Tema 1. Introducción a las estructuras y tipos de datos

`http://aulavirtual.uji.es`

José M. Badía, Begoña Martínez, Antonio Morales y José M. Sanchiz

{badia,bmartine,morales,sanchiz}@icc.uji.es

Estructuras de datos y de la información

Universitat Jaume I

Índice

Bibliografía

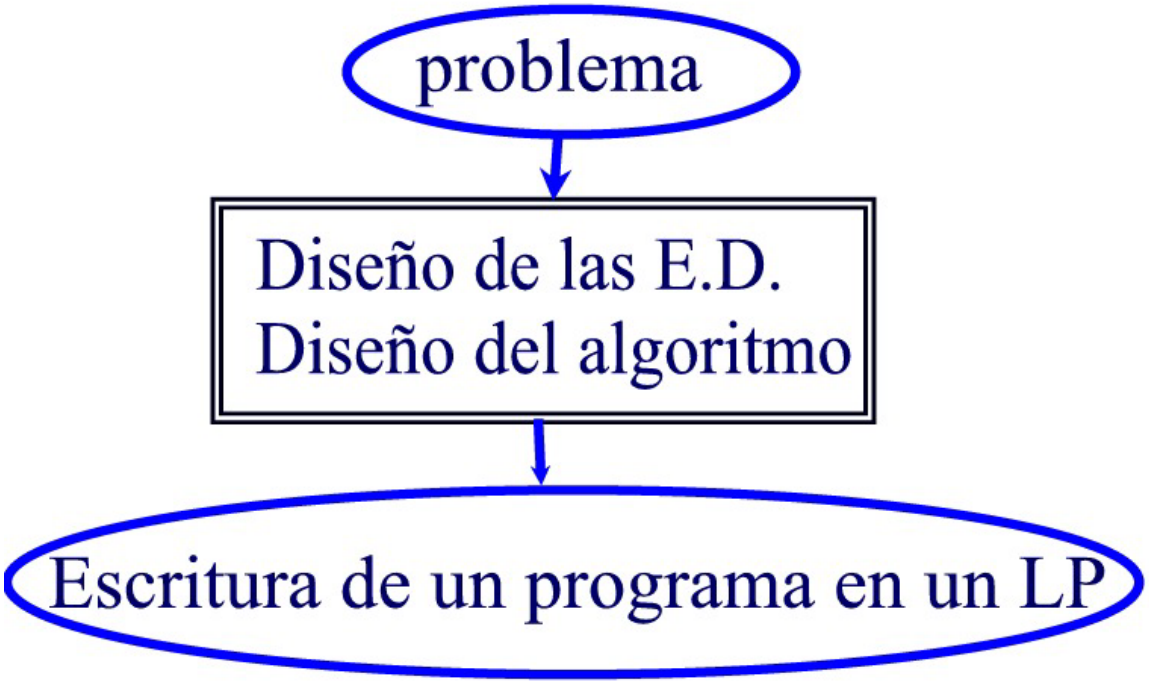
- *Diseño de Programas. Formalismo y Abstracción*, R. Peña Marí, Prentice Hall, 1998, Capítulo 5.
- (Joyanes y Zahonero '98), Capítulos 1 y 3
- *Fundamentos de Estructuras de Datos*, Z.N. Hernández et al., Thomson, 2005, Capítulo 1.

Objetivos

- Conocer los conceptos de abstracción y ocultación de la información para el desarrollo de software de calidad.
- Asimilar el papel que juega la abstracción de datos en el diseño de programas modulares y reutilizables.
- Asimilar el concepto de Tipo Abstracto de Datos (TAD) como un conjunto de valores y operaciones asociadas.
- Distinguir entre especificación e implementación de un TAD.
- Conocer la relación entre los TADs y las clases en C++

1 Introducción

Informática : Tratamiento automático de la información mediante un computador.



1 Introducción (II)

Objetivo de las *Estructuras de Datos*:

Estudio de la representación de la información (*datos*) que permitan obtener un algoritmo lo más sencillo posible.

