Guía práctica de estudio 05: Diagramas de flujo

Elaborado por:
M.C. Edgar E. García Cano
Ing. Jorge A. Solano Gálvez

Revisado por:
Ing. Laura Sandoval Montaño

Guía práctica de estudio 05: Diagramas de flujo

Objetivo:

Elaborar diagramas de flujo que representen soluciones algorítmicas vistas como una serie de acciones que comprendan un proceso.

Actividades:

- Analizar un problema.
- Crear un algoritmo para resolver el problema.
- Representar el algoritmo en diagrama de flujo.

Introducción

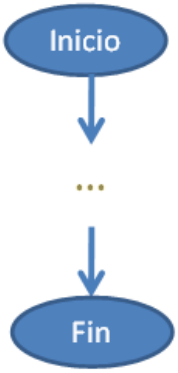
Un diagrama de flujo es la representación gráfica de un proceso, es decir, muestra gráficamente el flujo de acciones a seguir para cumplir con una tarea específica.

Dentro de las ciencias de la computación, un diagrama de flujo es la representación gráfica de un algoritmo. La correcta construcción de estos diagramas es fundamental para la etapa de codificación, ya que, a partir del diagrama de flujo es posible codificar un programa en algún lenguaje de programación.

Formas de los diagramas de flujo

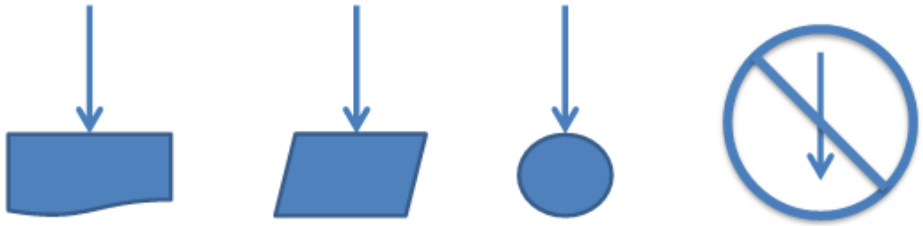
Los diagramas de flujo poseen símbolos que permiten estructurar la solución de un problema de manera gráfica. A continuación se muestran los elementos que conforman este lenguaje gráfico.

1. Todo diagrama de flujo debe tener un inicio y un fin.



2. Las líneas utilizadas para indicar la dirección del flujo del diagrama deben ser rectas, verticales u horizontales, exclusivamente.

3. Todas las líneas utilizadas para indicar la dirección del flujo del diagrama deben estar conectadas a un símbolo.



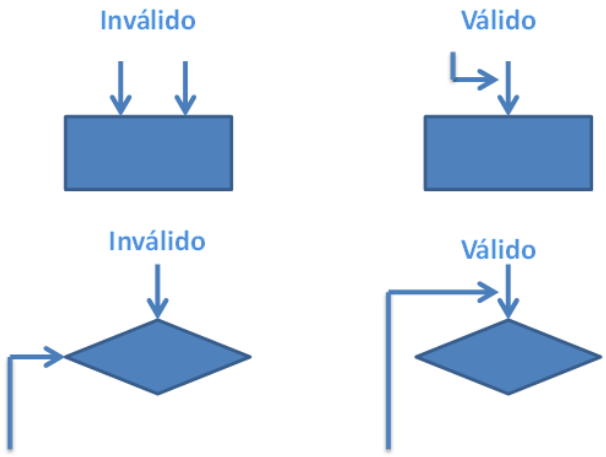
4. El diagrama debe ser construido de arriba hacia abajo (top-down) y de izquierda a derecha (left to right).

5. La notación utilizada en el diagrama de flujo debe ser independiente del lenguaje de programación en el que se va a codificar la solución.

6. Se recomienda poner comentarios que expresen o ayuden a entender un bloque de símbolos.

7. Si la extensión de un diagrama de flujo ocupa más de una página, es necesario utilizar y numerar los símbolos adecuados.

8. A cada símbolo solo le puede llegar una línea de dirección de flujo.



9. Notación de camello. Para nombrar variables y nombres de funciones se debe hacer uso de la notación de camello.

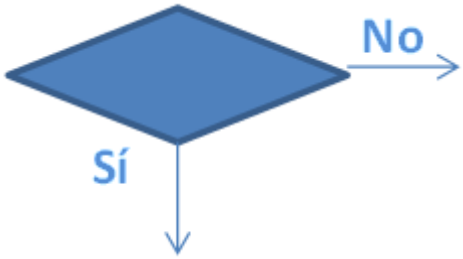
Los diagramas de flujo poseen símbolos que permiten estructurar la solución de un problema de manera gráfica. Por tanto es fundamental conocer los elementos que conforman este lenguaje gráfico.

