# Funciones

A veces en nuestros programas existen acciones que hemos de realizar varias veces a lo largo de los mismos, y que nos resultaría farragoso tener que copiar el código una y otra vez. De igual forma, cualquier modificación en algún elemento tendría que buscarse y repetirse todas las veces en que ese código saliera. Para automatizar todo esto cumpliendo el principio DRY de programación nos ayudan las **funciones.**

Las **funciones,** también llamados **métodos** o **procedimientos**, empaquetan y 'aíslan' del resto del programa una parte de código que realiza alguna tarea específica de forma que nos sea muy sencillo de manipular y reutilizar. Se definen al principio como pequeños subprogramas y después se "invocan" desde el algoritmo principal cuando se necesitan.

Las funciones pueden pueden devolver o no un resultado y también ser con o sin parámetros.

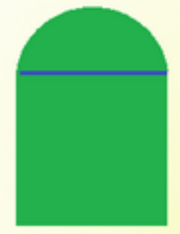
- Funciones que **devuelven resultado o no**:
  - Si la función no devuelve resultado en ocasiones se denomina **procedimiento** y es un conjunto de código con algo de relación que se invoca en el algoritmo principal de forma autónoma. Por ejemplo, la función *dibujarCuadrado (lado)* puede realizar todas las operaciones necesarias para dibujar un cuadrado dado su lado. En el algoritmo principal escribiríamos.
    - dibujarCuadrado(5);
  - Si la función devuelve resultado se comportará como un valor y siempre habrá que **almacenar dicho resultado en una variable**, por lo que se habrá de invocar dentro de una asignación y no de forma autónoma. Por ejemplo, la función *calcularAreaCuadrado(lado)* calcula la superficie de un cuadrado dada la longitud de su lado, pero en el programa principal deberá aparecer asignándose a una variable u ocupando el lugar de ella.
    - Resultado=calcularAreaCuadrado(5);
- **Parámetros** de una función: valores que necesita la función para poder ejecutarse. Por ejemplo una función que dibuja un cuadrado necesita que le pasen el valor del lado para dibujarlo. Sin embargo una función que realice una cuenta atrás del 10 al 1 no necesita ningún parámetro.En ese caso se escribe el paréntesis pero vacío.
  - dibujarCuadrado(10);
  - contarAtras10a1();

Copyright 2026

> Se podría parametrizar la función contarAtras (n) indicándole a la función el número a partir del que realizar la cuenta atrás. Te invitamos a explorar cómo sería.

Las funciones se suelen denominar con un verbo o un conjunto de varias palabras que comienza con un verbo en minúsculas; es decir, con la primera letra en minúscula y la primera letra de las palabras siguientes en mayúsculas.

Por ejemplo, vamos a imaginar un programa que calcula el área de una figura geométrica plana compuesta como la de la figura.

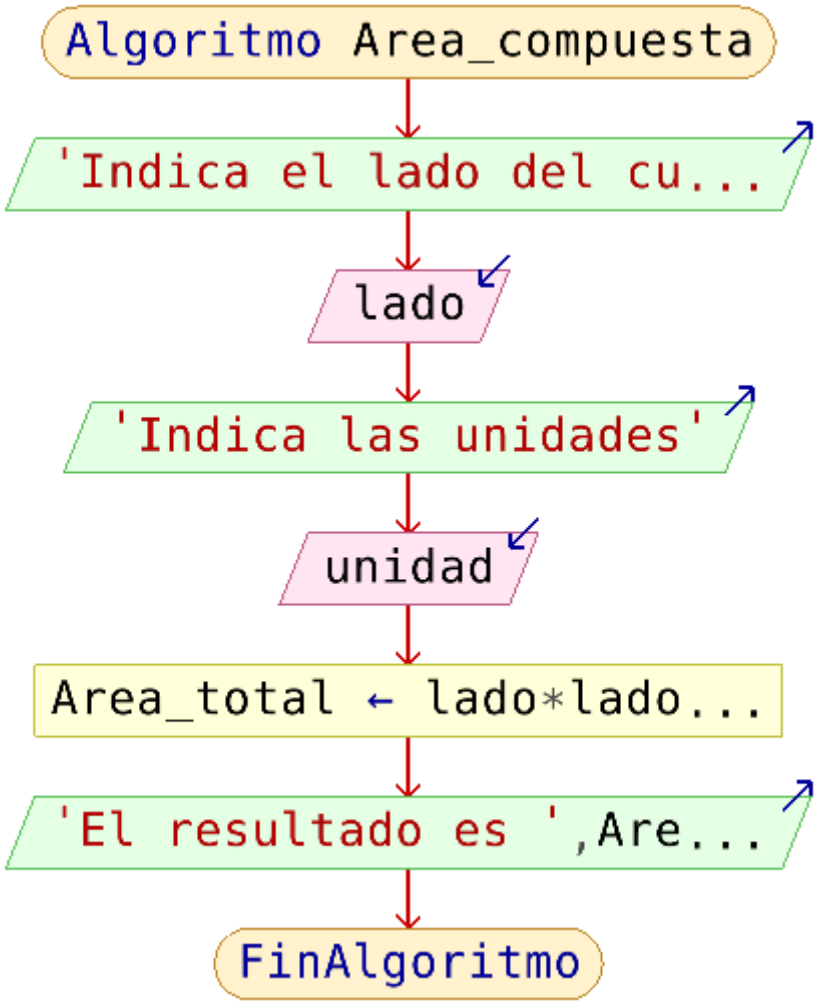


## Pasos 1 y 2: Análisis y diagramas de flujo del programa *Area compuesta*

Los elementos serían:

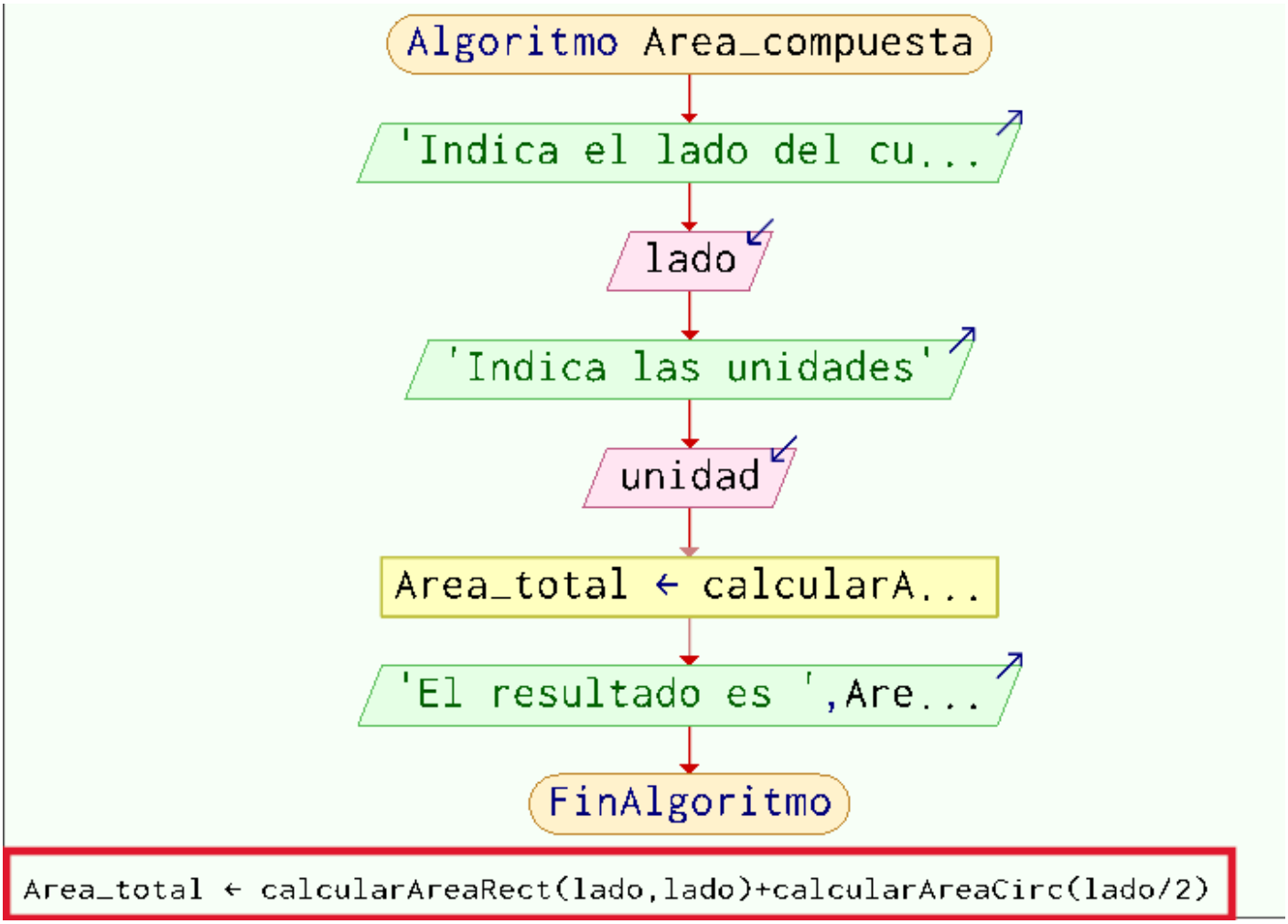
- **Inicio y fin** del algoritmo.
- **Salida**: pedir las medidas del lado del cuadrado inferior incluyendo unidades.
- **Entrada**: longitud del lado (que coincide con diámetro del círculo) y unidad de medida.
- **Almacenamiento**: lado, unidad, area_total.
- **Procesamiento**: suma, multiplicación, concatenación.

