UTN FRGP
TUP

Técnico Universitario en Programación

UNIVERSIDAD TECNOLÓGICA NACIONAL

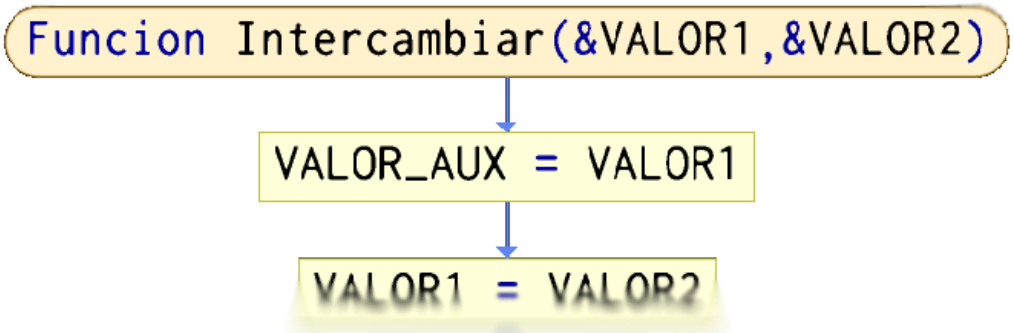
Facultad Regional Gral. Pacheco

Apuntes de clase de la asignatura

Programación I

SUBALGORITMOS

**** *FUNCIONES* ****



2024

Abel Oscar Faure

Lorena Raquel Palermo

INTRODUCCIÓN

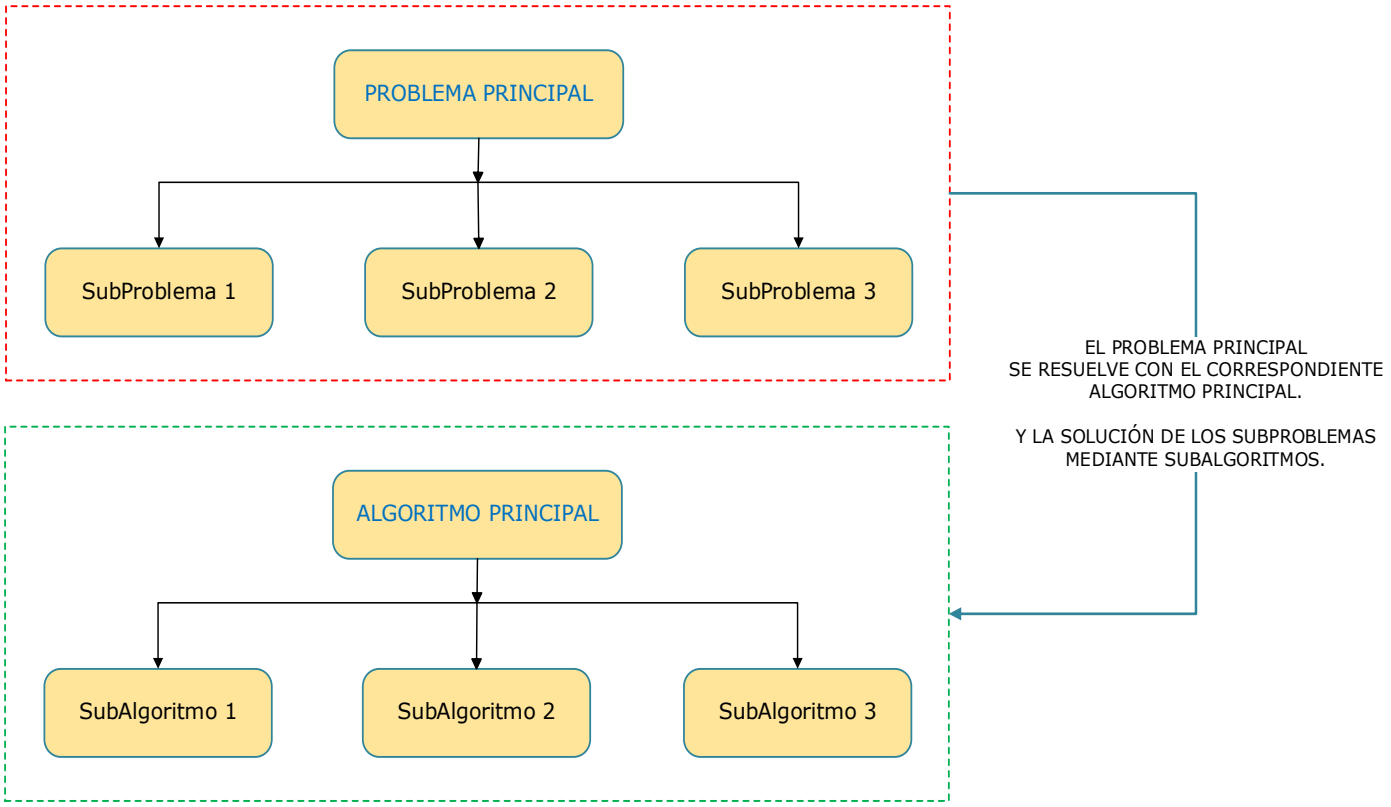
En conceptos generales, como vimos anteriormente, los paradigmas de programación definen enfoques para resolver problemas.

El paradigma imperativo, que utilizamos para diseñar algoritmos, se basa en una secuencia de órdenes que manipulan datos. Dentro de este paradigma, existen dos enfoques principales: la programación estructurada, que organiza el código en bloques lógicos mediante secuencias, decisiones y repeticiones, y la programación modular, que divide los problemas en subproblemas, facilitando su resolución paso a paso.

