Modularización

Uso de subprogramas

- Razones válidas para crear un subprograma
- Pasos para escribir un subprograma
- Acerca del nombre de un subprograma

Métodos

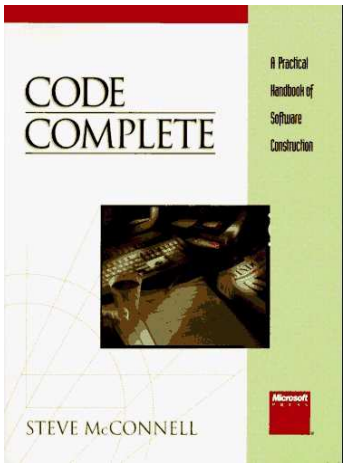
- Definición de los métodos: cabecera, cuerpo y signatura
- Uso de los métodos
- Paso de parámetros
- Devolución de resultados (sentencia `return`)
- Constructores (la palabra reservada `this`)
- Métodos estáticos

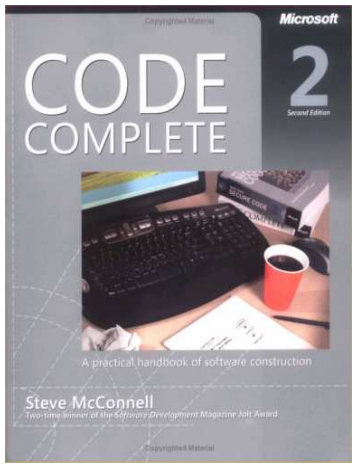
Ámbito de las variables

Cohesión y acoplamiento

Bibliografía

Steve McConnell: *"Code Complete"*.
Microsoft Press, 2004 [2ª edición] ISBN 0735619670.
Microsoft Press, 1994 [1ª edición] ISBN 1556154844.





Uso de subprogramas

Los lenguajes de programación permiten descomponer un programa complejo en distintos subprogramas:

- **Funciones y procedimientos**
  en lenguajes de programación estructurada

- **Métodos**
  en lenguajes de programación orientada a objetos

Razones válidas para crear un subprograma

ü Reducir la complejidad del programa (“divide y vencerás”).

ü Eliminar código duplicado.

ü Mejorar la legibilidad del código.

ü Limitar los efectos de los cambios (aislar aspectos concretos).

ü Ocultar detalles de implementación
(*ocultación de información*)
p.ej. algoritmos complejos

ü Promover la reutilización de código
p.ej. componentes reutilizables y familias de productos

ü Facilitar la adaptación del código a nuevas necesidades
p.ej. interfaces de usuario

ü Mejorar la portabilidad del código.

Pasos para escribir un subprograma

1. Definir el problema que el subprograma ha de resolver.

2. Darle un nombre no ambiguo al subprograma.

3. Decidir cómo se puede probar el funcionamiento del subprograma (de esta forma, desde el comienzo se piensa en cómo se utilizará el subprograma, lo que tiende a mejorar el diseño de un interfaz adecuado para el subprograma).

4. Escribir la declaración del subprograma:
   - La cabecera de la función en lenguajes estructurados.
   - La cabecera del método en lenguajes orientados a objetos.

5. Buscar el algoritmo más adecuado para resolver el problema.

6. Escribir los pasos principales del algoritmo como comentarios en el texto del programa.

7. Rellenar el código correspondiente a cada comentario.

8. Revisar mentalmente cada fragmento de código.

9. Repetir los pasos anteriores hasta quedar completamente satisfecho.

El nombre de un subprograma

Al crear un subprograma hemos de darle un nombre:

- Cuando el subprograma no devuelve ningún valor (procedimientos y métodos `void`):

  El nombre del subprograma suele estar formado por un verbo seguido, opcionalmente, del nombre de un objeto.

  Ejemplos:
  ```
  ingresar
  realizarTransferencia
  abonarImpuestos
  …
  ```

  El subprograma se encargará de realizar una operación independiente con respecto al resto del programa.

- Cuando el subprograma devuelve un valor (funciones y métodos):

  El nombre del subprograma suele ser una descripción del valor devuelto por la función o el método.

  Ejemplos:
  ```
  saldoActual
  saldoMedio
  …
  ```

  El subprograma se usará habitualmente para obtener un valor que emplearemos dentro de una expresión.

Observaciones

ü El nombre debe describir todo lo que hace el subprograma.

ü Se deben evitar nombres genéricos que no dicen nada (vg. `calcular`)

ü Se debe ser consistente en el uso de convenciones.