Representa el inicio o el fin del diagrama de flujo.

Datos de entrada. Expresa lectura de datos.

Proceso. En su interior se expresan asignaciones u operaciones.

Decisión. Valida una condición y toma uno u otro camino.



Escritura. Impresión del o los resultado(s).

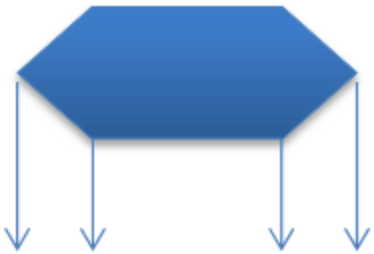
Dirección de flujo del diagrama.

Conexión dentro de la misma página.

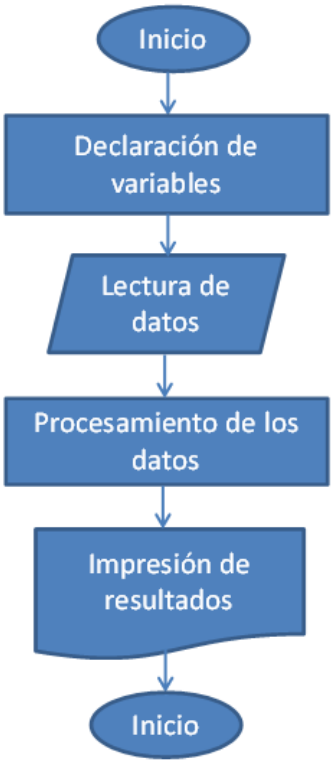
Conexión entre diferentes páginas.

Módulo de un problema. Llamada a otros módulos o funciones.

Decisión múltiple. Almacena un selector que determina la rama por la que sigue el flujo.



El diagrama de flujo para construir un diagrama de flujo es el siguiente:



Estructuras de control de flujo

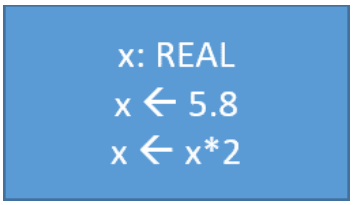
Las estructuras de control de flujo permiten la ejecución condicional y la repetición de un conjunto de instrucciones.

Existen 3 estructuras de control: secuencial, condicional y repetitivas o iterativas.

Estructura de control secuencial

Las estructuras de control secuenciales son las sentencias o declaraciones que se realizan una a continuación de otra en el orden en el que están escritas.

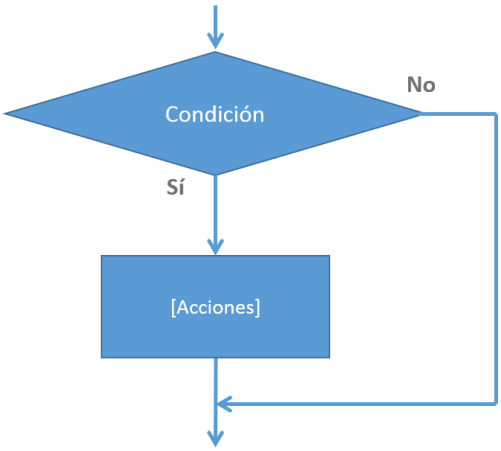
Ejemplo



Estructuras de control condicionales (o selectivas)

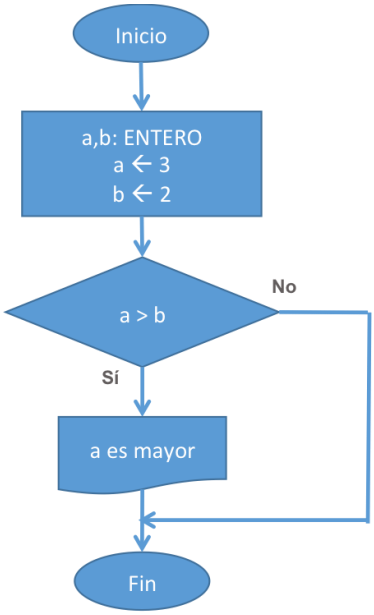
Las estructuras de control condicionales permiten evaluar una expresión lógica (condición que puede ser verdadera o falsa) y, dependiendo del resultado, se realiza uno u otro flujo de instrucciones. Estas estructuras son mutuamente excluyentes (o se ejecuta una acción o se ejecuta la otra).