Ambos enfoques permiten diseñar programas más claros, fáciles de entender y mantener.

La resolución de problemas complejos se simplifica al dividirlos en problemas más pequeños, llamados subproblemas.

Estos se resuelven mediante subalgoritmos, una técnica conocida como "divide y vencerás" (divide and conquer). Este método, llamado diseño descendente (top-down design), consiste en partir de un problema general y descomponerlo en subproblemas específicos. Cada parte o subalgoritmo puede desarrollarse de manera independiente.



Los subproblemas, desde el punto de vista algorítmico, se llaman subprogramas o subalgoritmos. Dependiendo del lenguaje de programación, los subalgoritmos pueden ser de dos tipos: funciones y procedimientos (o subrutinas). Estos subalgoritmos realizan tareas específicas y se escriben solo una vez, lo que permite su reutilización en distintas partes del programa, evitando duplicaciones innecesarias. Además, al ser independientes, es más fácil detectar errores y comprender el programa.

Un subalgoritmo puede hacer lo mismo que un algoritmo:

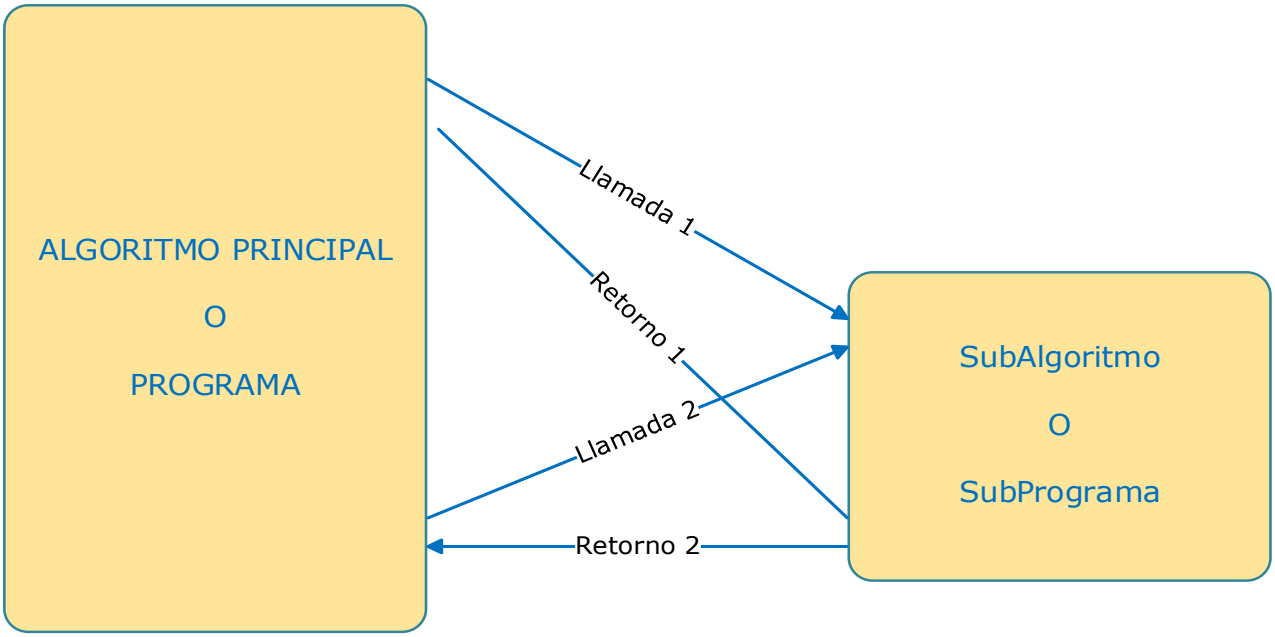
1- Recibir Datos.

2- Realizar Cálculos.

3- Devolver Resultados.

Sin embargo, se utiliza dentro del algoritmo para cumplir una tarea específica. El algoritmo principal llama o invoca al subalgoritmo, le pasa datos, y este último devuelve los resultados tras completar la tarea.

Haciendo un paralelismo con una oficina, el algoritmo principal sería como un jefe que asigna tareas a sus empleados (los subalgoritmos). Cuando el empleado termina la tarea, devuelve los resultados al jefe. Cada vez que se invoca un subalgoritmo, el control regresa al punto del algoritmo principal o SubAlgoritmo donde fue llamado.



Los SubAlgoritmos están diseñados para realizar tareas específicas: reciben una lista de valores, conocidos como argumentos, y devuelven un único valor.

SUBALGORITMOS

Hay dos aspectos clave en el uso de un subalgoritmo: la construcción (o definición) y la llamada.

Definición o construcción del subalgoritmo:

La definición es la parte donde se escribe y organiza el subalgoritmo para realizar una tarea específica. Esta parte incluye:

Nombre:

El nombre es lo que identifica al subalgoritmo y permite referirse a él cuando se necesita invocarlo o llamarlo desde otro punto del algoritmo. Este nombre debe ser único dentro del contexto del algoritmo y debe describir, de manera clara y concisa, la tarea o función que realiza el subalgoritmo.

