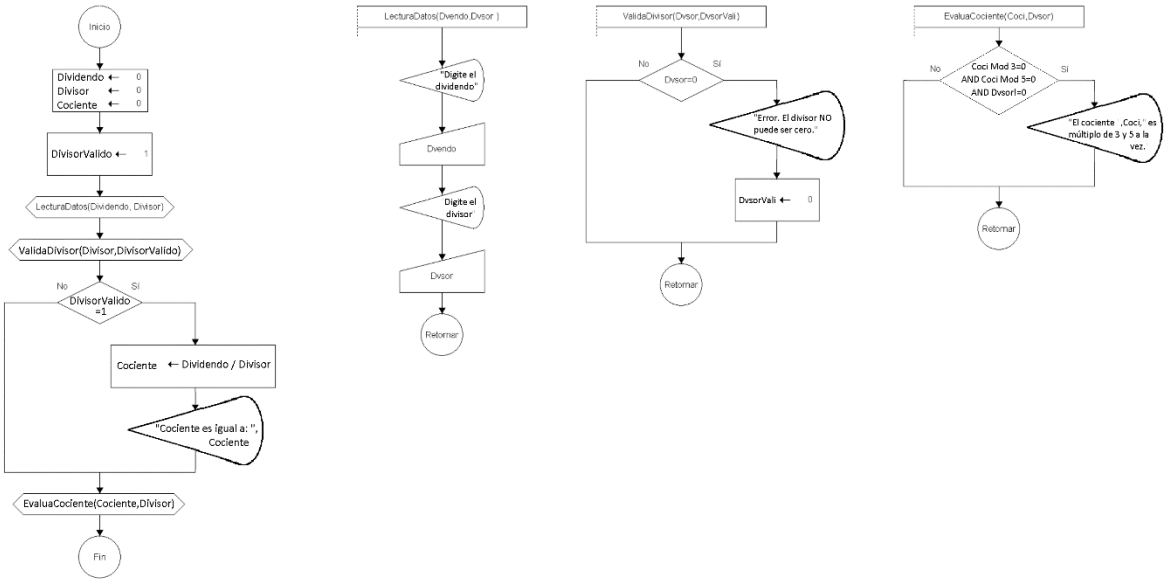


Figura 36 Diagrama con división de números y verificación de múltiplos de 3 y 5 con subprocesos

Nota: 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.

Ejemplo 2 Venta de entradas al cine.

Funcionamiento: Se reciben por teclado una opción, donde se establece si la persona es menor o mayor de edad. Si es menor de edad solo pueden vendérsele 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 ₡2500 y el de la VIP de ₡5500.

