Funciones y procedimientos

Sitio: Agencia de Aprendizaje a lo largo de la Vida
Curso: Tecnicas de Programación 1° G
Libro: Funciones y procedimientos

Imprimido por: Sebastian Puche
Día: domingo, 27 de octubre de 2024, 16:28

Tabla de contenidos

1. ¿En qué se diferencian las dos rutinas?

Cuando hablamos de métodos, módulos, submódulo, rutina, subrutina, subalgoritmo, funciones o procedimientos, estamos hablando básicamente de lo mismo. ¿En qué varían unos de otros? En la época o el lenguaje en que programemos.

Cada lenguaje varía la manera que tiene de declarar las funciones. Nosotros lo haremos en base a PSeInt, que es el programa con el que estamos trabajando. PSeInt, en su código lenguaje llama “funciones” tanto a las funciones propiamente dichas como a los procedimientos.

Para modularizar los programas, utilizaremos dos tipos diferentes de rutinas: los procedimientos y las funciones.

La diferencia entre un procedimiento y una función pasa más por lo conceptual que por su estructura.

Entonces, conceptualmente, un procedimiento es utilizado cuando se quieren procesar ciertos datos, pudiendo modificar el contenido de diferentes variables en el proceso y devolviendo todos estos cambios en los contenidos al programa principal, o desde dónde haya sido llamado.

Una función, en cambio, es pensada para resolver algo en particular y que nos devuelva un único resultado.

2. ¿Cuándo usar rutinas?

En el diseño de algoritmos existen varias consideraciones que llevan a la utilización de subalgoritmos o módulos.

- En primer lugar, la técnica de dividir un problema complejo en problemas más sencillos, que permitan ser diseñados y probados independientemente unos de otros, aún por distintas personas.
- En segundo lugar, la existencia de procesos que se repiten tal cual en varias partes de un mismo algoritmo y más aún, procesos comunes a algoritmos que resuelven distintos problemas. Un módulo es invocado desde un programa, que denominaremos principal, o desde otro módulo y pueden ser:

  - Funciones: devuelve un único valor a quién lo haya llamado (por lo general el programa principal).

  Por ejemplo,

  - la función *aleatoria ()* devuelve un número en forma *random.*
  - la función *Mayúsculas(texto)* que devuelve el texto de una cadena convertido a mayúsculas.

  - Procedimientos: puede devolver cero, uno, o más valores al programa principal y lo hace a través de parámetros.

  Por ejemplo

  - Los parámetros permiten la comunicación entre el módulo llamador y el módulo que es llamado, es decir, permiten el manejo de datos de entrada y salida (desde y hacia el módulo).

Tené en cuenta que todos los lenguajes de programación tienen funciones incorporadas, intrínsecas o internas y otras definidas por el usuario.

Una función es una operación que toma uno o más valores llamados argumentos y produce un valor denominado resultado (valor de la función para los argumentos dados).

Cada función se invoca utilizando su nombre en una expresión con los argumentos actuales o reales encerrados entre paréntesis.

Por ejemplo

La función *potencia* la invocaríamos de la siguiente manera:

```
a = potencia(3)
```

Esto invoca a la función potencia y le pasa como argumento el valor 3. Dentro de la función se realiza el cálculo para el parámetro (el valor 3) recibido y se devuelve el resultado. Cuando finaliza la ejecución de esta línea el valor de “a” será 9.

Todo procedimiento o función tiene un nombre con el cual lo invocamos desde dentro del programa similar al uso de las variables.

2.1. Análisis de un ejemplo

Imaginemos el ejemplo de 4 amigos que juegan a ser diversos módulos de un programa… Cada uno de nuestros personajes sabrá hacer sólo una tarea ¡y nada más que una tarea!

Los nombres y tareas de cada uno de los personajes son:

“Facundo” será nuestro programa principal.

“Camila” será quien sepa *decir* números naturales.