Parámetros o Argumentos:

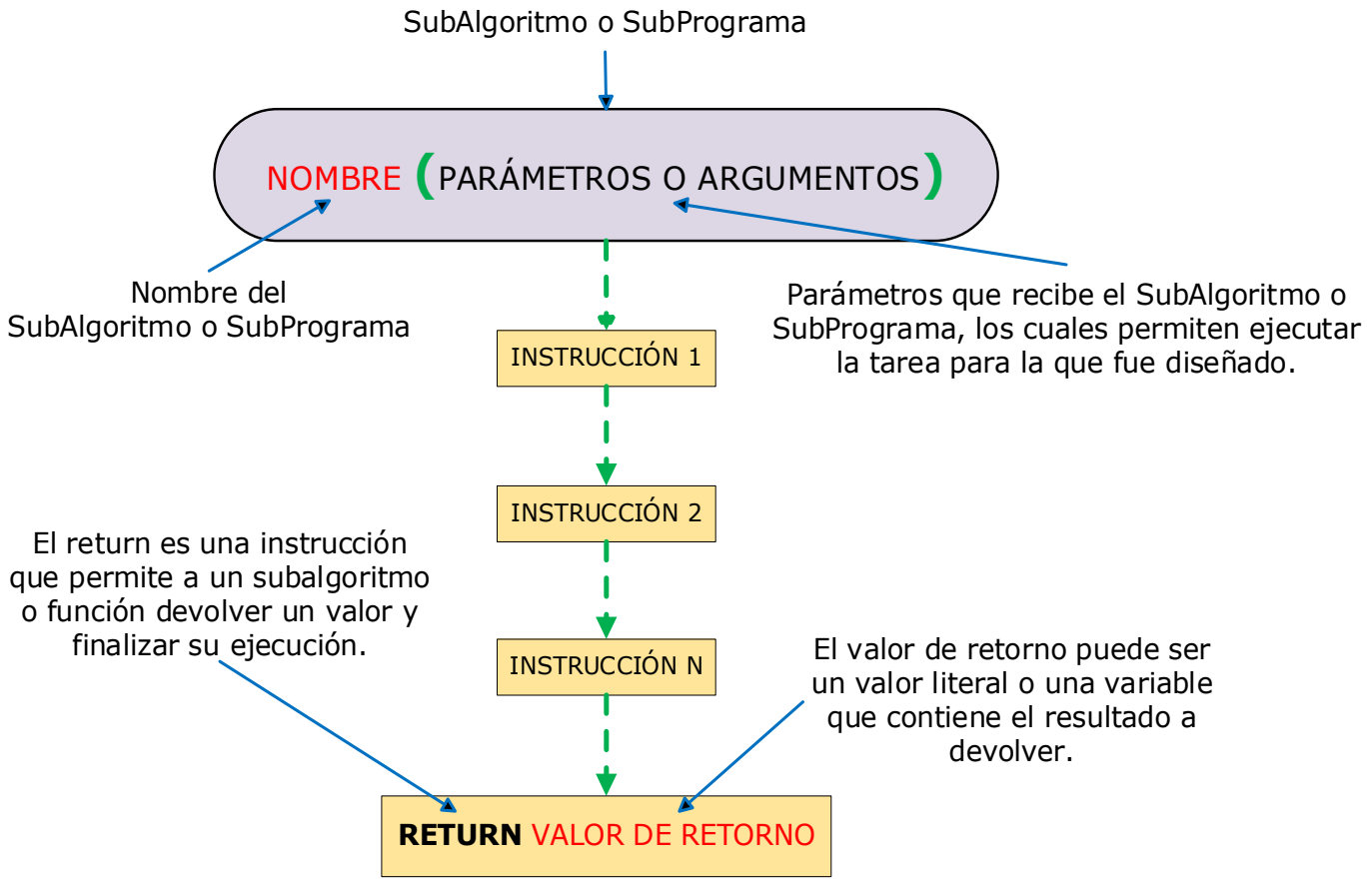
Los valores que recibe el subalgoritmo para trabajar se conocen como parámetros. Estos parámetros son los datos que el subalgoritmo necesita para realizar su tarea. Más adelante explicaremos los diferentes tipos de parámetros que puede recibir, como parámetros por valor y parámetros por referencia, así como sus características y cómo influyen en el comportamiento del subalgoritmo. También es posible que un subalgoritmo no reciba parámetros, en cuyo caso se ejecutará con los datos disponibles internamente.

Cuerpo:

El cuerpo de un subalgoritmo es el conjunto de instrucciones que este ejecuta para cumplir su función. Estas instrucciones pueden incluir operaciones, decisiones (como estructuras condicionales), bucles (como for o while), y cualquier otro tipo de procesamiento necesario. El cuerpo define el comportamiento del subalgoritmo y determina los pasos que seguirá al ser invocado.

Valor de retorno:

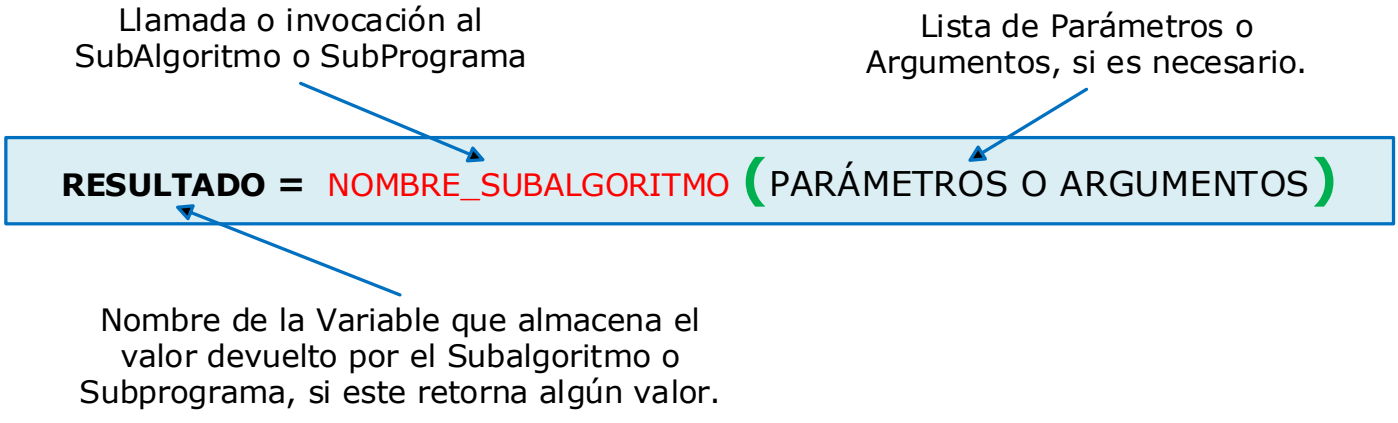
El valor de retorno es el resultado que un subalgoritmo devolverá al ser ejecutado, si así lo requiere. No todos los subalgoritmos tienen un valor de retorno, pero cuando aplica, este valor puede ser utilizado por el algoritmo principal u otros subalgoritmos para continuar con el procesamiento. El valor de retorno es especialmente importante en funciones que realizan cálculos o transformaciones, ya que permite obtener un resultado tras la ejecución del subalgoritmo.



Llamada al subalgoritmo:

La llamada al subalgoritmo es la parte donde lo invocas desde otro lugar del algoritmo, ya sea dentro del cuerpo del algoritmo principal o desde otro subalgoritmo.

En este momento, se le puede pasar un conjunto de valores o argumentos para que el subalgoritmo los procese, aunque también es posible que no reciba datos, dependiendo de cómo esté diseñado.

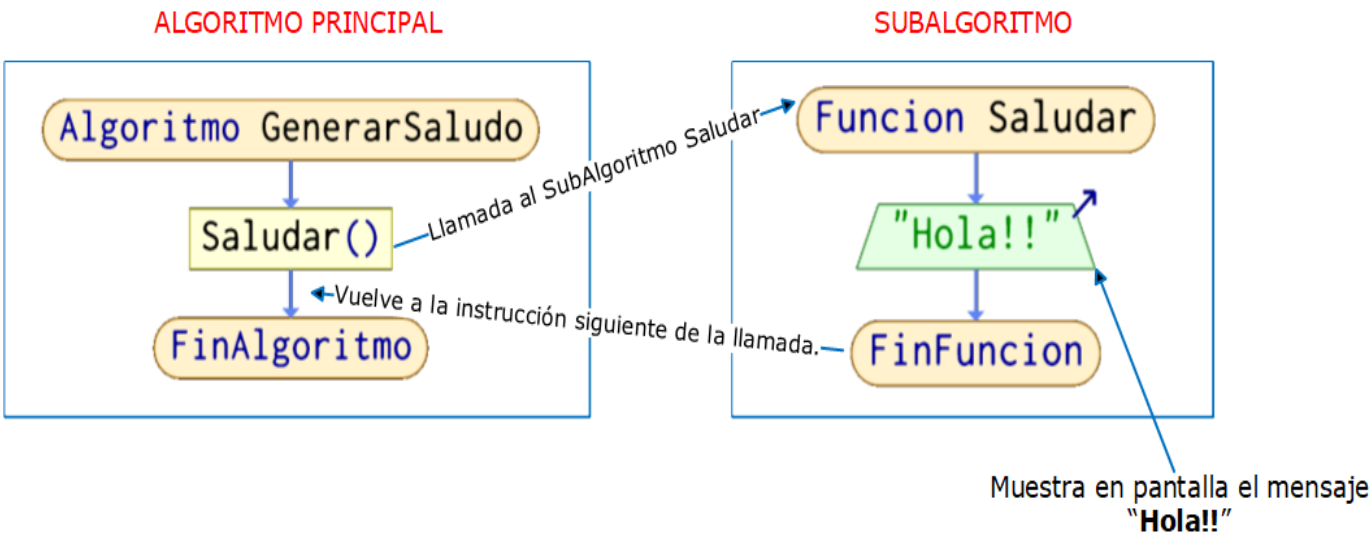


Conclusión:

La definición o construcción de un subalgoritmo incluye su nombre, los parámetros que recibe, el conjunto de instrucciones o cuerpo, el valor de retorno (si aplica). Por otro lado, la llamada al subalgoritmo es el momento en que se invoca desde otro punto del Algoritmo Principal o SubAlgoritmo, pasándole los argumentos necesarios para que procese y, en caso de tenerlo, devuelva un resultado

Además, se recomienda utilizar verbos en infinitivo para nombrar a los subalgoritmos, ya que describen claramente la acción que realizan. Ejemplos como calcularPromedio, obtenerDatos o procesarVenta ayudan a que el código sea más legible y comprensible, facilitando su mantenimiento.

Veamos un primer ejemplo de un SubAlgoritmo que no recibe parámetros ni retorna ningún valor. Crearemos un SubAlgoritmo que simplemente mostrará un mensaje con el texto "Hola!!".



Como podemos ver en la construcción del SubAlgoritmo, el cuerpo contiene únicamente una instrucción que muestra el texto “Hola!!” en pantalla. Este SubAlgoritmo no recibe parámetros ni retorna ningún valor, su única función es saludar.