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 37 Pseudocódigo cálculo de costo 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
14     Escribir "El total a pagar es de: ",CalculaPago(tipoCliente,cantEntradas,salaReg,salaVIP,tipoSala);
15 FinAlgoritmo
16
17 SubProceso LeerEntradas(cantEnt Por Referencia)
18     Escribir "¿Cuántas entradas desea?";
19     Leer cantEnt;
20 FinSubProceso
21
22 SubProceso total=CalculaPago(tipoCli,cantEnt,salR,salV,tipSal)
23     Definir total Como Entero;
24     total=0;
25     Si tipoCli="m" O tipSal="R" Entonces
26         total=cantEnt*salR;
27     SiNo
28         total=cantEnt*salV;
29     FinSi
30 FinSubProceso
```

Figura 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 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 además que esta función forma parte de la instrucción Escribir de la línea 14.

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

Funcionamiento: 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=Peso/(Estatura Mts*Estatura Mts)

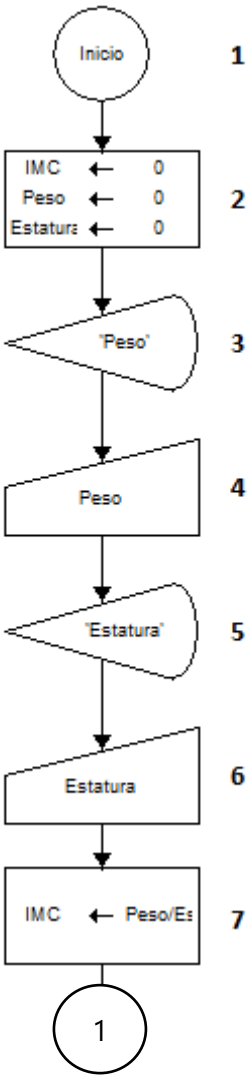
Nótese que la estatura en la fórmula está en metros por lo que se debe pasar el dato introducido por el usuario a esa unidad, es decir, se divide la estatura original entre 100.

Luego del cálculo se debe mostrar un mensaje del estado según la tabla presentada a continuación.

Tabla 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

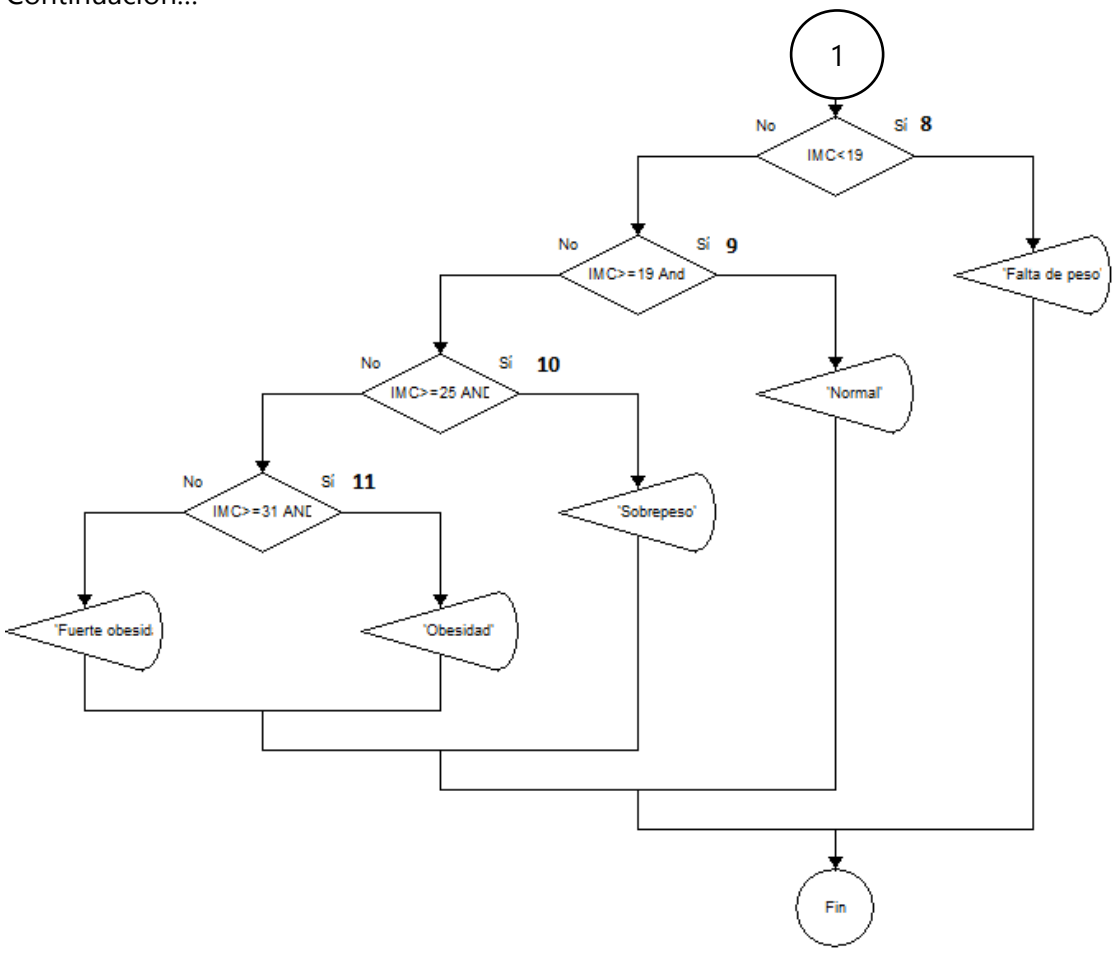
Algoritmo original



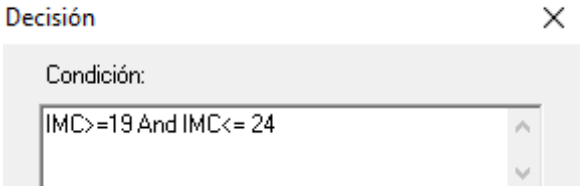
1. Inicio del algoritmo.
2. Inicialización de las variables que se requieren en el problema: la estatura, el peso y el IMC.
3. Mensaje al usuario para que indique el peso.
4. Lee desde teclado, el peso en kilogramos digitado por el usuario.
5. Mensaje al usuario para que digite la estatura en centímetros.
6. Lee desde teclado, la estatura en centímetros digitado por el usuario.
7. Cálculo del índice de masa corporal, según la fórmula: IMC <-Peso/(Estatura Mts*Estatura Mts)

Continúa...

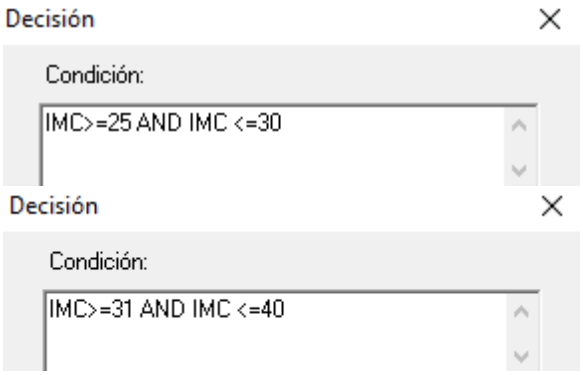
Continuación...



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



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



3. Si el IMC está entre 25 y 30, envía el mensaje de Sobrepeso.

4. 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.

Figura 39 Salida de pantalla diagrama de flujo para determinar el IMC

Algoritmo con subprocesos

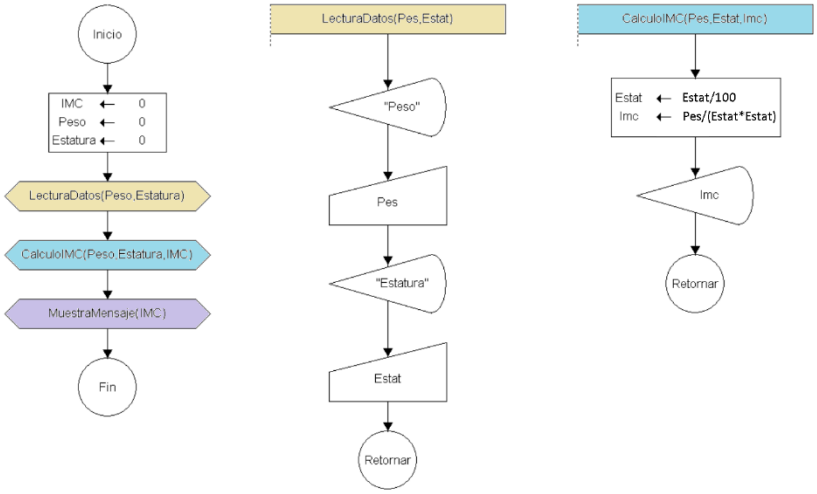


Figura 40 Primera parte diagrama de flujo con subprocesos para determinar el IMC

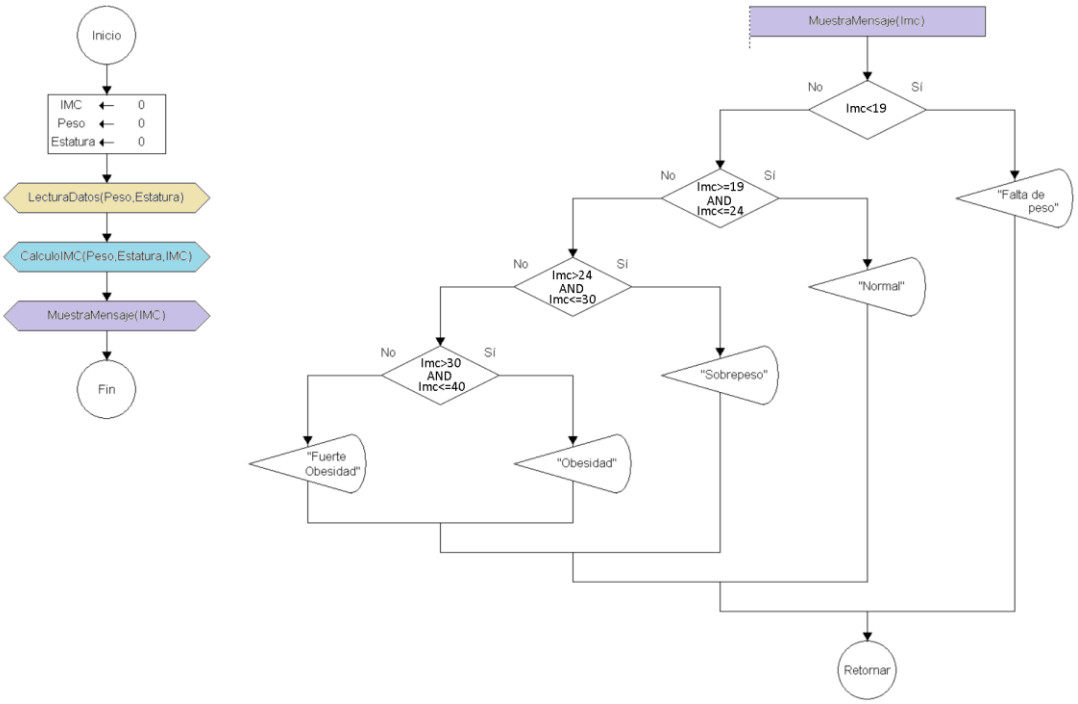


Figura 41 Segunda parte diagrama de flujo con subprocesos para determinar el IMC

En la modificación se utilizaron tres subprocesos (para mejor identificación se pintaron en un editor de imágenes), en el primer subproceso, **LecturaDatos**, 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** donde 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 muestra un determinado mensaje al usuario.

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*4*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 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 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 para el algoritmo principal.

Luego de la lectura, se 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 para 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 "El número ",base," elevado a ",exponente," es ",CalculaPotencia(base,exponente);
9
10 FinAlgoritmo
11
12 SubProceso IngresoValores(bas Por Referencia,expo Por Referencia)
13     Escribir "Escriba la base y el exponente.";
14     Leer bas,expo;
15 FinSubProceso
16
17 SubProceso tot=CalculaPotencia(bas Por Valor,expo Por Valor)
18     Definir tot,contador Como Entero;
19     tot=1;
20     contador=1;
21     Mientras contador<=expo Hacer
22         tot=tot*bas;
23         contador=contador+1;
24     FinMientras
25 FinSubProceso
```

Figura 44 Pseudocódigo con subprocesos modificado para calcular potencias

Ejemplo 5 Calculadora geométrica.