Diagrama de flujo (sin funciones):

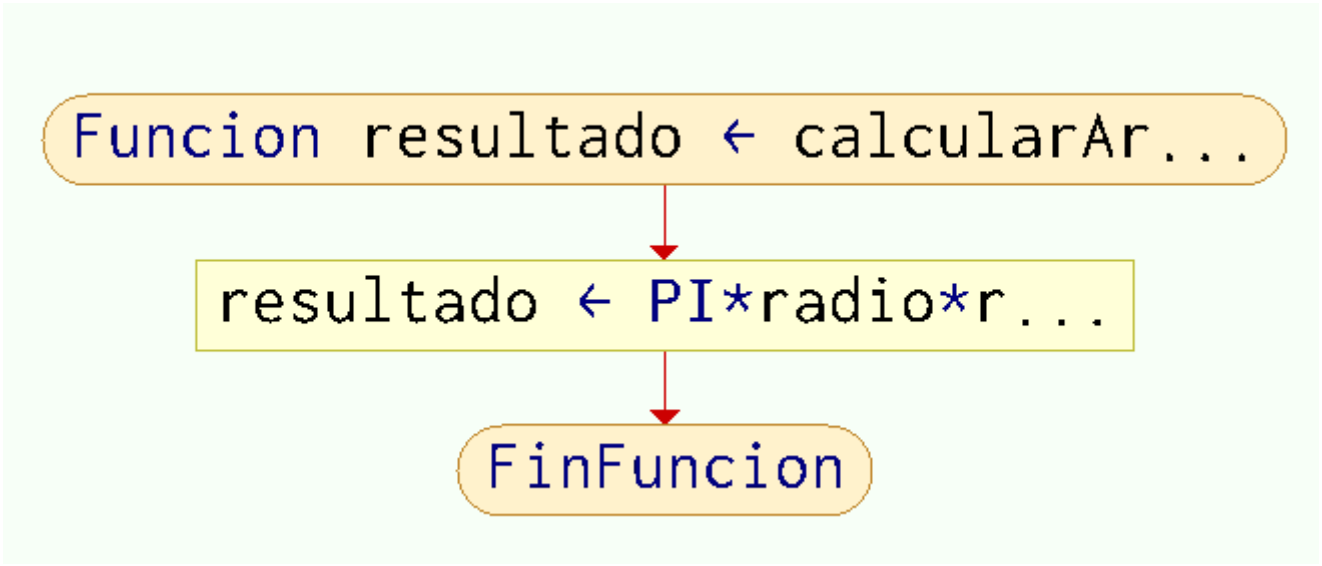



Otra forma de abordar su solución sería predefinir unas funciones que nos calcularan por ejemplo el área de un rectángulo y el área de un círculo. Veamos como quedaría el diagrama de flujo en ese caso.

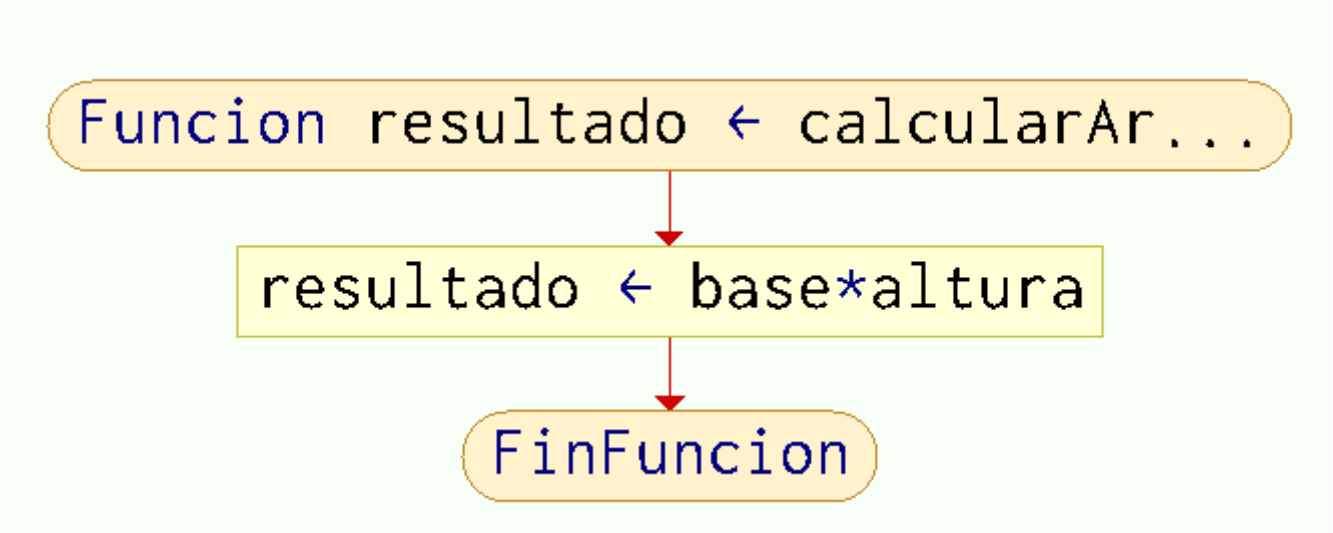
Copyright 2026




Y habría que realizar de forma independiente los diagramas de flujos de cada una de las funciones.