En el algoritmo principal, la única instrucción es llamar a este SubAlgoritmo. Una vez que se ejecutan las instrucciones del SubAlgoritmo, el control regresa al punto de llamada.

En este caso, como no hay más instrucciones en el algoritmo principal, la ejecución termina. En pantalla se habrá mostrado el mensaje "Hola!!".

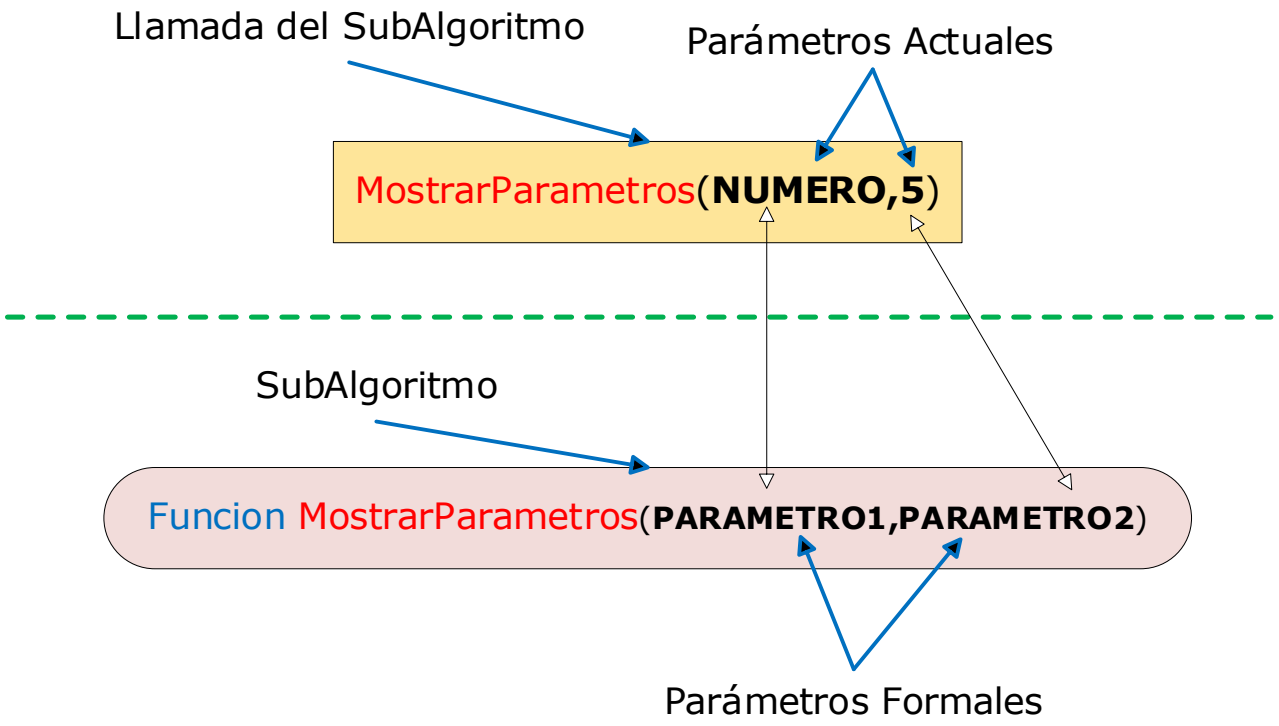
SUBALGORITMOS – PASO DE PARÁMETROS

En el ejemplo anterior, creamos un SubAlgoritmo que no recibía datos ni devolvía ningún valor. Sin embargo, si necesitamos llamar a un SubAlgoritmo y enviarle información para que realice una tarea específica, utilizamos algo llamado parámetros. Los parámetros son como pequeñas "cajas" que contienen los datos que el SubAlgoritmo utilizará para llevar a cabo la tarea requerida.

Existen dos tipos de parámetros:

Los parámetros actuales son los datos que el algoritmo principal le pasa al SubAlgoritmo cuando lo llama.

Los parámetros formales son los nombres que usa el SubAlgoritmo para recibir esos datos.



Imagina que los parámetros actuales son como **"invitados"** y los parámetros formales son como **"sillas en una mesa"**. Cuando llamas al SubAlgoritmo, los invitados (los datos) se sientan en las sillas (los parámetros formales) para que el SubAlgoritmo pueda trabajar con ellos.

Existen dos formas principales de enviar estos datos o parámetros al subalgoritmo:

- **Paso por valor.**
- **Paso por referencia.**

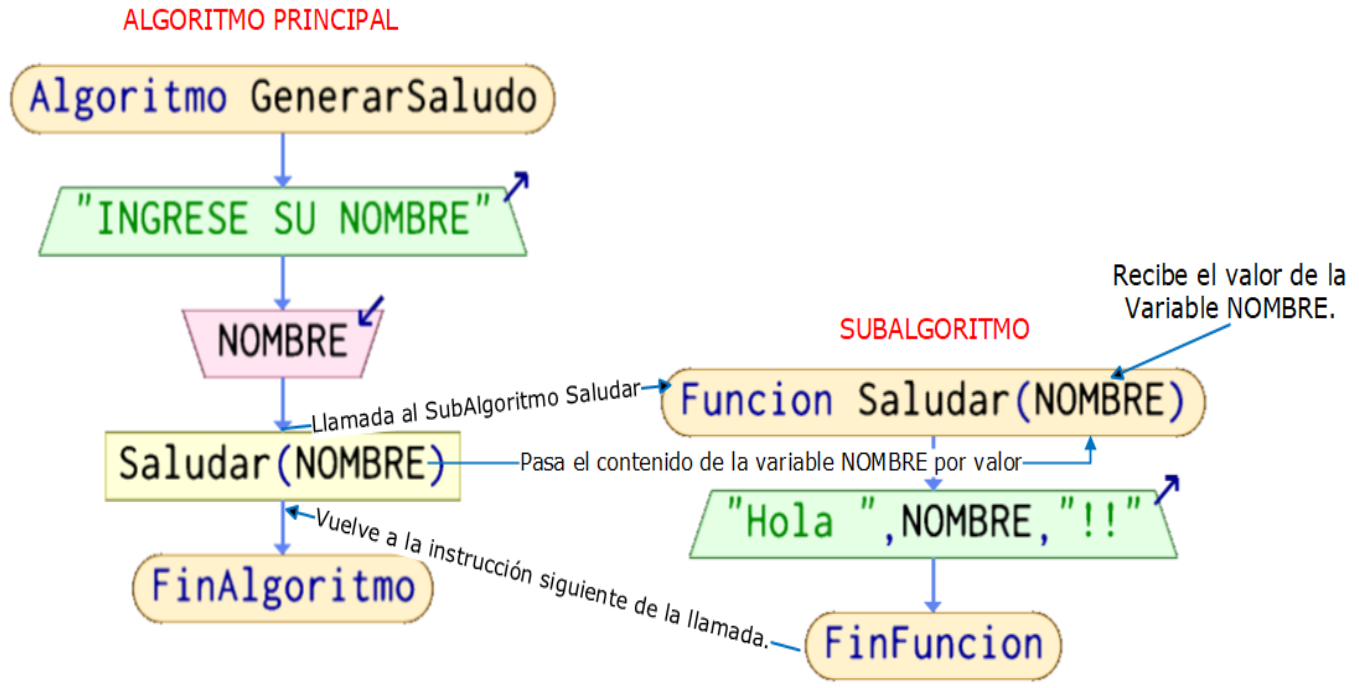
Paso por valor:

Cuando utilizamos el paso por valor, el SubAlgoritmo recibe una copia del valor que le pasamos. Es como si se le entregara una fotocopia del dato original, y el SubAlgoritmo trabaja solo con esa copia, no con el dato original.

Esto significa que, si el SubAlgoritmo modifica ese valor, la modificación solo afecta a la copia, pero el valor original que está en el algoritmo principal no cambia. Es como si escribieras en la fotocopia, pero el documento original permanece intacto.

Por ejemplo, si tienes un número en el algoritmo principal y lo pasas por valor a un SubAlgoritmo, este puede cambiar el número dentro del SubAlgoritmo, pero cuando la ejecución regresa al algoritmo principal, el número original no habrá cambiado. Este método es útil cuando no quieres que el SubAlgoritmo modifique los valores originales, sino que solo trabaje con una copia temporal.

Veamos un ejemplo sencillo de paso de parámetros por valor. Supongamos que, en el ejemplo anterior, el SubAlgoritmo no solo muestra el mensaje "Hola!!", sino que le agregamos el nombre que el usuario ingresa en el algoritmo principal. Es decir, el algoritmo principal pedirá un nombre, y luego el subalgoritmo mostrará un saludo personalizado con ese nombre. En este caso, el nombre ingresado se enviará al SubAlgoritmo como parámetro por valor, y el SubAlgoritmo usará ese dato para generar el mensaje completo sin modificar el valor original en el algoritmo principal.