Funcionamiento: Se solicita al usuario que elija una opción de un menú y de acuerdo a 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, este es el diagrama principal:

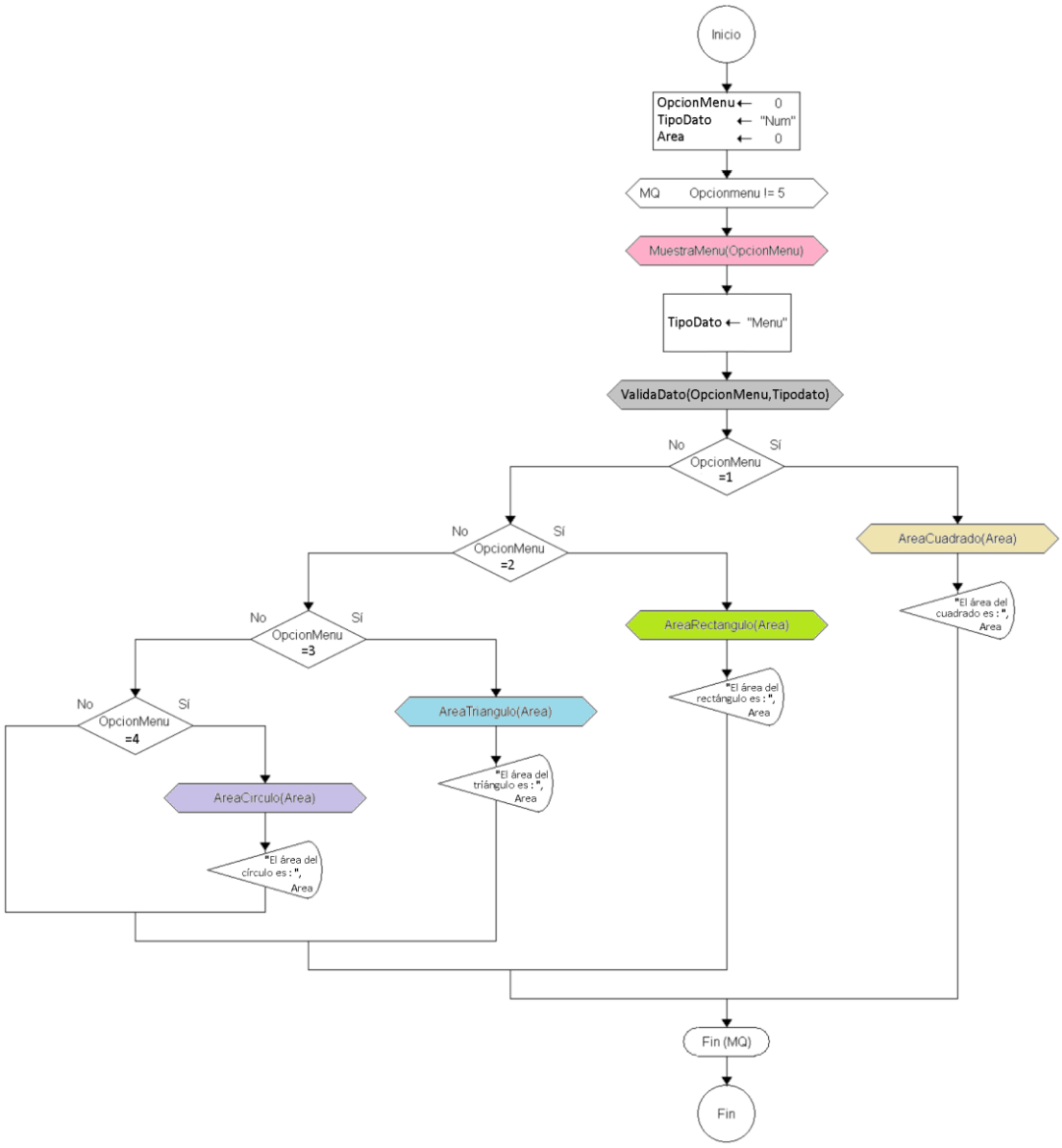


Figura 45 Diagrama principal de la calculadora geométrica

Vemos como en el diagrama principal se declaran las variables **OpcionMenu**, **TipoDato** y **Area**; la primera es para almacenar la opción del menú que digite el usuario, la segunda es para ayudar a validar la opción del menú en un subproceso llamado **ValidaDato** y la tercera variable es para almacenar el valor del área calculada, debido a que esta es una variable afín a cualquiera de las figuras geométricas.

Este diagrama está compuesto por un ciclo **Mientras**, el cual se repite mientras la opción del menú digitada se **diferente a 5**, dentro de este ciclo se muestra el menú y se lee la opción digitada por el usuario, por medio del subproceso **MuestraMenu**; luego se evalúa el valor de la variable **OpcionMenu** en el subproceso **ValidaDato**.

Posterior a la validación, se presentan **4 Si anidados**, en los que se llama a los respectivos subprocesos para el cálculo de áreas; este es el detalle de los subprocesos MuestraMenu y ValidaDato:

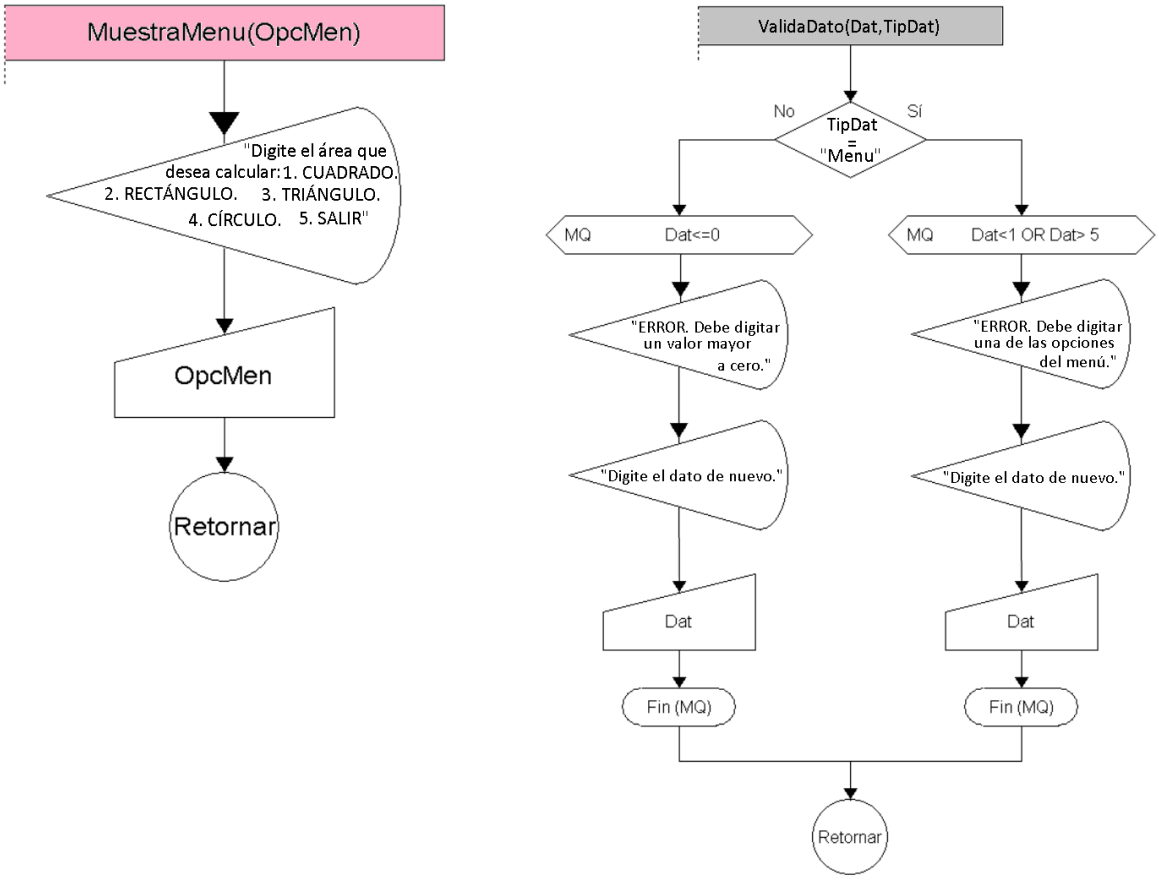


Figura 46 Diagrama se los subprocesos MuestraMenu y ValidaDatode la calculadora geométrica

El subproceso **ValidaDato** también se emplea para validar que los valores para el cálculo de áreas sea mayo a cero, por lo que 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 antes, en los Si anidados, se hace el llamado a los subproceso para el cálculos de la 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.

A continuación, se muestran los diagramas de los subprocesos:

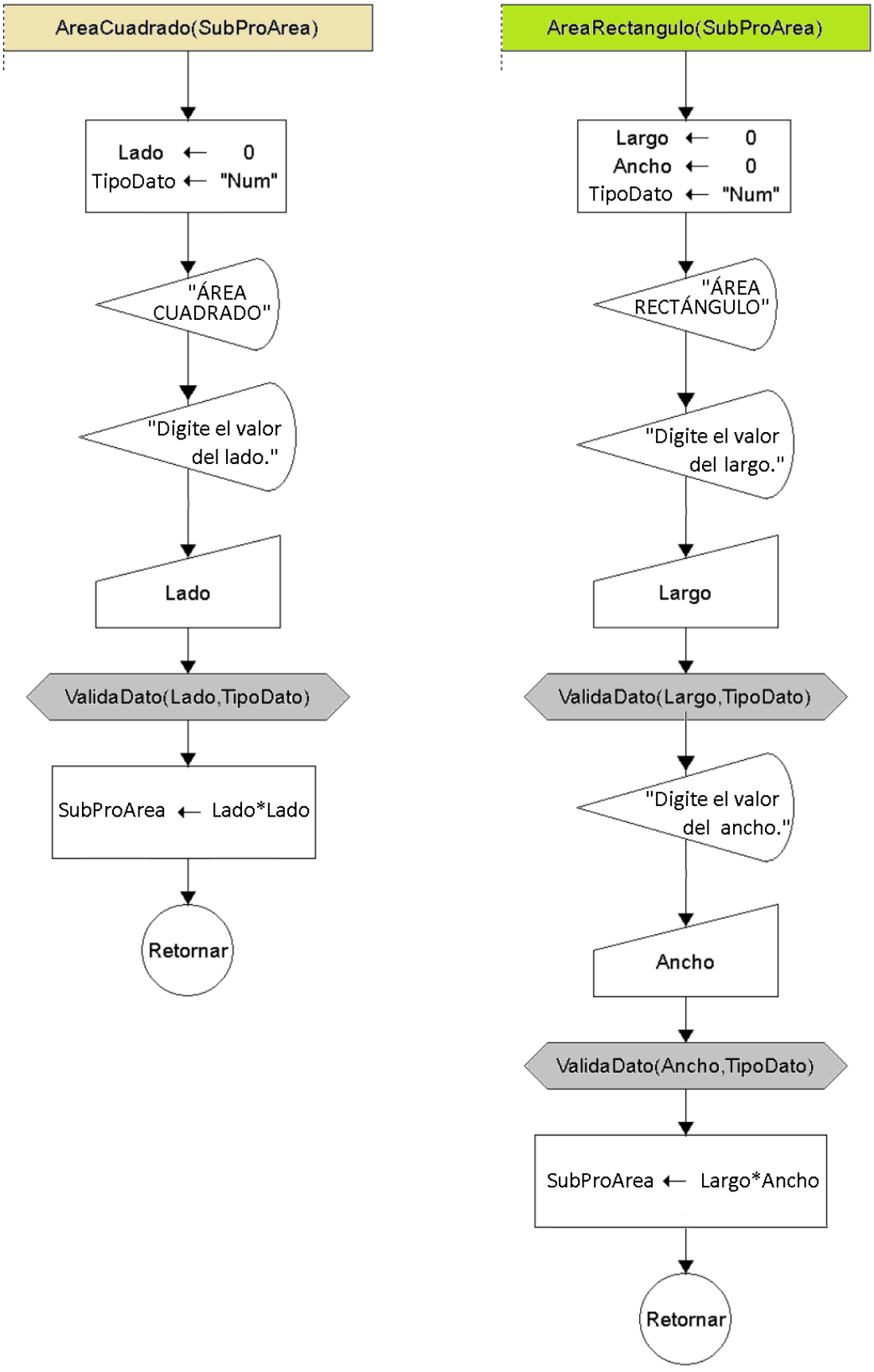


Figura 47 Diagramas de los subprocesos AreaCuadrado y AreaRectangulo de la calculadora geométrica