Copyright 2026

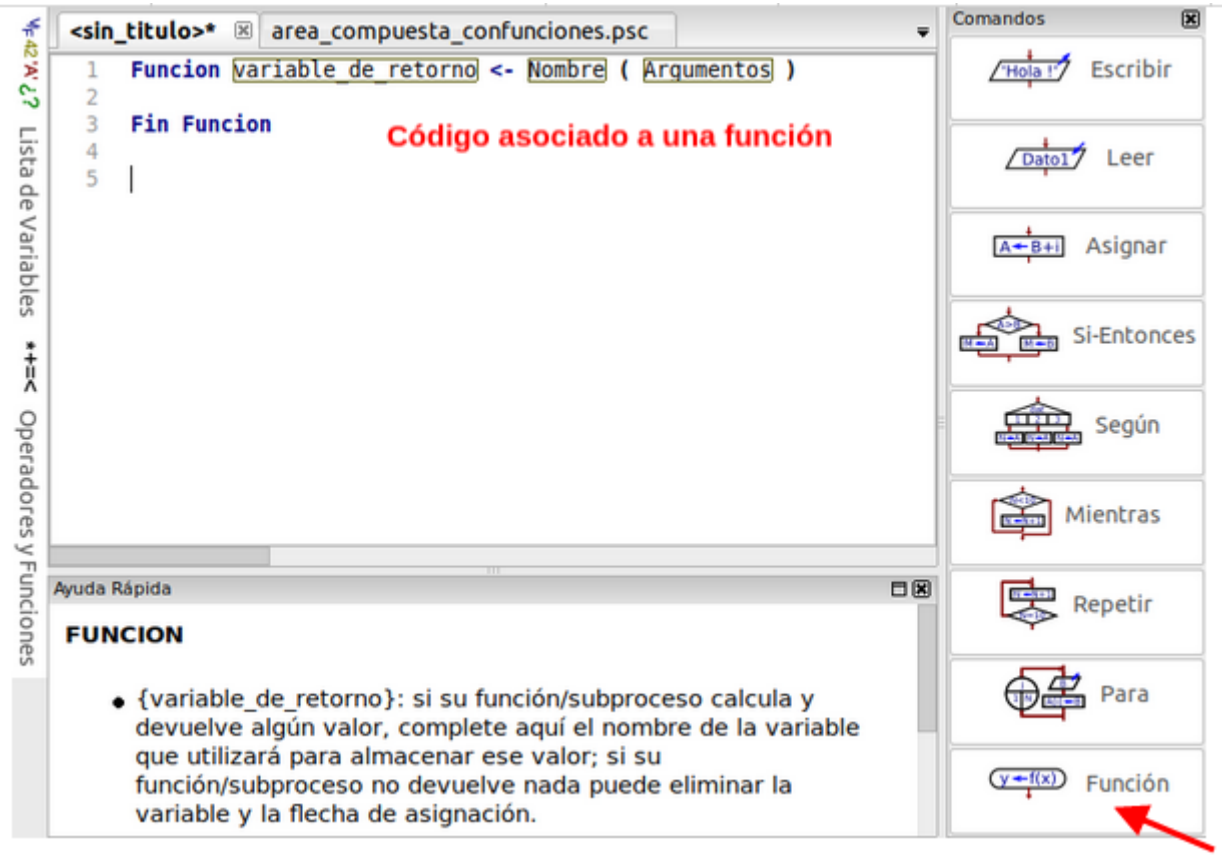



Las funciones necesarias en este caso devuelven un resultado, y tienen parámetros. Como se puede deducir fácilmente, esto tendría mucho más sentido en programas donde fuera necesario calcular áreas de diferentes figuras de forma repetida.