La estructura de control de flujo más simple es la estructura condicional SI (IF), su sintaxis es la siguiente:



Se evalúa la expresión lógica y si se cumple (si la condición es verdadera) se ejecutan las instrucciones del bloque [Acciones]. Si no se cumple la condición, se continúa con el flujo normal del programa.

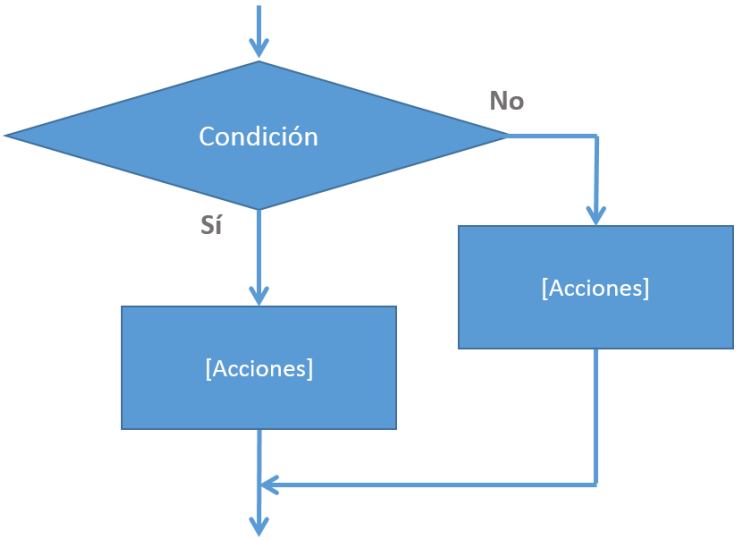
Ejemplo




```
// >>> a es mayor
```

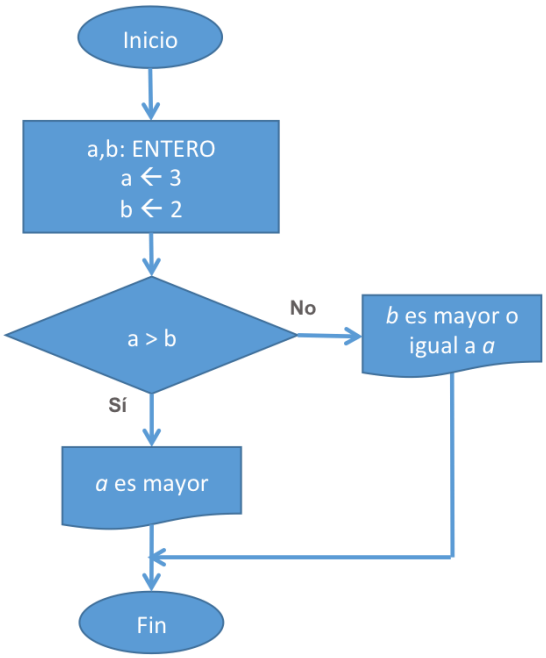
NOTA: La línea **//>>> valor**, indica el resultado que genera el ejemplo.

La estructura condicional completa es SI-DE LO CONTRARIO (IF-ELSE):



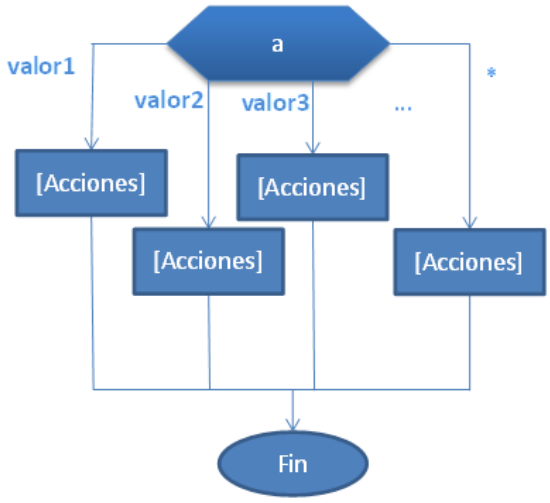
Se valúa la expresión lógica y si se cumple (si la condición es verdadera) se ejecutan las instrucciones del bloque Sí. Si no se cumple la condición se ejecutan las instrucciones del bloque No. Al final el programa sigue su flujo normal.