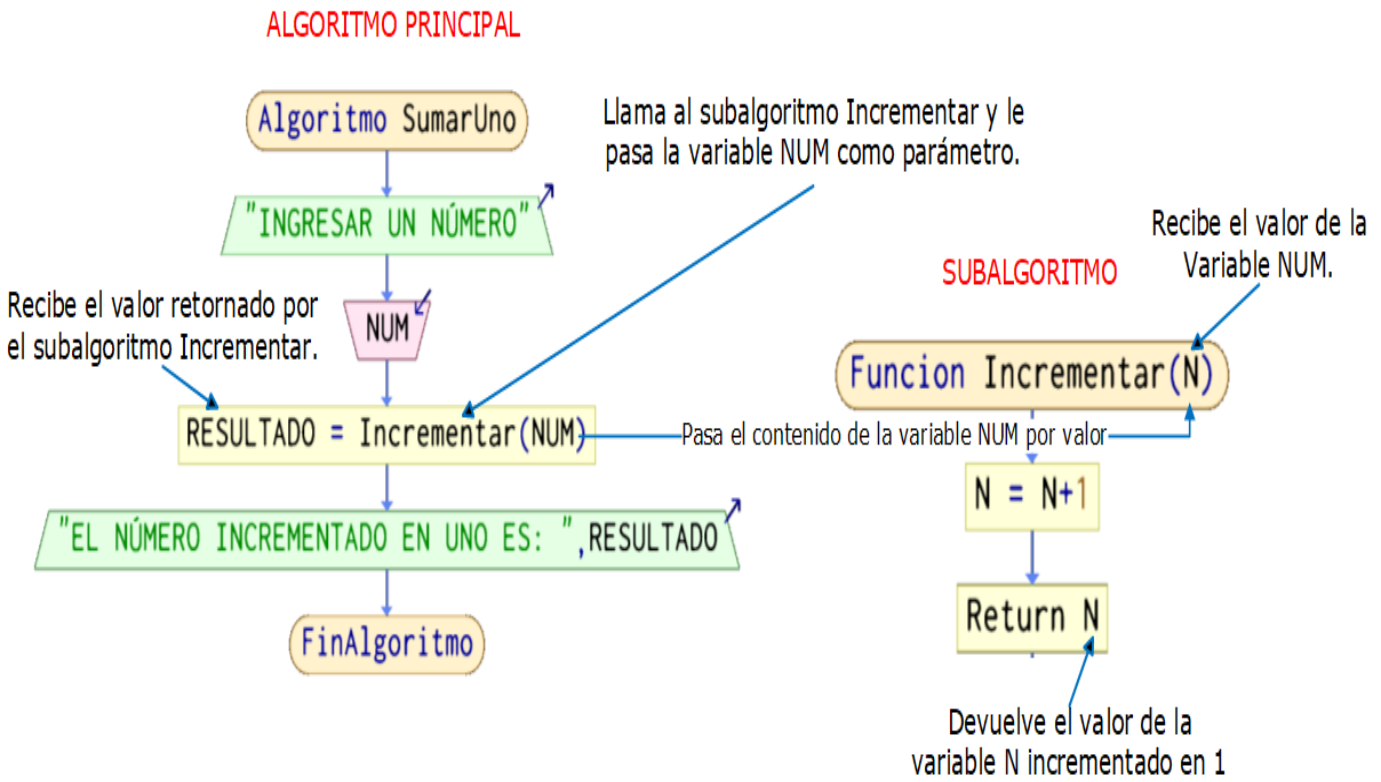


Cuando se ejecuta el algoritmo principal, se crea su propio espacio en la memoria, llamado ámbito o mapa de memoria, donde se guardan todas sus variables, como **NOMBRE**. Este espacio es independiente y exclusivo del algoritmo principal.

Cuando el SubAlgoritmo **Saludar** es llamado, también se crea su propio mapa de memoria o ámbito. Dentro de este SubAlgoritmo, existe otra variable llamada **NOMBRE,** pero, aunque tenga el mismo nombre que la variable del algoritmo principal, son dos variables diferentes porque cada una vive en su propio ámbito o espacio de memoria.

Al terminar la ejecución del SubAlgoritmo, su mapa de memoria, junto con la variable **NOMBRE** que estaba allí, se destruye. El valor de **NOMBRE** en el algoritmo principal queda intacto, ya que nunca se modificó directamente. Esto ocurre porque los dos ámbitos son independientes, y lo que ocurre en el SubAlgoritmo no afecta al algoritmo principal.

Ahora veamos un ejemplo donde el SubAlgoritmo retorna un valor. Supongamos que necesitamos un algoritmo que pida un número al usuario, y luego llame a un SubAlgoritmo que se encargue de incrementar ese número en 1. El SubAlgoritmo devolverá el número incrementado al algoritmo principal, donde se mostrará el resultado final al usuario.



En este caso, el SubAlgoritmo finaliza con la instrucción Return, devolviendo el valor de **N** incrementado en 1. Este valor devuelto por el SubAlgoritmo **Incrementar** se asigna a la variable **RESULTADO** en el algoritmo principal.

Paso por referencia:

El paso por referencia es una forma de enviar datos a un SubAlgoritmo donde, en lugar de enviar una copia del valor, se envía una referencia de la variable original.

Esto significa que el SubAlgoritmo no trabaja con una copia, sino con la variable original que se encuentra en el algoritmo principal.

¿Cómo funciona?

Cuando se usa el paso por referencia, el SubAlgoritmo accede directamente a la ubicación de memoria de la variable que se pasa como parámetro. Es como si el SubAlgoritmo recibiera las "llaves" de la variable original. Entonces, si el SubAlgoritmo modifica el valor de la variable, este cambio se reflejará directamente en el algoritmo principal, ya que ambos (algoritmo principal y SubAlgoritmo) están trabajando sobre la misma variable.

