2. Funciones. Declaración, invocación e implementación

Una función, como venimos comentando, es una parte del programa que se separa y se almacena de manera independiente del programa principal para que pueda reutilizarse sin tener que volver a escribir su código.

2.1. Comunicación entre los módulos de un programa

Ya sabemos que una aplicación puede dividirse en partes más pequeñas y sencillas que se comunican entre sí. De esa forma, evitamos duplicar código y clarificamos el funcionamiento del programa. Ahora nos interesa saber cómo se comunican entre sí las diferentes partes de una aplicación.

Veamos la siguiente figura:

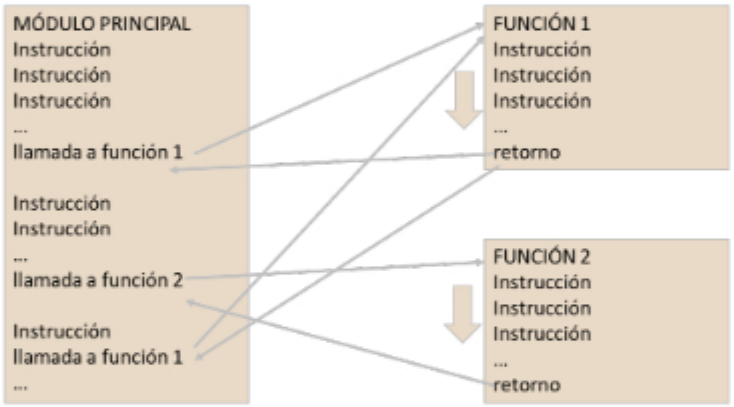

Figura 5.1. Llamadas entre bloques de código

Lo que está sucediendo en el esquema anterior es lo siguiente:

1) Empieza a ejecutarse el módulo principal.
2) Se realiza una llamada a la función 1. La ejecución del módulo principal queda en pausa.
3) Empiezan a ejecutarse las instrucciones de la función 1.
4) La función 1 devuelve el control (y, como veremos, puede que algún resultado) al módulo desde el que ha sido llamada.
5) La ejecución en el módulo principal sigue después del punto desde el que la función 1 fue llamada.
6) Se llama a la función 2. El módulo principal vuelve a quedar a la espera.
7) Se ejecutan las instrucciones de la función 2.
8) La función 2 devuelve el control (y puede que un resultado) al módulo desde el que ha sido llamada, reanudándose su ejecución.

9) Más adelante, el mismo módulo principal vuelve a llamar a la función 1 y se repite todo el proceso.

Trabajar con funciones evita, por ejemplo, que en la figura anterior el código de la función 1, que se llama dos veces desde el módulo principal, tenga que aparecer por duplicado dentro de dicho módulo.

Como hemos comentado, las funciones también pueden llamarse entre sí. El funcionamiento sería el mismo: la ejecución del bloque de código desde el que se llama a la otra función se detiene en el punto de la llamada, y cuando la función invocada termina de ejecutarse, se reanuda.

2.2. Partes de una función. Declaración e implementación.

Las funciones tienen, en general, dos partes:

- Una cabecera en la que se declara la función sin especificar qué hace ni cómo lo hace. Solo incluye el nombre, cuántos datos y de qué tipo se le pasan, y de qué tipo es el dato que devuelve. Estos datos que la función espera recibir se denominan **parámetros formales**.
- Un cuerpo en el que se definen las instrucciones que componen la función, y donde se determina el resultado que, en su caso, devolverá al módulo desde el que ha sido llamada.

La declaración, en algunos casos conocida también como prototipo, consiste en especificar la cabecera de la función simplemente a efectos de que, durante la compilación, el nombre de la función se reconozca como tal.

La implementación o definición de la función consiste en incluir el cuerpo de la misma con las instrucciones que debe ejecutar y el resultado que debe devolver.

IMPORTANTE

Estamos viendo el esquema general de una función, pero varias de esas partes no son obligatorias. Puede haber funciones que no devuelvan nada, y también funciones que no reciban ningún dato al ser llamadas. Lo que sí debemos tener en cuenta es que **una función solo puede devolver un dato**.

Por ejemplo, veamos la declaración de una función, en pseudocódigo, a la que le pasamos tres números y que nos devuelve el mayor de los tres:

```
entero mayor(entero numero1, entero numero2, entero numero3)
```

Aquí lo único que estamos especificando es que la función se llama «mayor», que se le pasarán tres números enteros cuando sea llamada, y que devolverá un número entero al módulo que la haya llamado. No entramos en detalles sobre qué hace ni cómo lo lleva a cabo.

Una vez completada la definición, el programa quedaría así:

```
entero mayor(entero numero1, entero numero2, entero numero3) {
      entero numeroMayor;
      si numero1 >= numero2 y numero1 >= numero3 entonces
            numeroMayor = numero1;
      sino si numero2 >= numero1 y numero2 >= numero3 entonces
            numeroMayor = numero2;
      sino
            numeroMayor = numero3;
      fin si
      devuelve numeroMayor;
}
```

Ahora ya queda más claro lo que hace nuestra función: recibe tres números enteros, calcula cuál es el mayor de los tres, lo guarda en una variable nueva y, finalmente, lo devuelve como resultado al módulo desde el que ha sido invocada.

2.3. Invocación

De lo visto anteriormente se deduce que, puesto que la función a la que se llama tiene la posibilidad de devolver un resultado, el módulo que llama a la función debe tener alguna forma de recibir dicho resultado.

Por ejemplo, si desde el módulo principal llamamos a la función mayor de esta forma:

```
mayor(6, 10, 8);
```

Le estamos pasando tres números (6, 10, 8), así que el programa calculará cuál es el mayor (10) y lo devolverá al módulo principal, aunque en este caso no lo estamos guardando ni usándolo de ninguna forma. Lo correcto sería hacer lo siguiente:

```
entero resultadoFuncion = mayor(6, 10, 8);
```

En este caso, el módulo principal llama a la función mayor pasándole los números 6, 10 y 8. A continuación, la función devuelve 10 como resultado y, después, en el módulo principal, ese resultado se guarda en la variable resultadoFuncion. Ahora podemos disponer de él de la manera más conveniente.

Otra opción, si queremos usar el resultado de la función solo una vez, sería:

```
escribir(mayor(6, 10, 8));
```

El resultado de la función se muestra por pantalla, pero no queda guardado en ningún sitio.

Los parámetros que se pasan a la función durante la invocación o llamada reciben el nombre de parámetros reales o actuales. Pasaremos a estudiar con más profundidad esta nomenclatura más adelante.

IMPORTANTE

Solo es posible invocar a funciones sin hacer referencia al resultado devuelto en los casos en los que la función no devuelve nada. Por ejemplo, una función `menu` que se limite a mostrar por pantalla las opciones de un menú podría ser llamada simplemente como `menu();`