Ejemplo

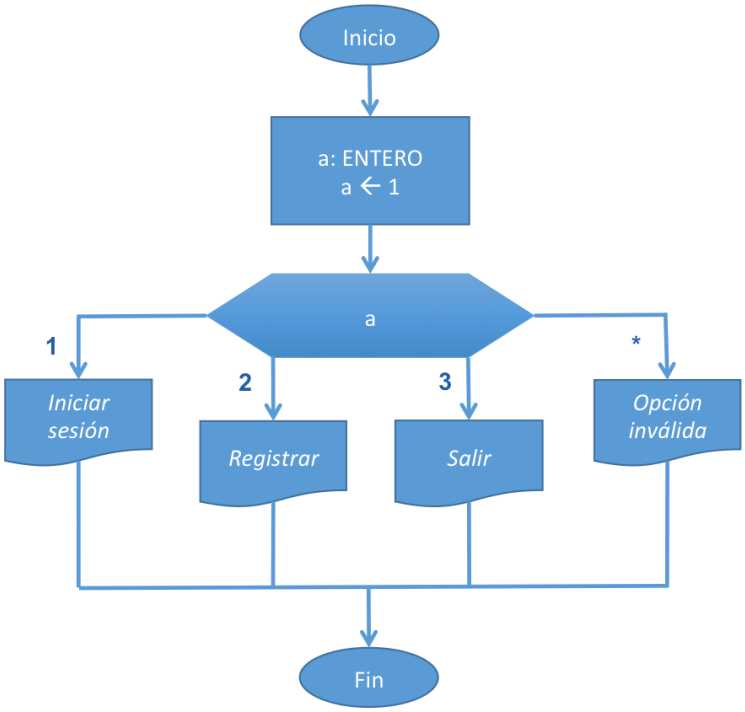



```
// >>> b es mayor
```

La estructura condicional SELECCIONAR-CASO valida el valor de la variable que está en el hexágono y comprueba si es igual al valor que está definido en cada caso (Líneas que emanan del hexágono). Si la variable no tiene el valor de algún caso se va a la instrucción por defecto (*).



