1. Descomposición modular. Técnicas

La **programación estructurada** surge como un paradigma de mejora sobre la programación lineal tradicional. Se basa en el uso de estructuras básicas: la secuencia de instrucciones, las estructuras alternativas (if, switch) y las estructuras iterativas o bucles (for, while). De esta forma, se evita el uso de saltos directos dentro del programa (el famoso y olvidado **GOTO**) que hacen difícil seguir la pista a la ejecución del programa. También se intenta evitar que el mismo código se repita muchas veces: un bloque de instrucciones que debe ejecutarse tres veces puede introducirse dentro de un bucle que se ejecute tres veces; de esta forma, el bloque solo aparecerá una vez.

IMPORTANTE

GOTO era una instrucción utilizada en los primeros lenguajes de programación que permitía saltar directamente a una línea del programa, sabiendo su número. Se trata de un vestigio del primigenio lenguaje ensamblador y de sus saltos directos. Hoy está totalmente obsoleta.

Sin embargo, hay otras situaciones en las que querríamos reutilizar un bloque de código, pero no podemos incluirlo en un bucle. De hecho, puede que necesitemos utilizarlo en otro fichero o incluso en otra aplicación. Ahí es donde aparece la **programación modular**.

La programación modular se basa en dividir el código fuente de una aplicación en distintos módulos (también llamados subrutinas o subprogramas) reutilizables, lo que facilita la legibilidad y el mantenimiento posterior del código.

La idea principal tras la programación modular es que un problema complejo se divide así en problemas más simples mediante un proceso que puede continuar hasta que los problemas resultantes se resuelvan también en unas cuantas instrucciones básicas. Está técnica recibe varios nombres: «refinamiento sucesivo», «divide y vencerás» o «análisis descendente» (*Top-Down*). ¡Empezamos la casa por el tejado!

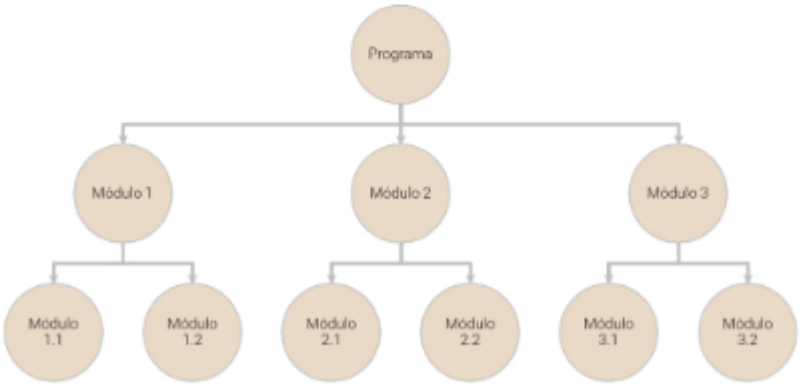

Figura 5.1. Descomposición modular

1.1. Ventajas de la programación modular

La programación estructurada presenta una serie de ventajas sobre la programación estrictamente lineal y con saltos directos (GOTO):

- Los programas son más fáciles de entender y su estructura es más clara.
- Se facilitan las pruebas y el mantenimiento de la aplicación.

Al usar técnicas de descomposición modular, obtenemos otras ventajas añadidas a las ya mencionadas:

- Facilita la colaboración entre el personal de desarrollo de software.
- Permite la reutilización de código de una manera sencilla y efectiva.
- Facilita el mantenimiento. Al evitar la repetición indiscriminada de código, si se descubre un error o si se quiere mejorar una parte del programa, solo habrá que efectuar los cambios en el módulo correspondiente.

1.2. Técnicas de programación modular

Como hemos comentado, la programación modular consiste en descomponer la aplicación en módulos independientes pero conectados entre sí, de forma que cada módulo contenga un bloque de código que no se repita en el resto pero que pueda ser accesible desde cualquier lugar en cualquier momento.