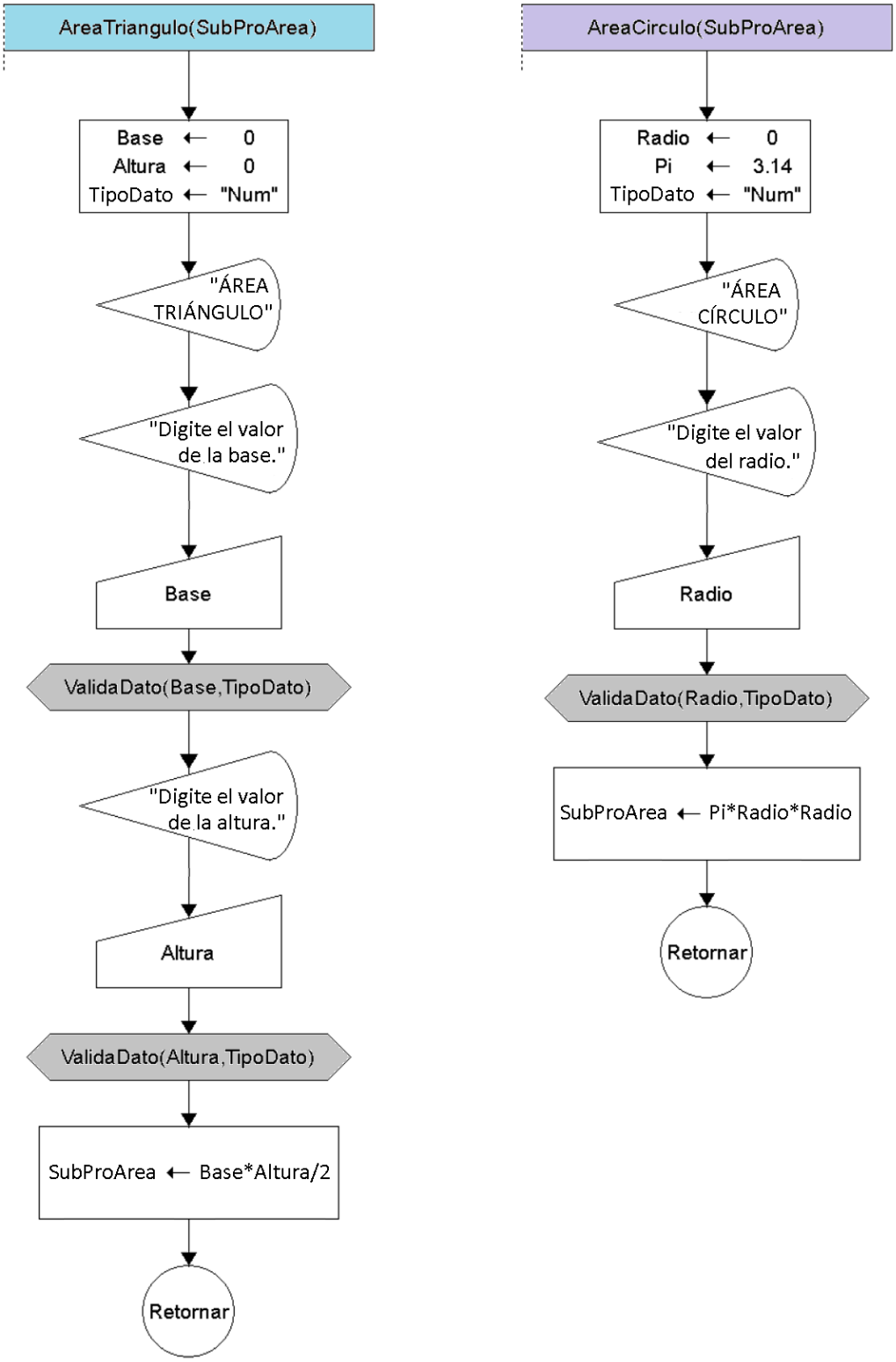


Figura 48 Diagrama se los subprocesos AreaTriangulo y AreaCirculo de la calculadora geométrica

En cada subproceso se declaran las variables necesarias para el cálculo del área y se calcula el valor del área, 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 a un subproceso (ValidaDato) desde los subprocesos de cálculo de áreas, esto nos indica que no solamente desde el principal se pueden invocar subprocesos.

Ejemplo 6 Modificación de matriz.

Funcionamiento: El pseudocódigo presenta un menú donde se pueden hacer diversas acciones sobre una matriz de 3x4, 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 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:

```
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 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 de 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.

Este es 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 51 Pseudocódigo del subproceso LlenaMatriz algoritmo para modificar una matriz

En este procedimiento llena la matriz, la cual se recibe Por Referencia; aunque debemos resaltar 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, ya que esto nos indica que la matriz ya tiene elementos. Finalmente se muestra la matriz, para esto se invoca al subproceso **MuestraMatriz**.

MuestraMatriz es la segunda opción en el menú y su detalle es el siguiente:

```
 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 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 primero se procede a mostrar la matriz utilizando dos ciclos **Para anidados**, pero si el valor de **MatrizLlena es falso**, se muestra un mensaje de **error y la matriz no se muestra** ya que se asume que está vacía, esto se da porque el usuario eligió la opción 2 antes que la 1, lo cual es libre de hacerlo, pero el pseudocódigo debe contemplar ese detalle e indicar cómo debe proceder.

La tercera opción del menú se relaciona con el subproceso **ModificaMatriz**:

```
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 53 Pseudocódigo del subproceso ModificaMatriz algoritmo para modificar una matriz

En **ModificaMatriz** se hace la **misma validación** con respecto a si la **matriz está llena o no**, luego de verificar que el arreglo contiene elementos se ejecuta la modificación, esta consiste es sumar una unidad al valor que tiene 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**, ya que es

en ese subproceso donde 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 es el siguiente:

```
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 54 Pseudocódigo del subproceso Salida algoritmo para modificar una matriz

En el algoritmo se emplea otro subprograma llamado **Mensaje**, este lo que hace es mostrar en pantalla un mensaje y esperar a que el usuario presione alguna tecla, su detalle es el siguiente:

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

Figura 55 Pseudocódigo del subproceso Mensaje algoritmo para modificar una matriz

1.7 Ejercicios de autoevaluación

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 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 de 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.
4. Modifique el ejercicio anterior, pero en este caso implemente la función SolicitudDatos en una instrucción de asignación.
5. Elabore un diagrama de flujo en DFD que solicite el nombre de una persona y envíe un saludo: Hola, *nombre*. El nombre debe aceptarlo en un subproceso y la muestra del mensaje en otro.
6. Elabore 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 colones.
    - Impuesto de renta 13%.

    Además, considere que el salario por hora del trabajador es de 2 000 colones y que las horas extra se pagan doble, la jornada es de 40 semanales. Elabore los siguientes subprocesos:
    - LecturaDatos: lee los datos del trabajador y el de sus horas laboradas.
    - CalculoSalarioBruto: debe calcular el salario bruto.
    - CalculoDeducciones: debe calcular las deducciones por cargas sociales.
    - CalculoSalarioNeto: debe calcular el salario neto, este es el salario bruto menos las deducciones de cargas sociales.
    - MuestraInforme: debe mostrar 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 consultar al usuario si desea procesar otro empleado, si la respuesta es afirmativa debe solicitar todos los datos nuevamente, de lo contrario sale del algoritmo.

7. Elabore un pseudocódigo que llene un vector de 100 posiciones con números aleatorios del 100 al 500, una vez lleno el arreglo, debe solicitar al usuario un número a buscar en el vector, valide que el número digitado por el usuario esté entre 100 y 500.

   Luego de la búsqueda el algoritmo debe informar al si el número buscado está o no en el vector y, de estarlo, debe indicar cuántas veces se encuentra, luego de esto debe consultar al usuario si desea hacer otra búsqueda, de ser así solicite de nuevo el número a buscar, de lo contrario muestre el vector y luego salga del algoritmo. Debe implementar subprocesos para todas las acciones del vector.

8. Elabore un pseudocódigo que calcule el promedio de una matriz, así como el número mayor y menor que hay en la misma. Trabaje con una matriz de 10x10, pero el usuario podrá elegir las dimensiones de la matriz (solo elige una porque el arreglo es cuadrado), debe validar 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.

9. Elabore un pseudocódigo que intercambie valores de dos celdas de un arreglo, el vector es de 10 posiciones y está lleno de nombres, se debe solicitar al usuario el número de dos celdas e intercambiar el contenido de esas celdas entre sí. Considere que debe hacer un subproceso para que el usuario llene el vector, otro para solicitar los números de las celdas, otro para hacer el intercambio y otro para mostrar el

arreglo. Primero debe solicitar los nombres, luego imprimir el vector, después solicitar las dos posiciones que desea cambiar y por último mostrar de nuevo el vector.