Ejemplo

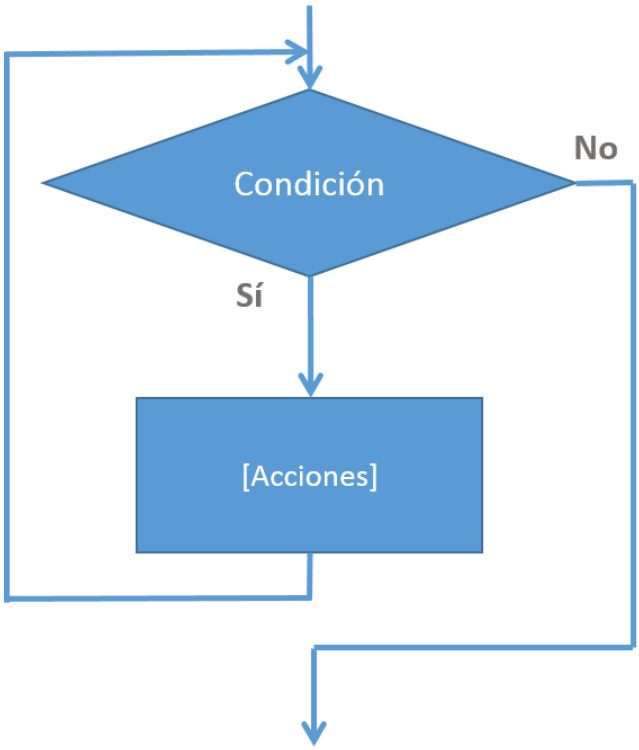



```
// >>> "Iniciar sesión"
```

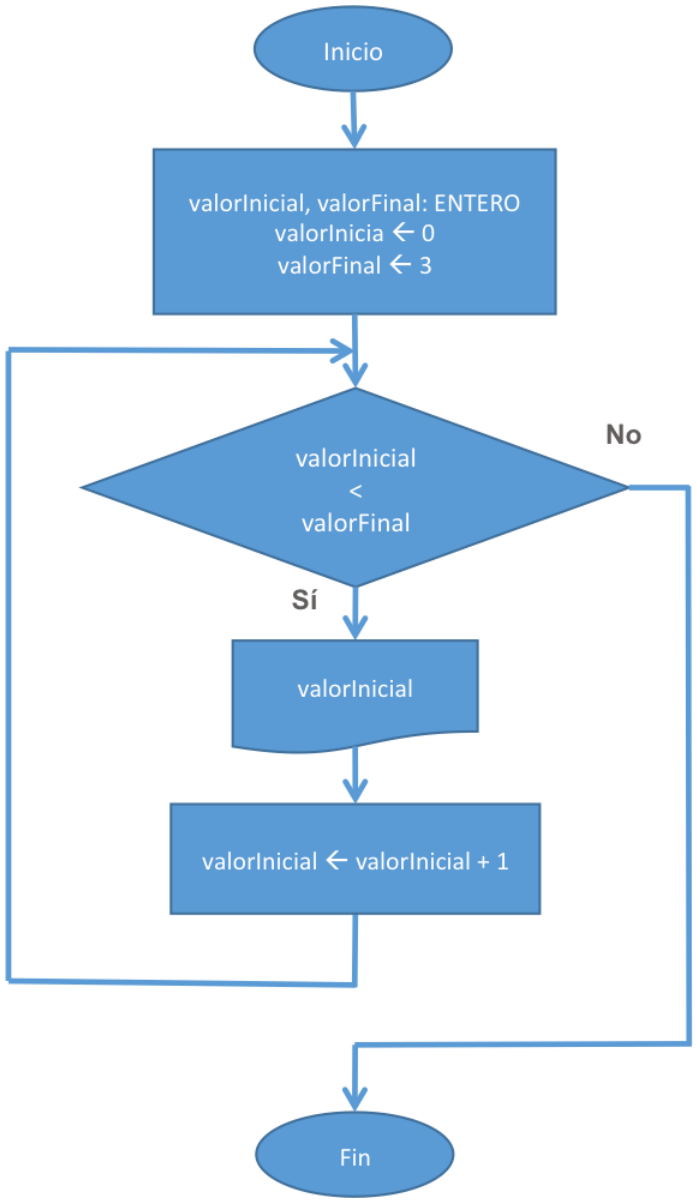
Estructuras de control iterativas o repetitivas

Las estructuras de control de flujo **iterativas o repetitivas** (también llamadas cíclicas) permiten ejecutar una serie de instrucciones mientras se cumpla la expresión lógica. Existen dos tipos de expresiones cíclicas MIENTRAS y HACER- MIENTRAS.

La estructura MIENTRAS primero valida la condición y si ésta es verdadera procede a ejecutar el bloque de instrucciones de la estructura, de lo contrario rompe el ciclo y continúa el flujo normal del programa.

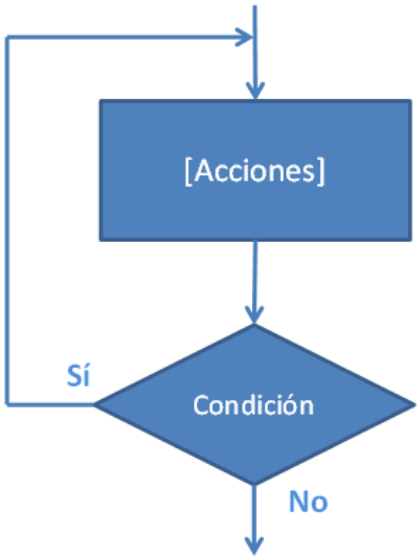


Ejemplo


Inicio
valorInicial, valorFinal: ENTERO
valorInicia ← 0
valorFinal ← 3
valorInicial
<
valorFinal
No
Sí
valorInicial
valorInicial ← valorInicial + 1
Fin