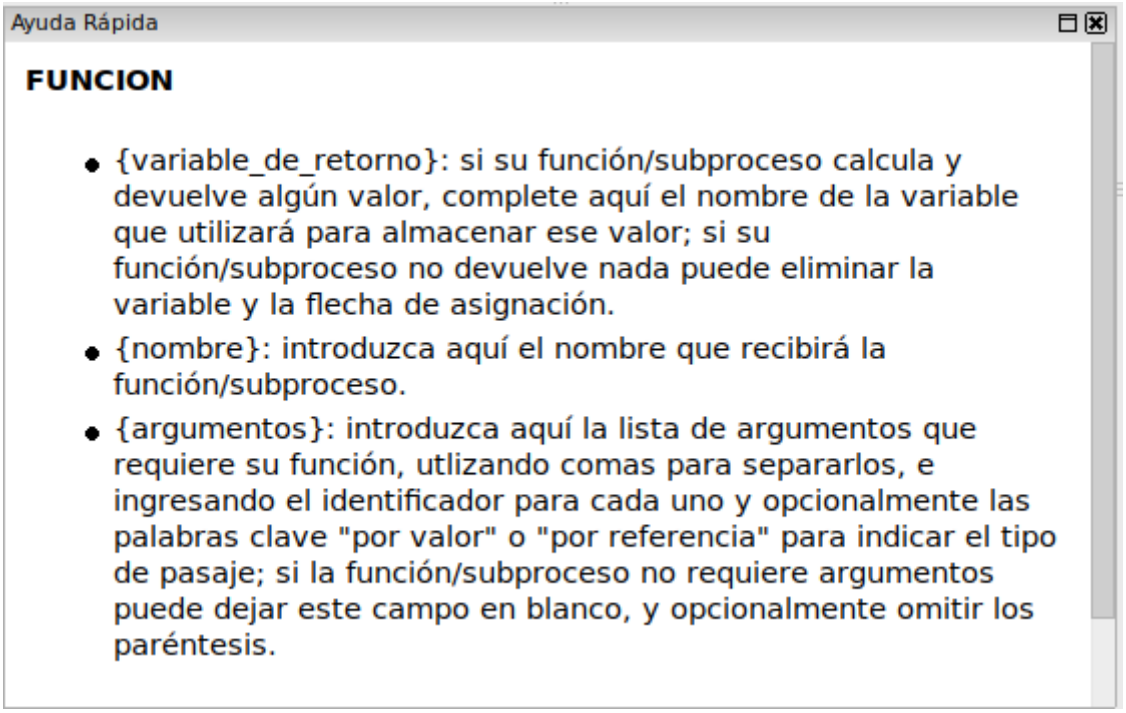
## Pasos 3, 4 y 5: Codificación, compilación y verificación del programa *Area_compuesta* con PSeInt.

En PSeInt para introducir funciones se definen al principio del código y utilizando el comando **Función**.

Copyright 2026




En la ventana de ayuda rápida nos explica cada uno de los elementos que aparecen en el código.




- **Variable de retorno**: sólo tiene sentido si la función devuelve un valor. Si no, se borra.
- **Nombre**: con el que posteriormente se invocará a la función en el algoritmo principal.

- **Argumentos**: o parámetros necesarios para la función.

Cambiamos pues el nombre del fichero y definimos dos funciones al inicio, una para calcular el área de un rectángulo y otra para calcular el área de un círculo. Ambas devuelven resultado y necesitan parámetros, la primera la base y la altura, y la segunda el radio. En PSeInt existe de forma predefinida la constante PI con el valor *3.1415926536* por lo que podemos usarla y no habrá que definirla.