1.8 Respuestas a los ejercicios de autoevaluación

1. Paso Por Valor.
2. Paso Por Referencia.
3. Pseudocódigo del tipo de cambio:

```
Algoritmo TipoCambio
    Definir Colon,Cambio Como Real;
    Colon=0;
    Cambio=0;
    SolicitudDatos(Colon,Cambio);
    Escribir "Con ",Colon," colones, puede comparar ",CalculoDolares(Colon,Cambio)," dólares.";
FinAlgoritmo

SubProceso SolicitudDatos(Col Por Referencia,Camb Por Referencia)
    Escribir "Digite la cantidad de colones que desea cambiar.";
    Leer Col;
    Escribir "Digite el tipo de cambio.";
    Leer Camb;
FinSubProceso

SubProceso Dolares=CalculoDolares(Col Por Valor,Camb Por Valor)
    Definir Dolares Como Real;
    Dolares=0;
    Dolares=Col/Camb;
FinSubProceso
```

Figura 56 Pseudocódigo algoritmo con subprocesos del tipo de cambio

4. Pseudocódigo del tipo de cambio con modificación en la implementación de la función:

```
Algoritmo TipoCambio
    Definir Colon,Cambio,Dolar Como Real;
    Colon=0;
    Cambio=0;
    Dolar=0;
    SolicitudDatos(Colon,Cambio);
    Dolar=CalculoDolares(Colon,Cambio);
    Escribir "Con ",Colon," colones, puede comparar ",Dolar," dólares.";
FinAlgoritmo

SubProceso SolicitudDatos(Col Por Referencia,Camb Por Referencia)
    Escribir "Digite la cantidad de colones que desea cambiar.";
    Leer Col;
    Escribir "Digite el tipo de cambio.";
    Leer Camb;
FinSubProceso

SubProceso Dolares=CalculoDolares(Col Por Valor,Camb Por Valor)
    Definir Dolares Como Real;
    Dolares=0;
    Dolares=Col/Camb;
FinSubProceso
```