“Luis” será quien sepa *sumar* 2 números y *devolver* un resultado.

“Mabel” será quien sepa *mostrar* un mensaje por pantalla.

Facundo le pide a Luis que sume 2 números, a lo que Luis le responde:

Facundo: Si, claro, lo sé hacer, es más, es lo único que sé hacer….. pero necesito que me pases los 2 números que querés que sume.

Entonces, Facundo se da cuenta que necesita esos números y por suerte conoce a Camila que es quien sabe dar un número.

Facundo: Camila… ¿me podés dar un número?

Camila: 8

Facundo: Camila…¿me podés dar otro número?

Camila: 4

Facundo: Luis, ¿cuál es el resultado de sumar 8 y 4?

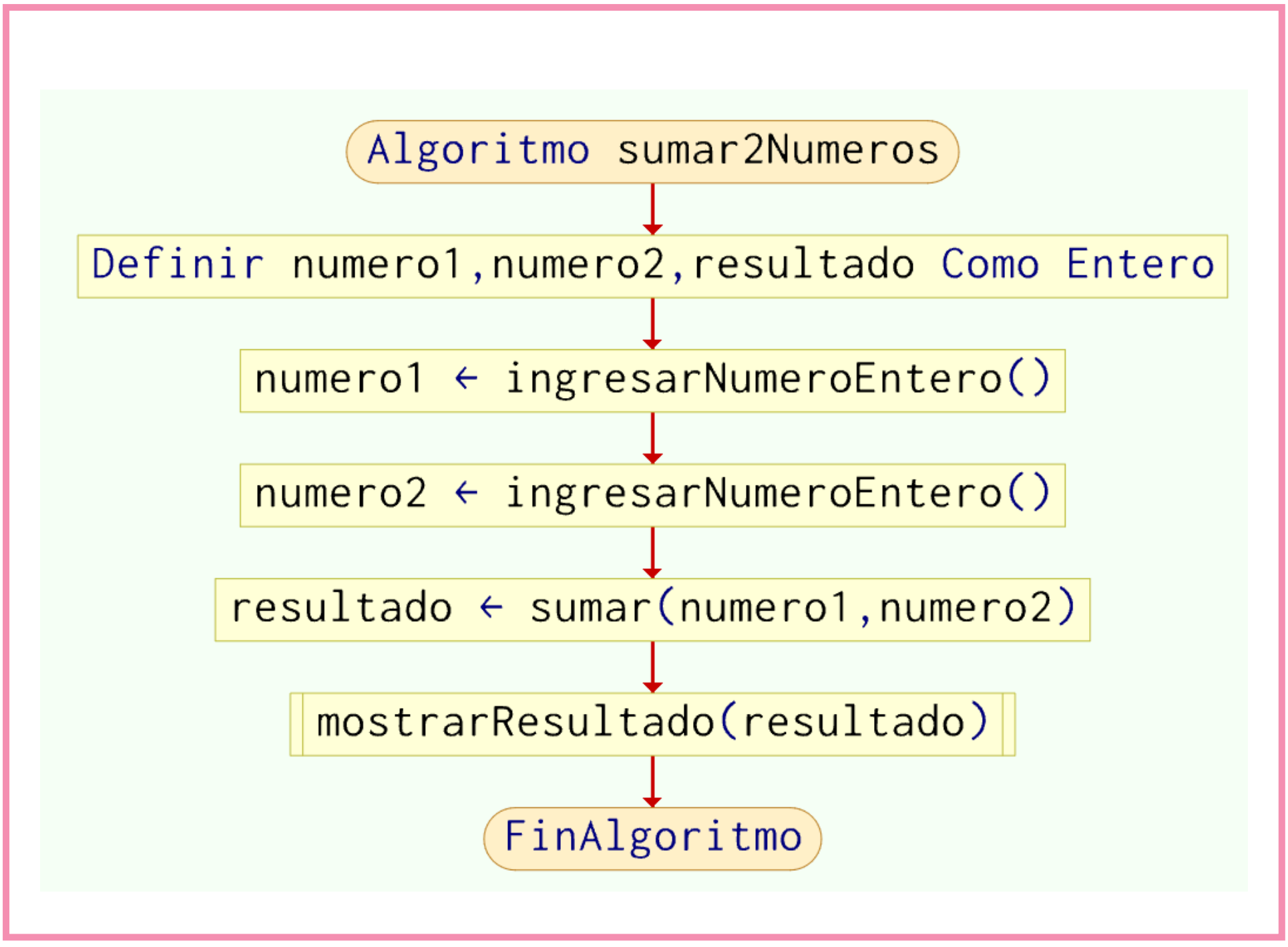
Luis: 12