area_compuesta_confunciones.psc

```
1  Funcion resultado ← calcularAreaCirc ( radio )
2      resultado←PI*radio*radio;
3  Fin Funcion
4  Funcion resultado ← calcularAreaRect( base,altura )
5      resultado←base*altura;
6  Fin Funcion
7
```

Una vez definidas las funciones, pasaremos a escribir el algoritmo principal. El argumento de las funciones será el valor introducido por el usuario y que corresponde con el lado del cuadrado.

```
Algoritmo Area_compuesta

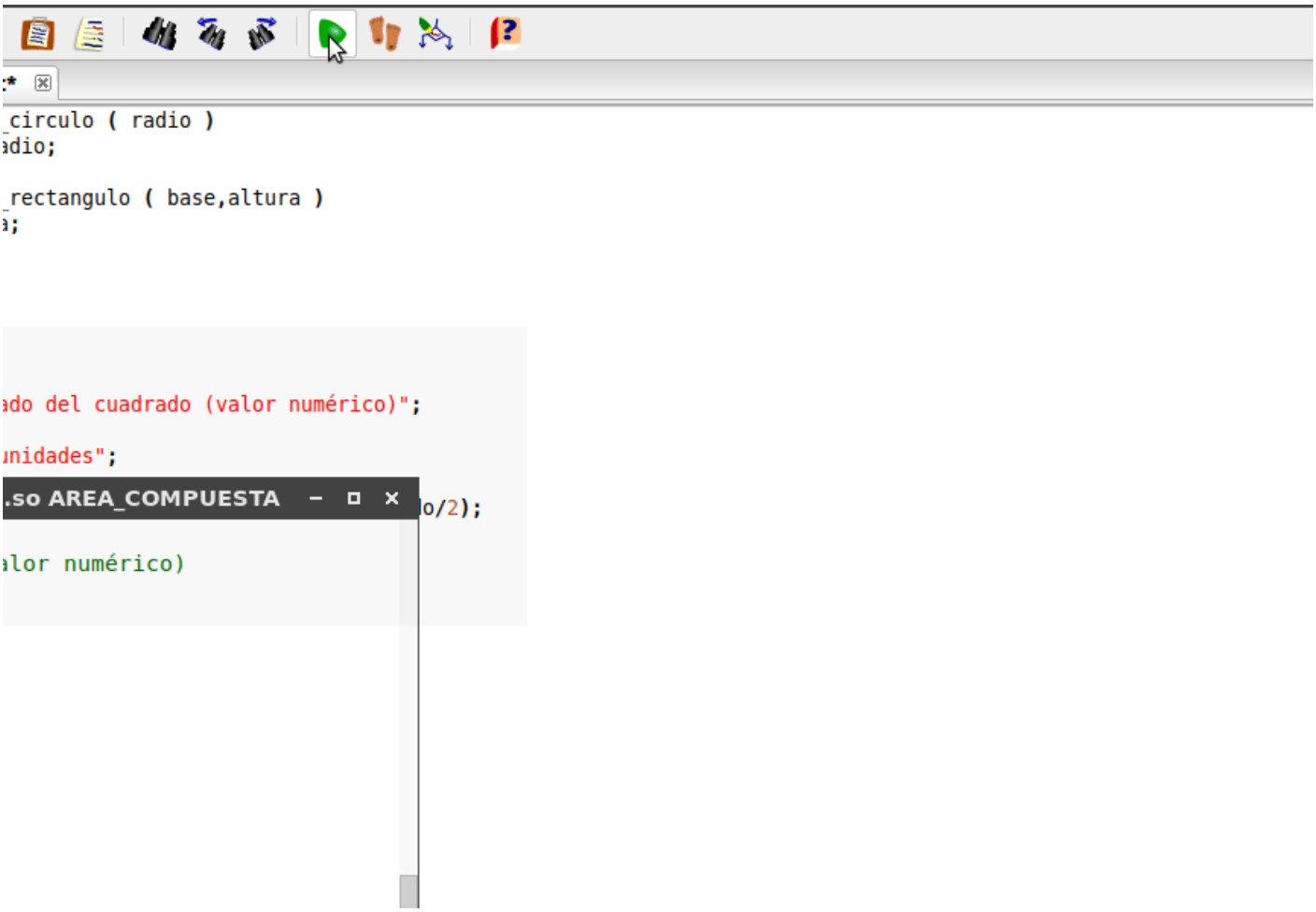
    Escribir "Indica el lado del cuadrado (valor numérico)";
    Leer lado;
    Escribir "Indica las unidades";
    Leer unidad;
    areaTotal←calcularAreaRect (lado,lado)+calcularAreaCirc(lado/2);
    Escribir "El resultado es ",areaTotal,unidad,"2";

FinAlgoritmo
```