Figura 57 Pseudocódigo modificado algoritmo con subprocesos del tipo de cambio

5. Diagrama de flujo del algoritmo de saludo:

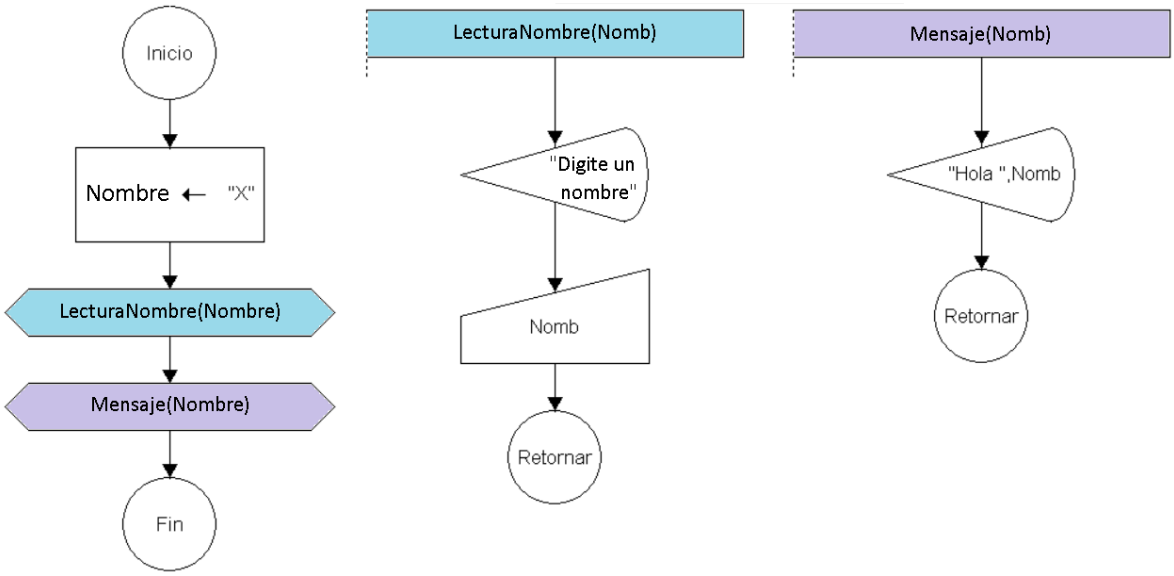


Figura 58 Diagrama de flujo del algoritmo saludo

6. Diagrama de flujo del algoritmo de cálculo de salario:

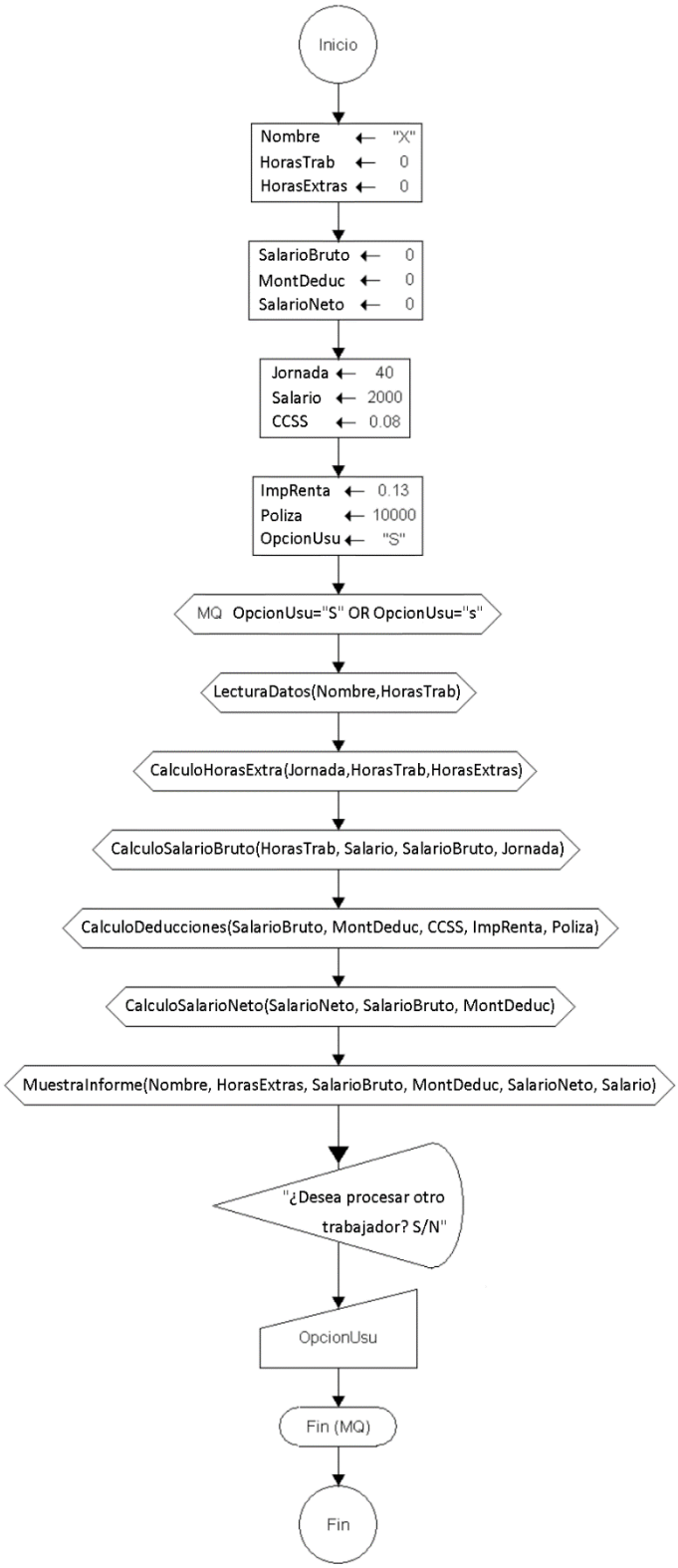


Figura 59 Diagrama de flujo principal algoritmo cálculo de salario

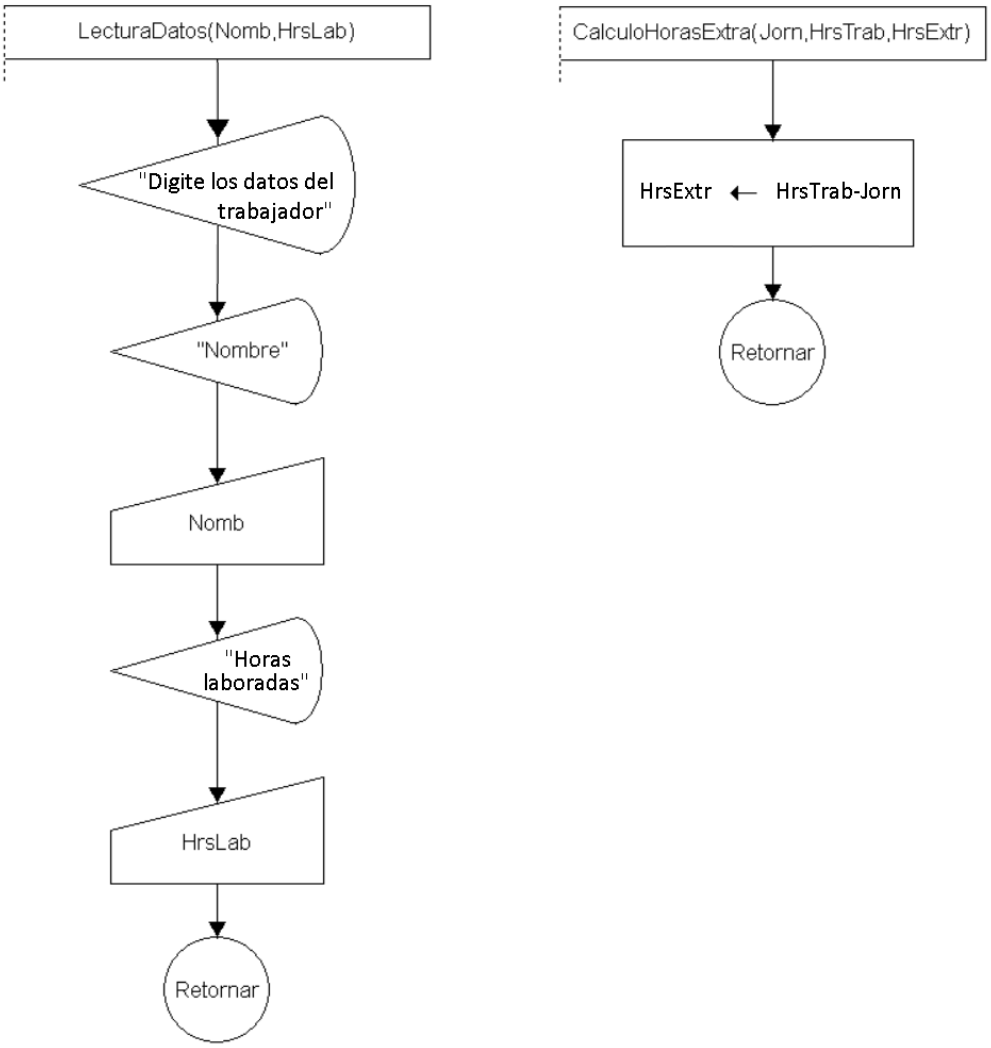


Figura 60 Diagrama de flujo subprocesos LecturaDatos y CalculoHorasExtra algoritmo cálculo de salario

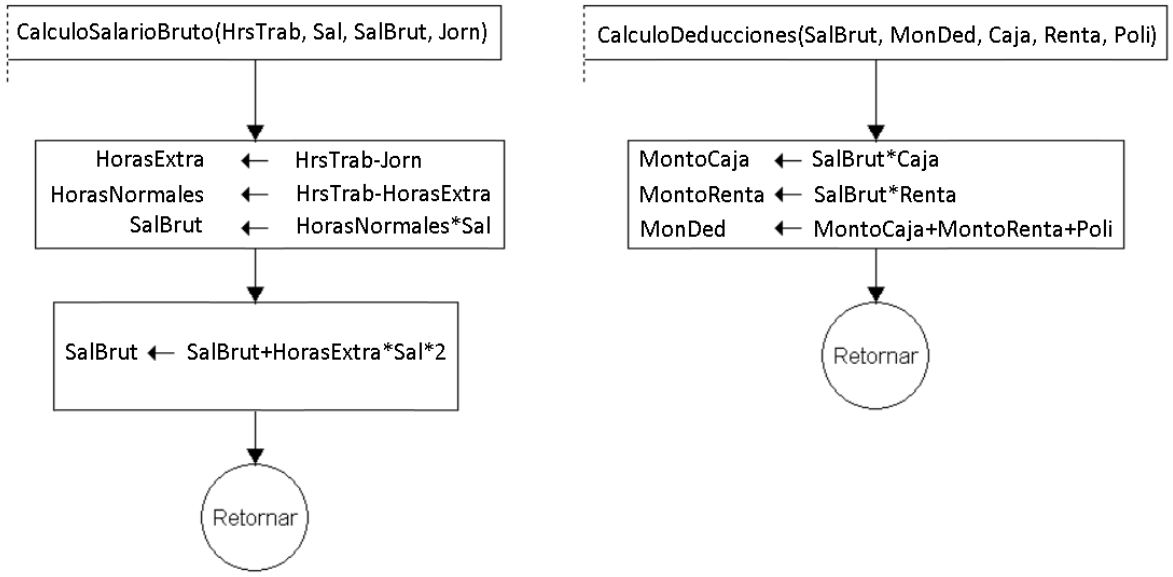


Figura 61 Diagrama de flujo subprocesos CalculoSalariBruto y CalculoDeducciones algoritmo cálculo de salario