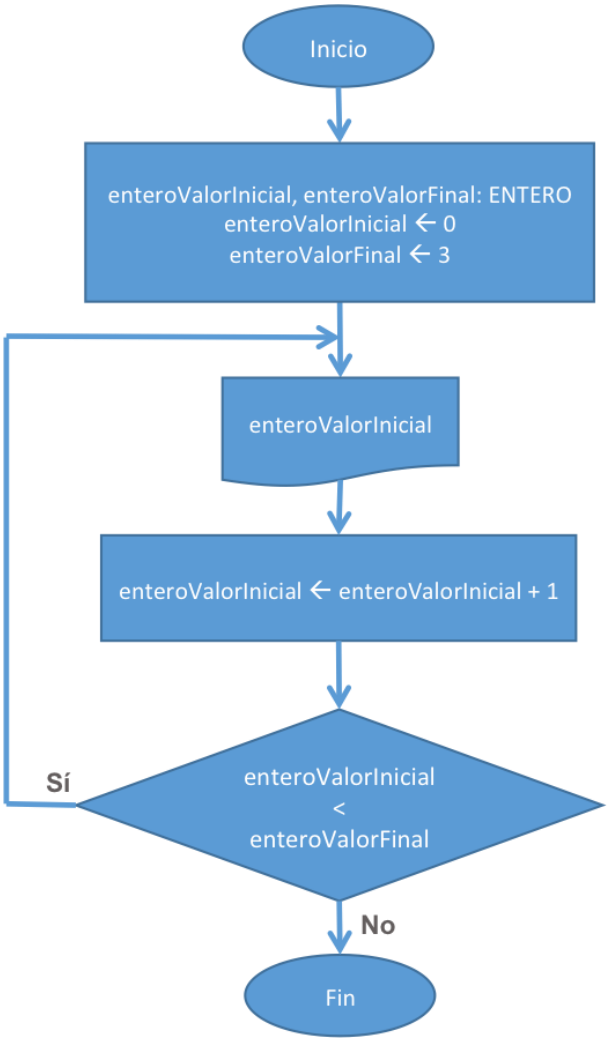
```
//>>> 0
//>>> 1
//>>> 2
```

La estructura HACER-MIENTRAS primero ejecuta las instrucciones descritas en la estructura y al final valida la expresión lógica.



Si la condición se cumple vuelve a ejecutar las instrucciones de la estructura, de lo contrario rompe el ciclo y sigue el flujo del algoritmo. Esta estructura asegura que, por lo menos, se ejecuta una vez el bloque de la estructura, ya que primero ejecuta y después pregunta por la condición.

