Introducción al diseño modular

El diseño modular es una generalización del diseño descendente de programas. Se trata de dividir el problema original en varios subproblemas más pequeños, cada uno de ellos con una misión bien determinada dentro del marco general del problema, que interaccionan clara y mínimamente de tal forma que todos ellos juntos solucionan el problema inicial. Si algunos subproblemas siguen siendo demasiado complicados, se les aplica el mismo proceso y, así sucesivamente hasta llegar al estado en que todos los subproblemas son lo bastante sencillos como para detener el proceso.

El resultado es una estructura jerárquica que refleja las descomposiciones efectuadas: cada descomposición es el resultado de abstraer las características más relevantes del problema que se está tratando de los detalles irrelevantes en el nivel de razonamiento actual, los cuales adquieren importancia en descomposiciones sucesivas.

Desde el punto de vista de su gestión, cada subproblema es un TAD que se encapsula en lo que se denomina **módulo**.

Un **módulo** es una unidad de programa que agrupa o encapsula un conjunto de definiciones de objetos (constantes, tipos de datos, variables, acciones, funciones, ...) permitiendo ocultar todos los detalles internos relativos a su representación o ejecución. Además, cada módulo puede ser compilado de forma diferente e independiente.

Programando con TADs no nos basamos en las acciones para la descomposición modular, sino en los objetos sobre los que se actúa.

Debemos dividir la solución al problema en módulos y la forma de encontrar los módulos se basa en el uso de TADs. De esta forma un **programa es una colección estructurada de implementaciones de TADs**.

La declaración de un módulo consta de:

- **Encabezamiento** – nombre.
- **Parte pública** – declaración de objetos visibles desde el exterior – interfaz.
- **Parte privada** – descripción interna de los objetos anteriores que está oculta al exterior – implementación.

La mejora respecto al diseño descendente proviene de la ocultación de la representación de los datos (privacidad de la representación) y de la limitación de su manipulación al ámbito que define el tipo (protección).

Algunas propiedades de la programación modular:

- **Abstracción**. Los usuarios de un TAD no necesitan conocer detalles de implementación.
- **Corrección**. Un TAD sirve como unidad en las pruebas de programas (análogo a las funciones en el diseño descendente). La utilización de una especificación formal permite la verificación formal de la aplicación y la demostración de la corrección de un programa.

- **Eficiencia**. La implementación de un tipo se retrasa hasta conocer las restricciones de eficiencia sobre sus operaciones y así se pueden elegir los algoritmos óptimos. Sin embargo, la inaccesibilidad de la implementación fuera de los módulos de definición de los TADs comporta a menudo problemas de eficiencia.
- **Legibilidad**.
    - La especificación de un TAD es suficiente para entender su significado.
    - Los programas que usan TADs son más fáciles de leer porque no usan estructuras de datos sino que llaman a las operaciones definidas del tipo.
- **Modificabilidad y mantenimiento**. Las modificaciones posteriores no suelen requerir analizar el problema completo, sino solo algunas partes si afectan a las demás.
- **Organización**. La visión de una aplicación como un conjunto de TADs con significado.
- **Reusabilidad**. Los TAD diseñados en una especificación pueden ser reutilizados en otros contextos con mínimos cambios.
- **Seguridad**. La imposibilidad de manipular directamente la representación evita el mal uso de los objetos del tipo y, en particular, la generación de valores incorrectos.