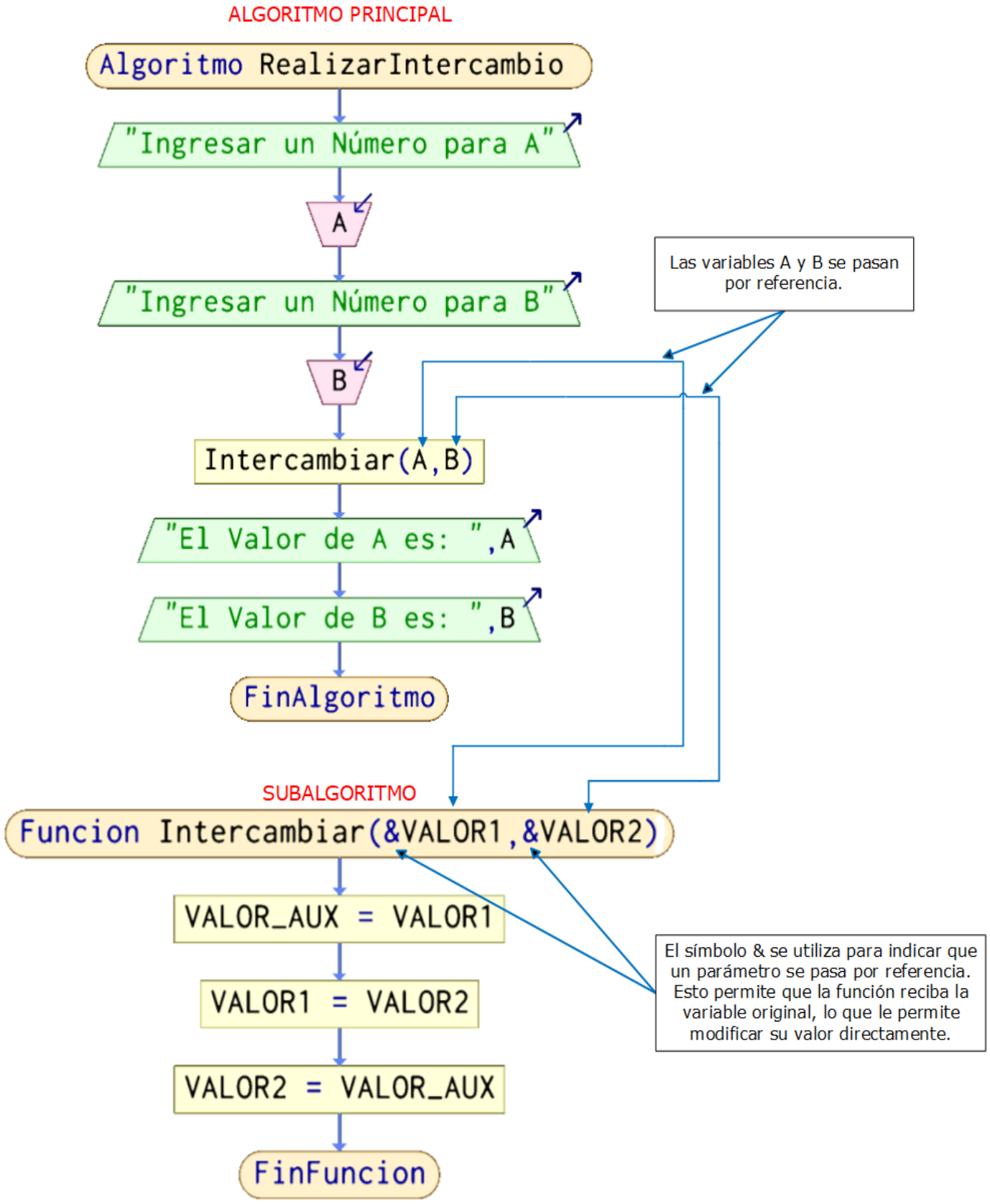
Esto es importante porque, en el caso de las funciones, solo pueden devolver un único valor a través de la instrucción Return. Sin embargo, en muchas ocasiones, necesitamos devolver más de un valor o realizar múltiples cambios en el programa principal.

El paso por referencia es útil en estos casos, ya que permite que el SubAlgoritmo modifique directamente varias variables en el algoritmo principal, sin limitarse a devolver un solo resultado.

Veamos un ejemplo que nos pide crear un algoritmo que solicite al usuario ingresar dos números, los cuales se guardarán en dos variables diferentes. A continuación, se utilizará un SubAlgoritmo para intercambiar los valores de estas variables. Finalmente, se mostrarán los valores de las variables ya intercambiados.

Ejemplo: Si el usuario ingresa 3 y 8, y las variables utilizadas son A y B (donde A=3 y B=8), después de llamar al SubAlgoritmo, los valores deberán quedar intercambiados, es decir: A=8 y B=3.

El SubAlgoritmo será responsable de realizar el intercambio de los valores.



Este ejercicio consiste en crear un algoritmo principal que solicite al usuario ingresar dos números, los cuales se almacenarán en las variables **A** y **B**. A continuación, se llama a un SubAlgoritmo llamado **Intercambiar**, que recibe los valores de **A** y **B** por referencia y es responsable de intercambiar los valores de am-

bas variables. La clave del ejercicio está en el uso del paso por referencia, que permite que cualquier modificación realizada dentro del SubAlgoritmo afecte directamente a las variables originales del algoritmo principal.

Al iniciar, el algoritmo principal solicita al usuario que ingrese dos números. Supongamos que el usuario introduce los valores 3 y 8, donde **A** se almacena con el valor **3** y **B** con el valor **8**. Luego, se llama al SubAlgoritmo **Intercambiar**, pasando por referencia las variables **A** y **B** como alias **VALOR1** y **VALOR2**, respectivamente. Esto significa que **VALOR1** y **VALOR2** son referencias directas a las variables originales **A** y **B**. El operador & utilizado en el SubAlgoritmo indica que lo que se recibe es la variable original permitiendo su modificación.

Dentro del SubAlgoritmo, se utiliza una variable auxiliar llamada **VALOR_AUX** (que solo existe dentro del ámbito o mapa de memoria del SubAlgoritmo **Intercambiar**) para llevar a cabo el intercambio. Primero, el valor de **VALOR1** (es decir, el valor de **A**) se almacena en **VALOR_AUX**. A continuación, el valor de **VALOR2** (el valor de **B**) se asigna a **VALOR1**, y finalmente, el contenido de VALOR_AUX (que contenía el valor original de **A**) se asigna a **VALOR2**. De esta forma, los valores de **A** y **B** quedan intercambiados.

Una vez que el SubAlgoritmo ha finalizado, el control regresa al algoritmo principal. Gracias al paso por referencia, los valores de **A** y **B** han sido modificados correctamente, por lo que el algoritmo principal ahora mostrará los valores **A** = **8** y **B** = **3**. Este ejercicio demuestra cómo un SubAlgoritmo puede manipular variables del algoritmo principal mediante el uso del paso por referencia.