Ejemplo

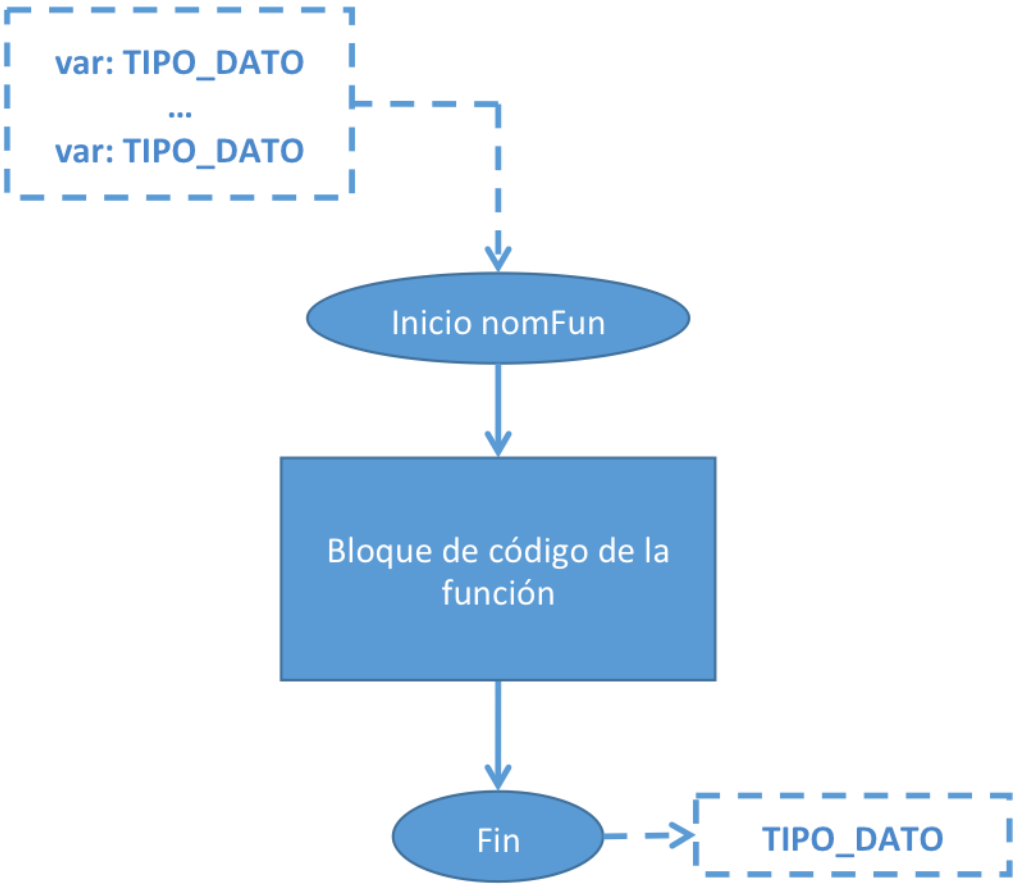

Inicio
enteroValorInicial, enteroValorFinal: ENTERO
enteroValorInicial ← 0
enteroValorFinal ← 3
enteroValorInicial
enteroValorInicial ← enteroValorInicial + 1
enteroValorInicial
<
enteroValorFinal
Sí
No
Fin

```
// >>> 0
// >>> 1
// >>> 2
```

Funciones

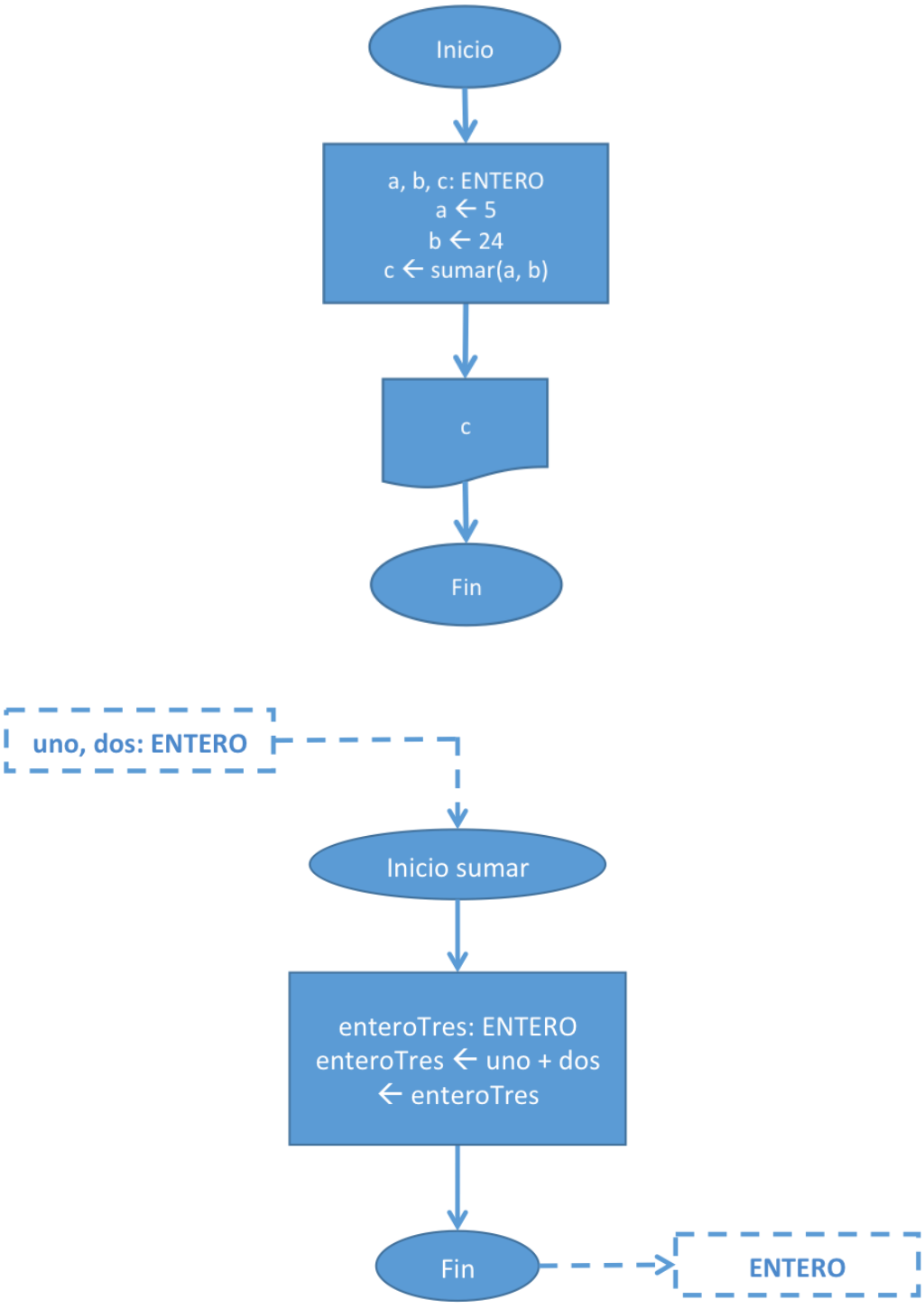
Cuando la solución de un problema es muy compleja se suele ocupar el diseño descendente (divide y vencerás). Este diseño implica la división de un problema en varios subprocesos más sencillos que juntos forman la solución completa. A estos subprocesos se les llaman módulos o funciones.