- Identificar y desarrollar de modo formal las entidades y las operaciones que definen la información que se trata, así como determinar qué tipos de problemas pueden resolverse utilizando estas operaciones y entidades.
- Determinar la representación más idónea de estas estructuras abstractas e implementarlas.

Ejemplos:

- Buscar un libro en la biblioteca
- Uso de números romanos o arábigos

2 Abstracción de datos

Abstracción:

- Simplifica la comprensión de fenómenos complejos.
- Reconocer aspectos importantes e ignorar los detalles que no tengan relación directa con la solución.

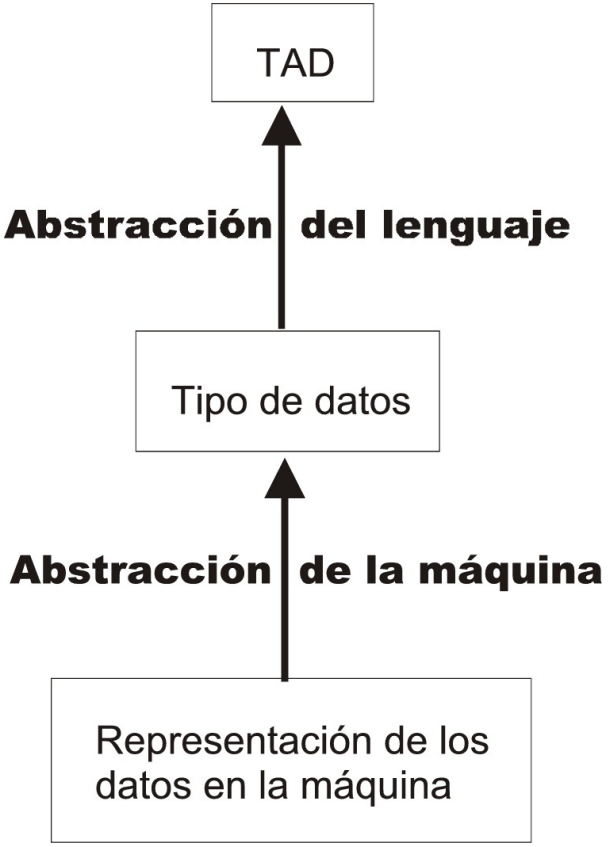
Ventajas:

- Solución de problemas complejos
- Proporciona una solución más general

3 Tipos de datos y ocultación

Abstracción de datos: manipulación de los datos a nivel lógico y abstracción de su implementación física en una determinada máquina o en un determinado lenguaje de programación

Los lenguajes de programación suponen el primer nivel de abstracción al proporcionar los **tipos de datos**.

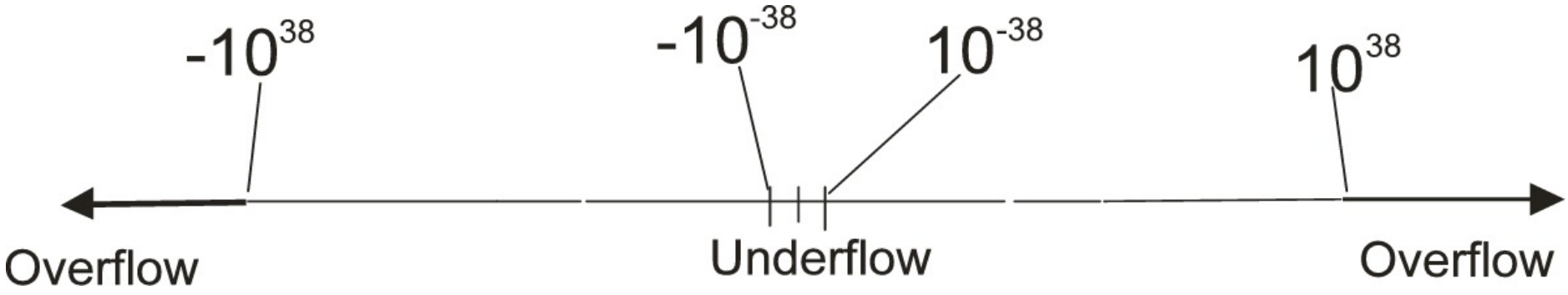


3 Tipos de datos y ocultación (II)

Tipos de datos: conjunto de valores posibles junto con las operaciones definidas para su manipulación.