Facundo: Mabel, mostrá el resultado.

En este “jueguito” que pusimos como ejemplo, vemos como nuestro programa principal va pidiendo lo que necesita a los distintos módulos.

2.2. Con PSeInt

Veamos cómo queda el ejemplo anterior pero con PSeInt el programa principal



En el diagrama hay llamadas a rutinas:

- 2 funciones (*ingresarNumeroEntero* y *sumar*)
- 1 procedimiento (*mostrarResultado*).

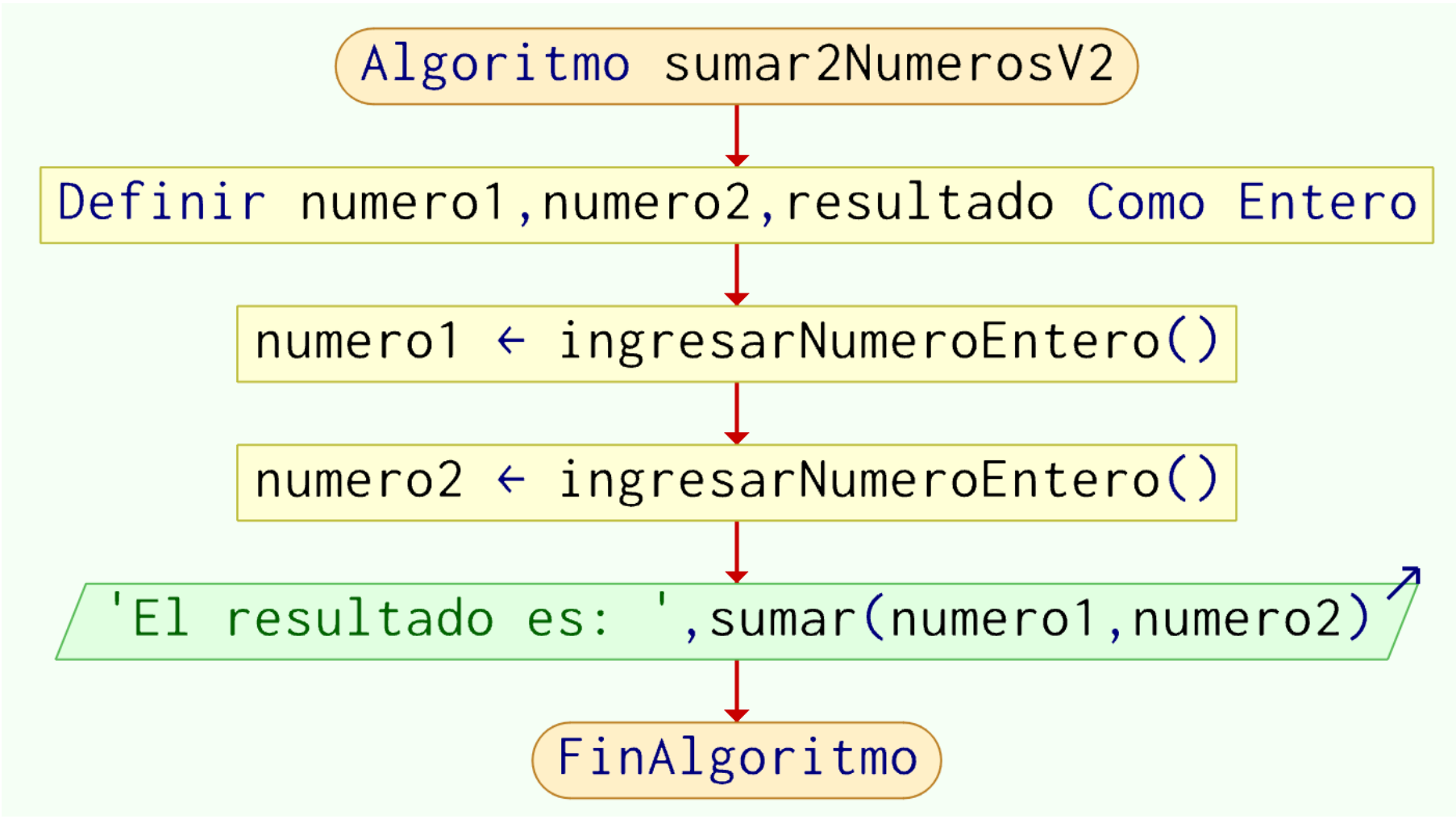
- ¿Por qué las primeras son funciones? Como dijimos, porque devuelven siempre un valor.
- ¿Por qué (*mostrarResultado*) es un procedimiento? Como dijimos, porque no devuelve ningún valor, sólo lo recibe y lo muestra por pantalla.