Una función está constituida por un identificador de función (nombre), de cero a n parámetros de entrada y un valor de retorno:



nomFun es el nombre con el que llama a la función. Las funciones pueden o no recibir algún parámetro (tipo de dato) como entrada, si la función recibe alguno se debe incluir en el recuadro inicial (el que apunta al nombre de la función). Todas las funciones pueden regresar un valor al final de su ejecución (un resultado) para ello se debe definir el dominio del conjunto de salida (tipo de dato).

Ejemplo


Inicio
a, b, c: ENTERO
a ← 5
b ← 24
c ← sumar(a, b)
c
Fin
uno, dos: ENTERO
Inicio sumar
enteroTres: ENTERO
enteroTres ← uno + dos
← enteroTres
Fin
ENTERO

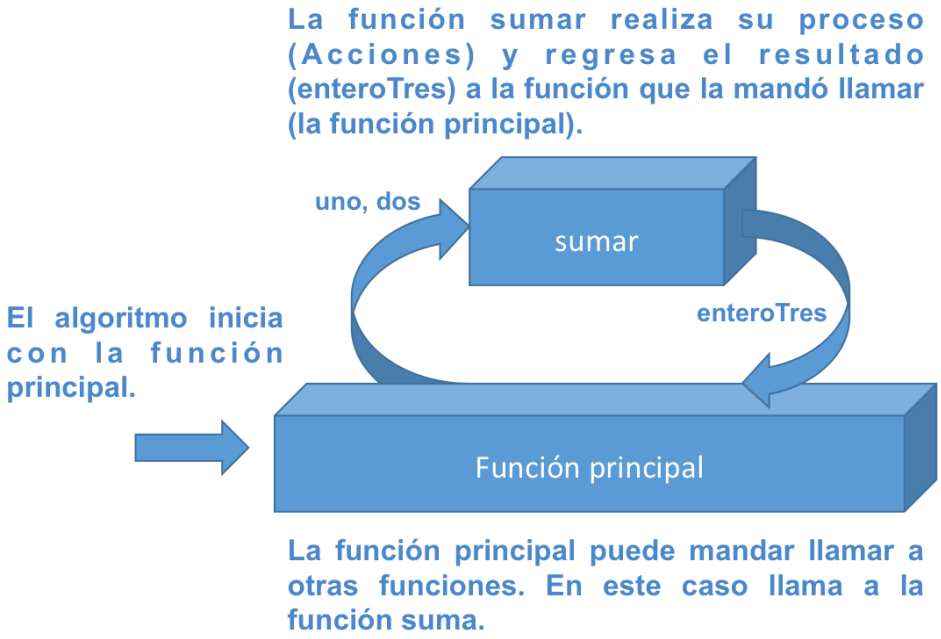
```
// >>> 29
```

Descripción

La primera función que se ejecuta es 'principal', ahí se crean las variables (uno y dos) y, posteriormente, se manda llamar a la función 'sumar'. La función 'sumar' recibe como parámetros dos valores enteros y devuelve como resultado un valor de tipo entero, que es la suma de los valores que se enviaron como parámetro.

Para la función 'principal' los pasos que realiza la función 'sumar' son transparentes, es decir, solo manda a llamar a la función y espera el parámetro de retorno.

La siguiente figura permite analizar la función a través del tiempo. El algoritmo inicia con la función principal, dentro de esta función se hace una llamada a una función externa (sumar). Sumar realiza su proceso (ejecuta su algoritmo) y devuelve un valor a la función principal, la cual sigue su flujo hasta que su estructura secuencial llega a su fin.



Bibliografía

- Metodología de la programación. Osvaldo Cairó, tercera edición, México D.F., Alfaomega 2005.

- Metodología de la programación a través de pseudocódigo. Miguel Ángel Rodríguez Almeida, primera edición, McGraw Hill