Ejemplo: Calcular el producto escalar de 2 vectores de números reales.

- Tipo de datos: **float** en C++
  - Valores:



  - Operaciones: +, -, *, /, >, <, ==, ...

3 Tipos de datos y ocultación (III)

v[i]*v[j] en ensamblador del MIPS R2000

```
#$3 <-- i; $4 <-- j
#$5 <- dirección inicial de v
sll  $6, $3, 2   #*4
add  $6, $6, $5  #dir. elem v[i]
lw   $7, 0($6)   #$7=v[i]
sll  $6, $4, 2
add  $6, $6, $5  #dir. elem v[j]
lw   $8, 0($6)   #$8=v[j]
mult $7, $8      #hi lo
mfhi $7          #$7
mflo $8
```

Producto escalar de dos vectores

```
float ProdEscalar(float v[], float w[], int N) {
  int i;
  float x;
  for (i=0; i<N; i++)
    x= x + v[i]* w[i];
  return x;
}
```

3 Tipos de datos y ocultación (V)

Los lenguajes de programación (LP) proporcionan herramientas para manejar distintos tipos de datos.

- Tipos predefinidos
  - proporcionados por el LP
  - sólo nos preocupamos de su uso
- Tipos definidos por el usuario
  - elegir y definir una estructura de datos de las proporcionadas por el LP para su representación
  - elegir y definir operaciones de manipulación
  - garantizar el correcto comportamiento del tipo

3 Tipos de datos y ocultación (VI)

Ocultación: Esconder los detalles de implementación de un determinado objeto - *tipo de dato u operación* - y mostrar sólo su especificación.

Qué $\Longleftrightarrow$ Cómo

Ejemplo: tipo **float** (real)

- *Qué*: definir variables de tipo *float*, acceder a las variables, modificar su contenido, operar con las variables
- *Cómo*: codificación de los números reales con precisión simple en la máquina

3 Tipos de datos y ocultación (VII)

La técnica que se utiliza para conseguir la ***ocultación*** es el ***encapsulamiento***.

Encapsulamiento: Aislar las cuestiones de implementación de un objeto de las de su especificación.

- Operaciones, tareas $\Longrightarrow$ Funciones en C++
- Tipos abstractos de datos $\Longrightarrow$ Clases en C++

3 Tipos de datos y ocultación (VIII)

Ventajas del uso de la abstracción y la ocultación en el desarrollo de software:

- Simplificación del código resultante. Más fácil de entender.
- Menos errores.
- Más fácil de mantener y depurar.
- Posibilidad de realizar cambios sin afectar a todo el código.
- Reusabilidad de tipos de datos y funciones.

4 Concepto de TAD

Un **Tipo Abstracto de Datos** (TAD) es una entidad que consta de:

- Una colección de datos que definen los valores válidos para el tipo.
- Un conjunto de operaciones de manipulación de los datos.
- Una especificación del tipo independiente de cualquier implementación.
  Dicha especificación consiste en:
  - el concepto lógico del tipo
  - una serie de axiomas o propiedades que definen el comportamiento de las operaciones de manipulación

4 Concepto de TAD (II)

TAD Bolsa

Usa Bool

Operaciones

`Crearbolsa:` $\longrightarrow$ `bolsa`

Crea una bolsa vacía

`Poner: bolsa x elemento` $\longrightarrow$ `bolsa`

Añade un elemento a la bolsa

`DameUno: bolsa` $\longrightarrow$ `elemento`

Devuelve un elemento de la bolsa (no lo saca)

