Implementación modular de TAD

1º) Definir lo que aparecerá en la parte pública o ***interfaz*** del módulo, basándonos en la especificación del TAD:

- identificadores válidos
- perfiles o cabeceras de cada operación (con su especificación):
  - parámetros de entrada, salida o entrada-salida
  - comunicación de situaciones de error (operaciones parciales…)

→ *Este 1er paso es requisito previo para poder distribuir el trabajo de programación en el equipo*

Implementación modular de TAD

2º) Decidir cómo representar los valores del tipo en base a tipos básicos predefinidos, constructores básicos de tipos (como vectores y registros), y/u otros TAD definidos previamente

→ ***representación interna*** concreta del nuevo TAD

- Deberá permitir implementar las operaciones definidas para el tipo, de forma eficiente tanto en coste en memoria como en tiempo
- La representación interna deberá permanecer oculta
  - El uso del nuevo tipo solo será posible mediante las operaciones definidas en la interfaz del tipo (***encapsulación*** = *privacidad* + *protección*)

Implementación modular de TAD

3º) Implementar cada operación de la interfaz del TAD y las operaciones auxiliares que resulten de interés/utilidad, de acuerdo a la representación interna definida

- las operaciones de la interfaz serán accesibles para los usuarios del nuevo tipo
- el resto de las operaciones sólo serán utilizables en la implementación del TAD

Implementación modular de TAD

Guía para llegar desde la especificación hasta una implementación en pseudocódigo u otro lenguaje de programación modular

- Las operaciones 0-arias (constantes)
  - Se implementarán como constantes, o como procedimientos o funciones sin parámetros
- Las demás operaciones se implementan como procedimientos o funciones
  - Varias operaciones con el mismo dominio y distinto rango podrán implementarse como un procedimiento que devuelva varios resultados
    - Especialmente si se van a utilizar a menudo de forma conjunta
    - Importante si de esa forma el coste de obtener los resultados se reduce
- Las operaciones con casos o situaciones de error (e.g. las parciales):
  - Mecanismos de protección frente a errores dependientes del lenguaje de programación a utilizar
    - Manejo de excepciones → identificar o definir excepciones a utilizar
    - Parámetros de salida de error en cada operación

Operaciones: Procedimientos o funciones

- Procedimientos y funciones encapsulan un bloque de acciones (código) reutilizable, y simplifican la lectura de los programas que escribimos

Procedimientos:

*{Un **procedimiento** es una **acción o instrucción virtual**, que una vez definido se puede utilizar como otra instrucción mas.*

Un procedimiento puede definirse con 0 ó mas parámetros, cada uno de ellos podrá ser de:

- Entrada (ent): dato que el procedimiento recibe para ser utilizado en sus acciones

- Salida (sal): dato que el procedimiento comunica como resultado de sus acciones

- Entrada y Salida (e/s): dato que el procedimiento recibe para ser utilizado en sus acciones y cuyo valor actualizado comunica como resultado de sus acciones. }

procedimiento <nombre>(**ent** <parámetros_1>:<tipo_1>;
sal <parámetros_2>:<tipo_2>;
e/s <parámetros_3>:<tipo_3> ...)

<*declaraciones locales* de constantes, tipos de datos, variables, proced., funciones...>

principio

<secuencia de acciones>

fin

Uso:

contador:=0;

nombre(2.5, “A”, valorMedio, contador);

escribir(valorMedio); escribir(contador); …

Operaciones: Procedimientos o funciones

Funciones:

*{ Una **función** es un **valor virtual**, es decir, se puede utilizar dentro de una expresión y el resultado es un valor. Puede tener 0 ó más parámetros, todos son de entrada, y sólo puede devolver un resultado.}*

función <nombre>(<parám_1>:<tipo_1>; <parám_2>:<tipo_2> ...) **devuelve** <tipo_fun>

<*declaraciones locales* de constantes, tipos, variables, proced., funciones...>

principio

<secuencia de acciones>

devuelve <valor_de_tipo_fun>

{tras devolver el valor, la función termina}

fin

Uso:

cálculo:= 23.5 + nombre(2.7, x, y)

- *Al implementar una operación de un TAD…*
  - *si sólo hay que devolver **un dato resultado**, podrá optarse por procedimiento o por función*
  - *sino, (devolver **0, 2 o más resultados**) tendrá que optarse necesariamente por un procedimiento*

Implementación modular de TAD

- Al implementar la operación se puede decidir que:

a) el resultado sea la actualización de uno de los parámetros del dominio
   - parámetro del dominio y resultado = un único parámetro (**ent/sal**)
   - sólo posible con procedimientos (las funciones sólo tienen parámetros de entrada)
   - se evita ocupar nueva memoria para los datos resultado y el tiempo de copiar todo lo que no resulta modificado (más **eficiente**, en tiempo y memoria)
   - combinando su uso con el de una operación *copiar* (o *duplicar)*, siempre se podrán generar nuevas copias separadas de los datos (valores previo y posterior a la modificación)

b) el resultado sea una copia distinta en memoria del parámetro del dominio:
   - el parámetro del dominio será de entrada y el resultado será de salida (o el valor devuelto por una función)
   - se ocupa memoria adicional, independiente, para valor y resultado
   - combinando su uso con el de una operación *copiar* (o *duplicar)*, siempre se podrá hacer que el nuevo valor sustituya al original (en memoria)