Al ser funciones que devuelven un resultado hemos tenido que definir una variable areaTotal que almacene el resultado de lo obtenido por las funciones. De forma más correcta y generalizable tendríamos que haber predefinido previamente las variables lado y areaTotal como números reales, y unidad como string, si bien PSeInt nos permite ser un poco más laxos en el procedimiento. La última línea es la concatenación del valor obtenido, la unidad introducida y un 2 puesto que la unidad será elevada al cuadrado.

Solo nos queda ejecutar el programa y verificar su funcionamiento.

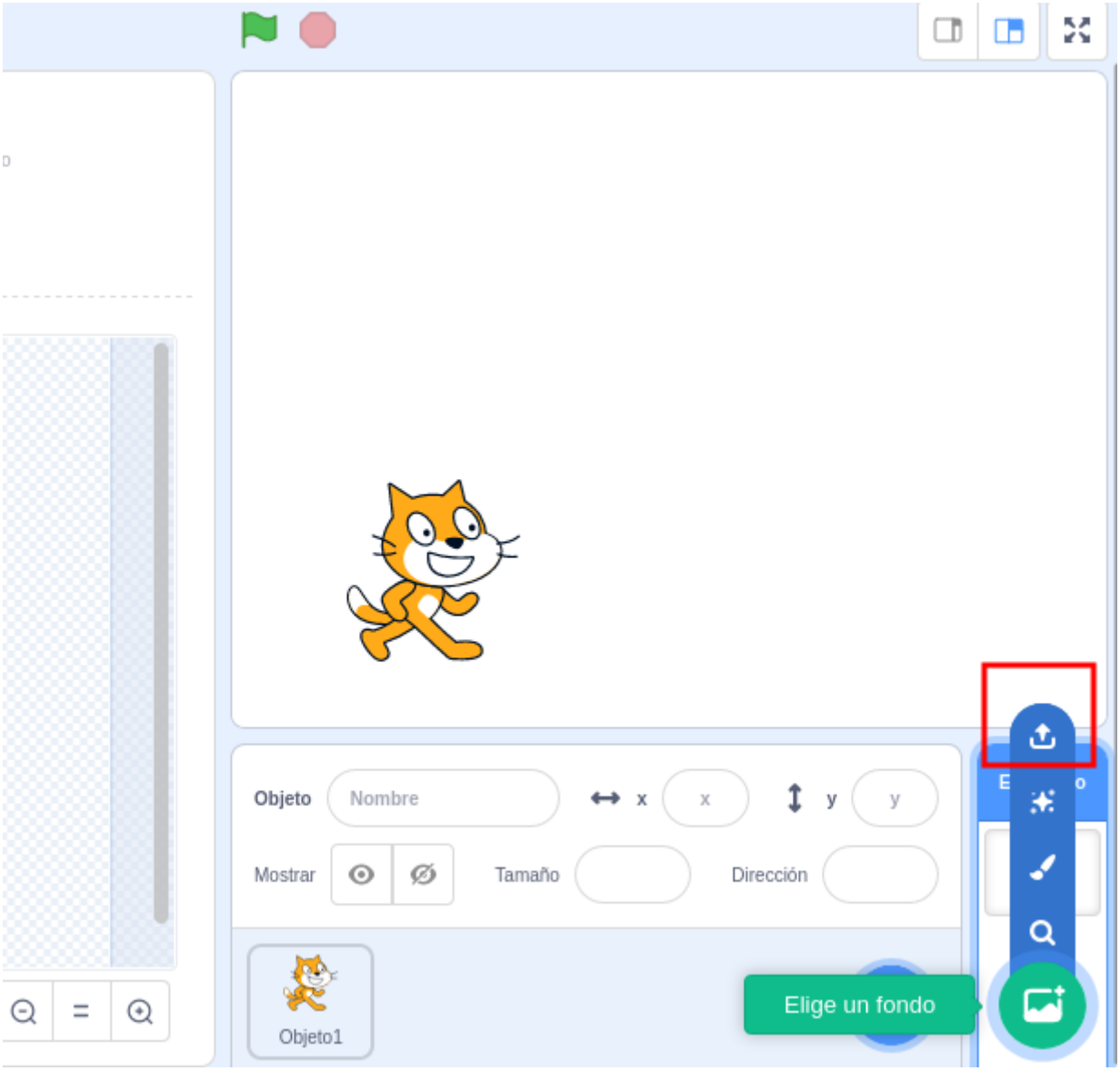
Copyright 2026




## Pasos 3, 4 y 5: Codificación, compilación y verificación del programa *Area_compuesta* con Scratch.

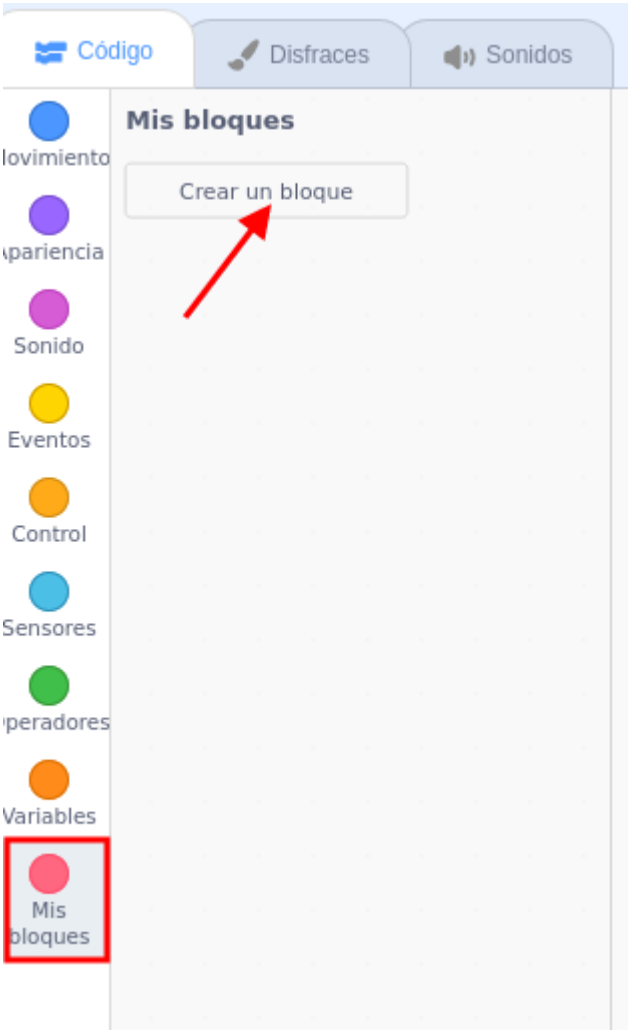
En Scratch no existe de forma explícita la opción de crear funciones que devuelven un resultado. Solo existe lo que hemos llamado procedimientos, eso sí con o sin parámetros. Eso se hace mediante la creación de nuevos bloques. Para crear funciones que devuelvan un resultado tendremos que crear dichos bloques y asignarles dentro del mismo el valor calculado a una variable predefinida. Veámoslo con el ejemplo.

En primer lugar cargaremos la imagen de la figura cuyo área queremos calcular como fondo del escenario. Para ello en la parte inferior derecha hacemos clic sobre el botón verde y escogemos la opción de cargar un escenario.




A continuación crearíamos las variables necesarias. Como en este programa las únicas variables necesarias son el lado del cuadrado y su unidad, y eso hemos visto que va a ser almacenado por defecto como respuesta del usuario, no es necesario. Tampoco vamos a necesitar crear una variable areaTotal para almacenar el resultado puesto que lo mostraremos directamente concatenando operadores y así nos ahorraremos un bloque.