`Sacar: bolsa x elemento` $\longrightarrow$ `bolsa`

Retira de la bolsa una copia del elemento

`Esvacia: bolsa` $\longrightarrow$ `bool`

Dada una bolsa, dice si esta vacía

4 Concepto de TAD (III)

Axiomas $\forall$ b $\in$ bolsa, $\forall$ e, f $\in$ elemento

1) `DameUno(Crearbolsa)=`*error*

2) `DameUno(Poner(b,e))= e`

3) `Sacar(Crearbolsa, e)=` *error*

4) `Sacar(Poner (b, e), f)=` *Si* `e=f` *entonces* `b`
sino `Poner (Sacar(b, f), e)`

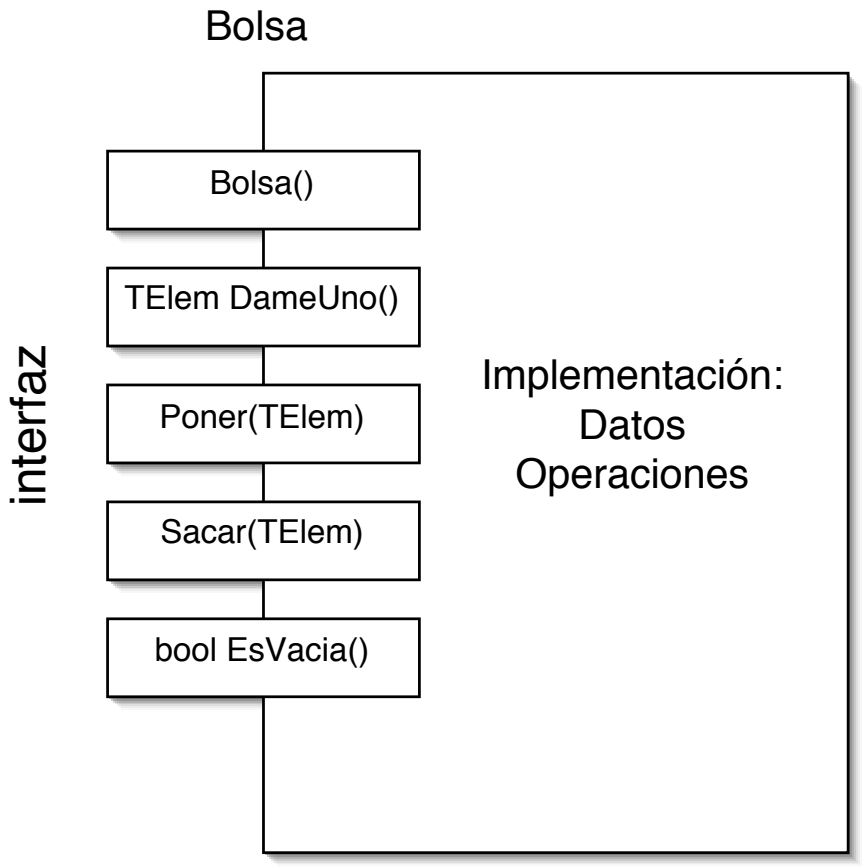
5) `Esvacia(Crearbolsa) = cierto`

6) `Esvacia(Poner (b, e)) = falso`

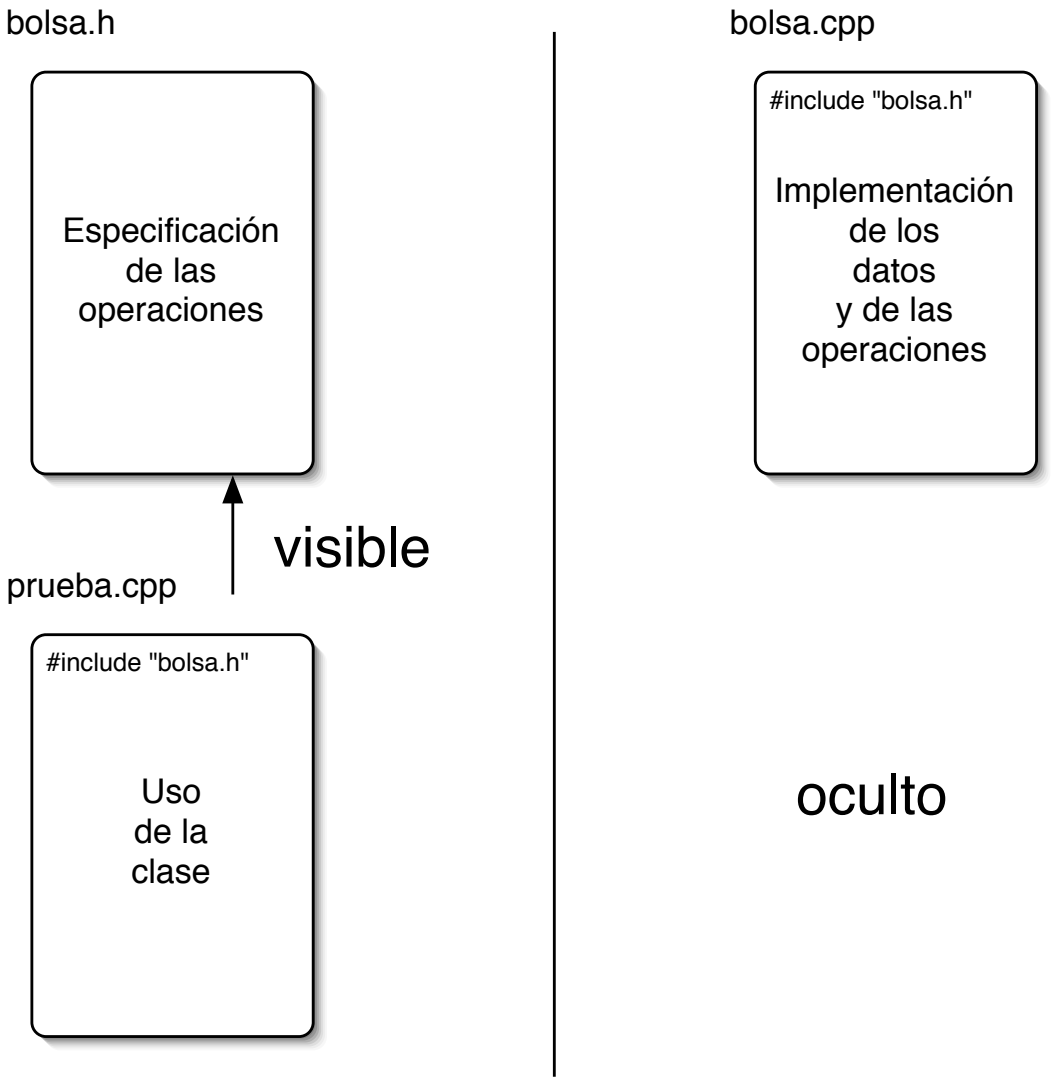
4 Concepto de TAD (IV)

- Un TAD es una colección de valores y un **conjunto de operaciones**, las únicas permitidas para manipular esos valores.
- Existen dos módulos totalmente independientes: la especificación del tipo y la implementación del tipo $\Longrightarrow$ encapsulamiento
- Los usuarios del TAD no conocen su implementación $\Longrightarrow$ ocultación
- Las operaciones definidas deben permitir generar cualquier valor del tipo.
- La especificación es independiente del lenguaje de programación.

TAD ≡ clase


Bolsa
Bolsa()
TElem DameUno()
Poner(TElem)
Sacar(TElem)
bool EsVacia()
interfaz
Implementación:
Datos
Operaciones

Encapsulamiento y ocultación



5 TADs en C++ (III)

Diferenciar 3 niveles en el trabajo con tipos/estructuras de datos:

- Uso
- Especificación
- Implementación

Ejemplo de uso de la tipo Bolsa:

- A partir de la especificación del TAD Bolsa, escribir programa que la vacíe

5 TADs en C++ (IV)

Ejemplo de uso de la clase. `prueba.cpp`

```
#include "bolsa.h"
int main() {
  Bolsa b;
  TElem e;
  ...
  while (!b.EsVacia()) {
    e = b.DameUno();
    b.Sacar(e);
  }
}
```