La descomposición modular no puede conseguirse de cualquier modo. Hay que intentar buscar un equilibrio en la cantidad, complejidad y extensión de los módulos. Tan problemático puede ser dividir una aplicación de mil líneas en 100 módulos de 10 líneas, como hacerlo en 2 bloques de 500. Al final, lo mejor es basarse en conceptos como la independencia funcional, que incluye a su vez otros factores como la cohesión y el acoplamiento:

Cohesión	Acoplamiento
Todas las partes de un módulo deben contribuir a realizar una única función.	Cada módulo debe ser independiente y relacionarse lo menos posible con el resto, salvo para llamarse unos a otros y repartirse las tareas.

Tabla 5.1. Cohesión y acoplamiento

El módulo ideal es independiente del resto, realiza una única tarea y solo se comunica con el resto para realizar la tarea y devolver los resultados al módulo desde el que ha sido llamado. Es decir, se busca la máxima cohesión (modularidad) y el mínimo acoplamiento (interdependencia).

En la práctica, la descomposición modular empieza dividiendo el programa principal en módulos más o menos equilibrados en cantidad y tamaño. Posteriormente se sigue dividiendo cada módulo en otros más pequeños hasta que cada uno de ellos resuelve un problema concreto en una cantidad manejable de líneas de código. De hecho, suele decirse que si el contenido del módulo se puede

representar mediante un algoritmo sencillo que no ocupe más de una página, no tiene sentido seguir descomponiendo el módulo. Es lo que antes te hemos presentado como análisis descendente o *top-down*.

Con la programación orientada a objetos, la descomposición modular se lleva hasta el extremo, puesto que los módulos se convierten en métodos asociados a clases diferenciadas, salvo aquellos módulos que realizan tareas transversales que pueden ser comunes a todas las clases (e incluso a diferentes aplicaciones). Incluso así, esos módulos pueden agruparse en clases de utilidades. Profundizaremos en este concepto, llamado **descomposición orientada a objetos**, en siguientes unidades.

Un ejemplo práctico podría ser una aplicación para gestionar una agenda de contactos. La alternativa a realizar todo el programa en un solo bloque de código sería utilizar la programación modular. De esa forma, empezaríamos a descomponer el programa en módulos hasta que cada uno de ellos realizara solo una tarea concreta. El resultado podría ser similar a este:

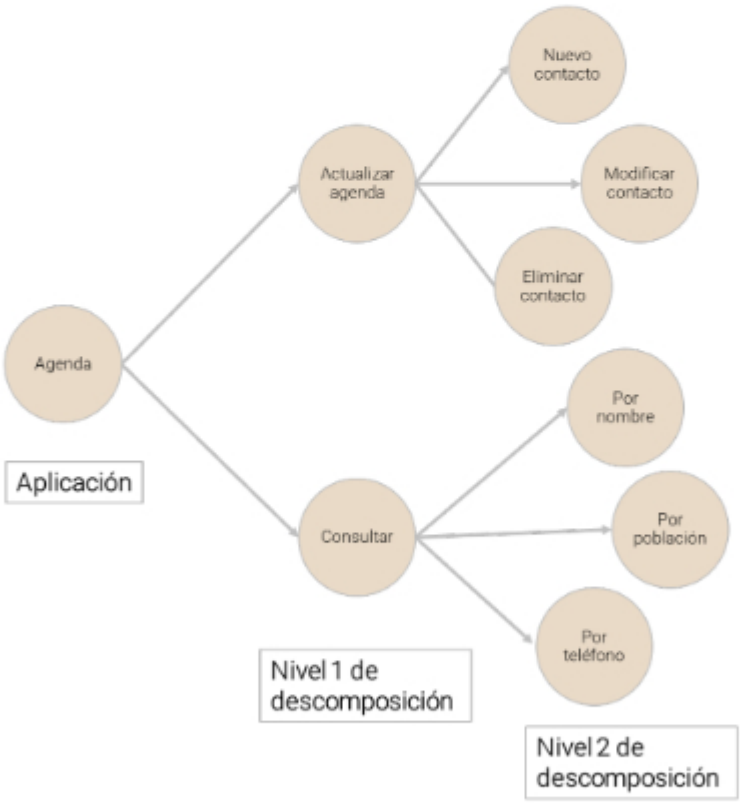

Figura 5.2. Descomposición modular de una aplicación sencilla