A continuación crearemos las funciones calcularAreaRect(base, altura) y calcularAreaCirc (radio) Para ello Scratch nos permite crear nuevos bloques a los que les asociemos un código. Eso se hace desde **Mis Bloques** y pinchando en **Crear un bloque.**

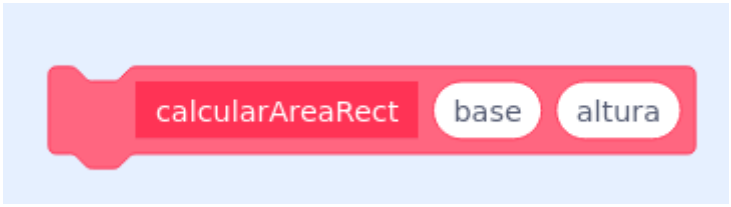


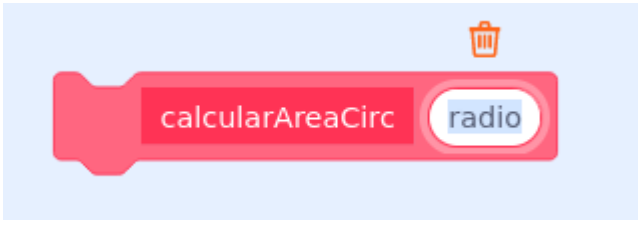

Al clicar sobre Crear un bloque se nos abre una ventana donde elegir el identificador de la función/bloque, así como opciones de añadirle argumentos o parámetros a esa función. Pueden ser texto, número, variable booleana o simplemente una etiqueta de texto.

Copyright 2026



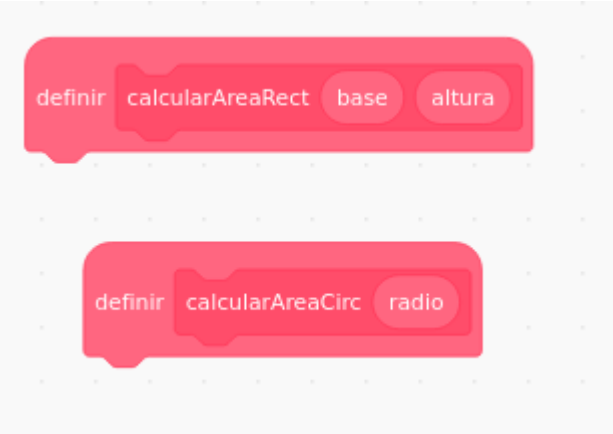

En nuestro caso crearemos dos nuevos bloques. El del área del rectángulo requiere dos entradas numéricas (base y altura), y el del área del círculo una única entrada numérica (radio).




Copyright 2026

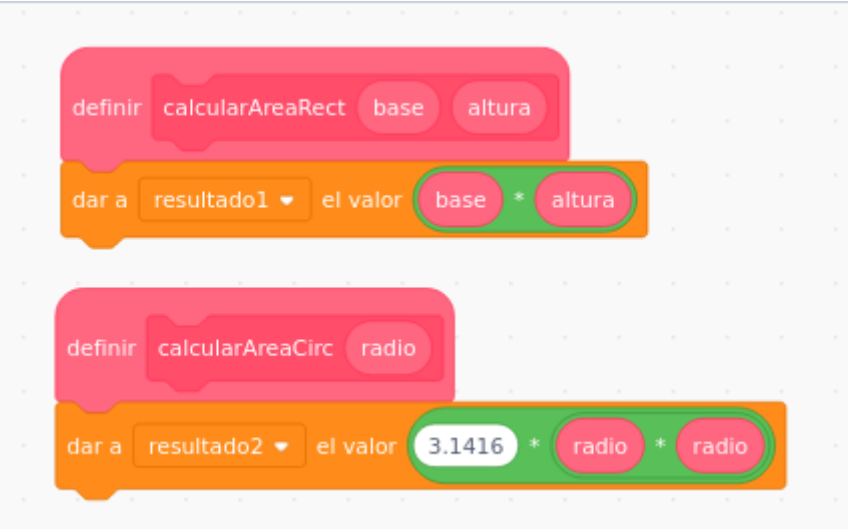
En ese momento en nuestra área de código aparecen dos nuevos elementos para que definamos bajo ellos los bloques que van a componer cada uno de estos subprocedimientos.




Como hemos dicho antes, son funciones que devuelven un resultado, por lo que primero habremos de crear variables para poder almacenar en ellas ese resultado. Les llamaremos resultRect y resultCirc.




Mediante operadores aritméticos en este caso, añadiremos los bloques correspondientes al cálculo del área de cada figura, dados sus parámetros.




Por último, una vez definidas las funciones/bloques, solo nos quedará el algoritmo principal, que tendrá este aspecto:




La última línea incluye la concatenación de texto con los resultados parciales de las funciones creadas. Solo quedaría comprobar su funcionamiento y depurar posibles errores.

Prúebalo aquí.

https://scratch.mit.edu/projects/750180283/embed

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

Copyright 2026




Revision #15
Created 2022-09-15 18:20:01 CEST by Ana López Floría
Updated 2024-01-11 13:07:03 CET by Mariola Palacio

Copyright 2026