El diagrama de flujo es levemente distinto para distinguir entre una función y un procedimiento.

¿Pero qué diferencia, además de la cantidad de valores que devuelve un módulo, hay entre una función y un procedimiento?

- Con el valor que retorna una función, podemos:

1. Guardar en una variable (como en el ejemplo anterior).
2. Mostrarlo en forma directa.
3. Utilizarlo como argumento de llamada de otra función o de sí misma.



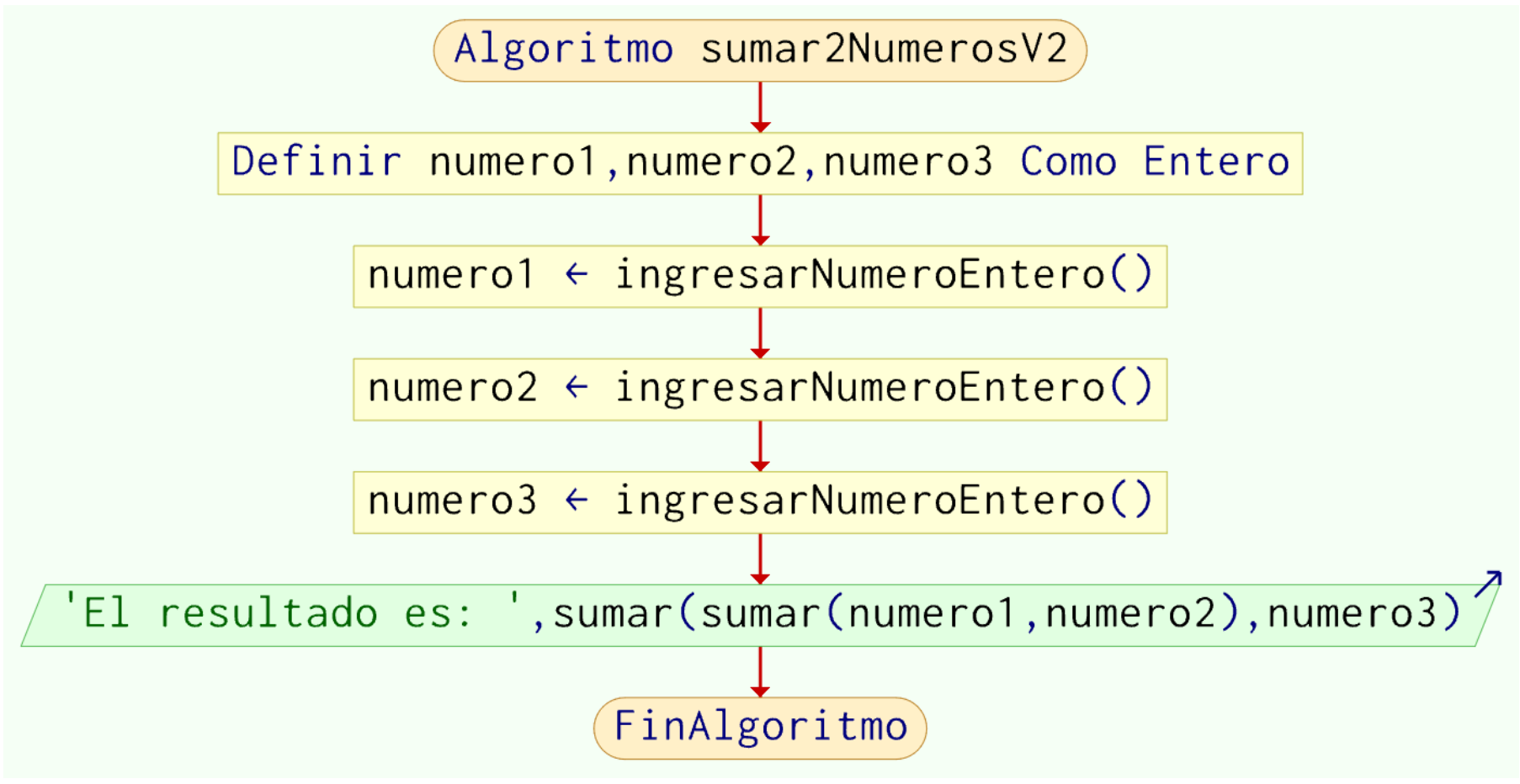
En el ejemplo anterior, modificamos el algoritmo y vemos como en una salida por pantalla podemos llamar a la función sumar pasándole como argumento los 2 números pedidos, y el valor que retorna, es mostrado a continuación del texto.

- Imaginemos que queremos sumar ahora 3 números.

  Una posible solución es hacer una nueva rutina que admita 3 números y retorne la suma.

  Pero como ya tenemos un módulo que sabe *sumar 2 números*, la llamamos con los argumentos *numero1 y numero2* y al resultado que nos retorne, lo utilizamos como argumento junto con el numero3 para que nos *retorne la suma de los 3 valores.*

Nuestro algoritmo quedaría:



- ¿Y si fuesen 4 los números a sumar?