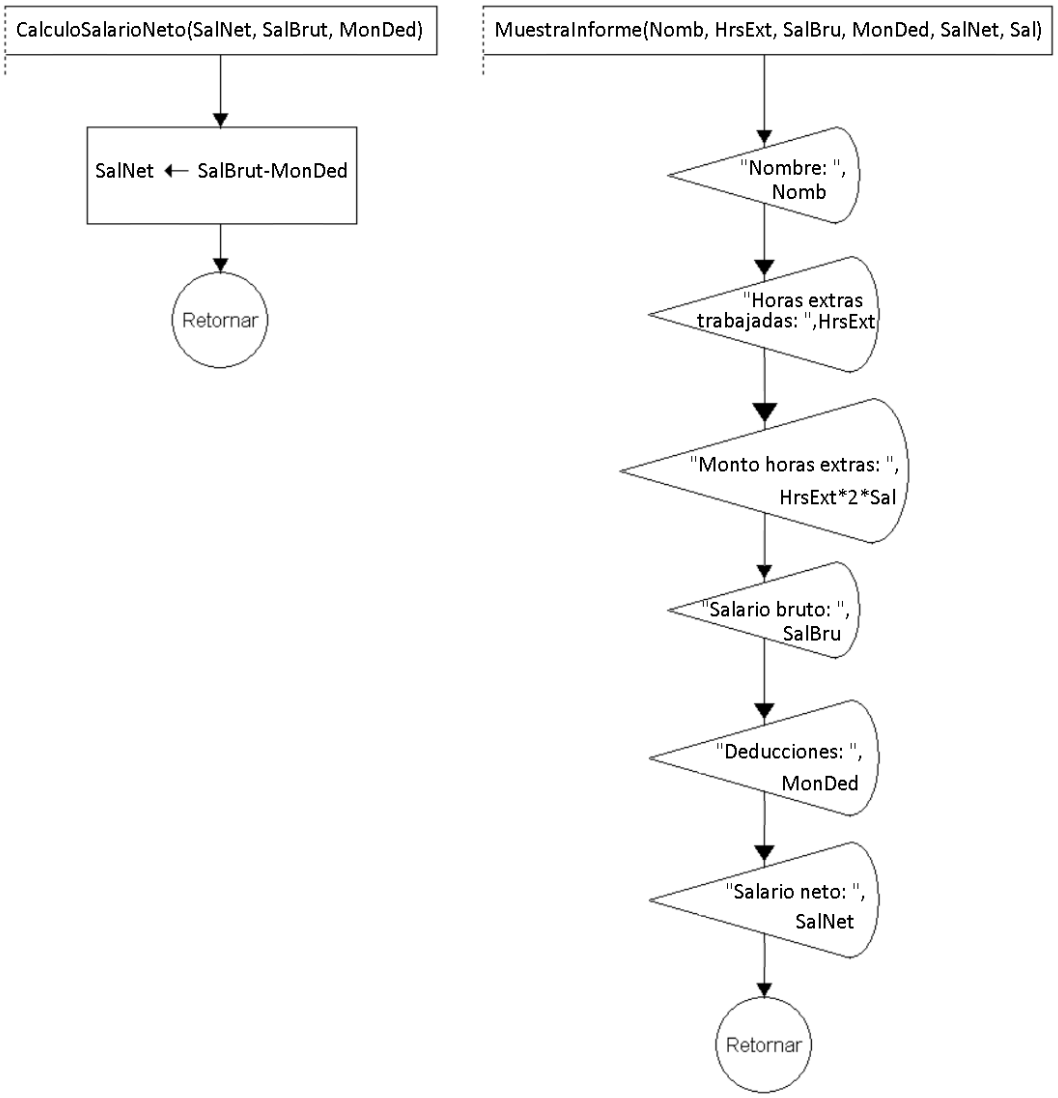


Figura 62 Diagrama de flujo subprocesos CalculoSalarioNeto y MuestraInforme algoritmo cálculo de salario

7. Pseudocódigo búsqueda en vector:

```
Proceso BusquedaenVectorConSubProc

    //Declaraciones de variables y vector
    Definir Vector, NumeroBuscado, CantidadCoincidencias Como Entero;
    Definir Opcionusuario Como Caracter;

    //Dimensionamiento del arreglo
    Dimension Vector(100);

    //Inicializaciones de variables
    NumeroBuscado=0;
    CantidadCoincidencias=0;
    Opcionusuario="S";

    LlenaVector(Vector);
    Mientras Opcionusuario="S" o Opcionusuario="s" Hacer

        //Solicitud, lectura y validación del número a buscar
        SolicitaNumero(NumeroBuscado);

        //Búsqueda del número en el vector
        CantidadCoincidencias=Busqueda(Vector,NumeroBuscado);


        //Muestra de resultados
        MuestraResultados(CantidadCoincidencias,NumeroBuscado);

        Escribir "¿Desea hacer otra búsqueda? S/N";
        Leer Opcionusuario;
    FinMientras
    MuestraVector(Vector);
    Escribir "Presione cualquier tecla para salir.";
    Esperar Tecla;

FinProceso
```

Figura 63 Pseudocódigo principal algoritmo búsqueda en vector

```
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 64 Subprocesos LlenaVector, MuestraVector y SolicitaNumero algoritmo búsqueda en vector

```
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 65 Subprocesos Busqueda, MuestraResultados algoritmo búsqueda en vector

8. Pseudocódigo promedio y número mayor y menor de una matriz:

```
Proceso PromedioMayorMenorMatriz

    //Declaraciones//
    Definir Matriz Como Entero;
    Definir Sumatoria,NumMayor,NumMenor,TamanioMatriz Como Entero;
    Definir Promedio Como Real;

    //Dimensionamiento de la matriz//
    Dimension Matriz(10,10);

    //Inicialización de variables//
    Promedio=0;
    Sumatoria=0;
    TamanioMatriz=0;

    //Dimensionamiento de la matriz por el usuario//
    DimensionaMatriz(TamanioMatriz);

    //Llena la matriz//
    LlenaMatriz(Matriz,TamanioMatriz);

    //Calcula la sumatoria de la matriz//
    Sumatoria=SumatoriaMatriz(Matriz,TamanioMatriz);

    //Calcula el promedio//
    Promedio=Sumatoria/(TamanioMatriz*TamanioMatriz);

    //Determina el número mayor y menor de la matriz//
    DeterminaMayMen(Matriz,NumMayor,NumMenor,TamanioMatriz);

    //Muestra la matriz y el informe//
    MuestraMatriz(Matriz,TamanioMatriz);
    MuestraInforme(Sumatoria,Promedio,NumMayor,NumMenor);

FinProceso

SubProceso DimensionaMatriz(TamMat Por Referencia)
    Escribir "Bienvenid@";
    Repetir
        Escribir "Digite el tamaño de la matriz.";
        Leer TamMat;
        Si TamMat<1 O TamMat>10 Entonces
            Limpiar Pantalla;
            Escribir "Error. El tamaño de la matriz debe estar entre 1 y 10.";
        FinSi
    Hasta Que TamMat>=1 Y TamMat<=10
FinSubProceso
```

Figura 66 Pseudocódigo principal algoritmo de trabajos con 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 67 Procedimiento LlenaMatriz y Función SumatoriaMatriz algoritmo de trabajos con una matriz

```
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 68 Procedimiento DeterminaMayMen algoritmo de trabajos con una matriz

```
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 69 Subprocesos MuestraMatriz y MuestraInforme algoritmo de trabajos con una matriz

9. Pseudocódigo cambio de contenido en 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 70 Pseudocódigo principal algoritmo de cambio en vector

```
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 71 Procedimientos LlenaVector y MuestraVector algoritmo de cambio en vector

```
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 72 Procedimientos SolicitudCeldas y CambiaContenido algoritmo de cambio en vector

Referencias

Hernández, L. (2014). Apuntes de clase Fundamentos de la programación. Universidad Complutense de Madrid. Recuperado de: https://www.fdi.ucm.es/profesor/luis/fp/fp.pdf