Siguiendo la misma estrategia podríamos hacer…..

```
Algoritmo sumar2NumerosV2
Definir numero1,numero2,numero3,numero4 Como Entero
numero1 ← ingresarNumeroEntero()
numero2 ← ingresarNumeroEntero()
numero3 ← ingresarNumeroEntero()
numero4 ← ingresarNumeroEntero()
'El resultado es: ',sumar(sumar(numero1,numero2),sumar(numero3,numero4))
FinAlgoritmo
```

Entonces, ¿Cómo se llama a una función? Como vimos, a una función no se la llama explícitamente, sino que se la invoca o referencia mediante un nombre y una lista de argumentos o parámetros actuales. El algoritmo o programa principal llama o invoca a la función con el nombre de esta última en una expresión seguida de una lista de argumentos que deben coincidir en cantidad, tipo y orden con los de la función que fue definida. La función devuelve siempre un único valor.

> La pregunta del millón es: ¿Cuándo armar un módulo?
>
> La respuesta es muy sencilla: siempre que detectemos una tarea.

Muchas veces, los/as programadores/as cometen el error de decir: no voy a armar un módulo para 3 líneas. Ese es un error conceptual, nunca hablamos de cantidad de sentencias que debe tener como mínimo un módulo sino de la tarea que este realiza.