

Profesores:

Alfonso López Ruiz

Práctica 4

Programación de TAD genérico arborescente en C++

Estructuras de datos y algoritmos

Grado en Ingeniería Informática – Curso académico 2025-2026



OBJETIVOS DE LA PRÁCTICA

- ▶ Implementar un **TAD genérico en memoria dinámica con estructura arborea**.
 - ▶ Colección de elementos interdependientes.
 - ▶ **Árbol binario**.
- ▶ Implementar un **TAD no genérico**.
 - ▶ **Evento**.
- ▶ Comprobar de manera automática el funcionamiento del TAD genérico.
 - ▶ **Leer instrucciones y datos** de fichero.
 - ▶ **Construir un fichero de salida** con los resultados.

TAD no genérico



#include

practica4.cpp

#include

TAD Árbol de búsqueda

arbol.hpp

TAD genérico

ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN INTERDEPENDIENTES*

espec **coleccionesInterdependientes**

usa **booleanos, naturales**

parámetro **formal**

género **ident, val**

operación

{Se requiere que el género **ident** tenga definidas las siguientes operaciones.

Las operaciones **igual** y **anterior** definen una relación de orden total (i.e., permiten organizar los datos de la colección en forma de secuencia ordenada)}

igual: ident s1, ident s2 -> booleano

{devuelve verdad si y solo si s1 es igual que s2.}

anterior: ident s1, ident s2 -> booleano

{devuelve verdad si y solo si s1 es anterior a s2.}

fpf

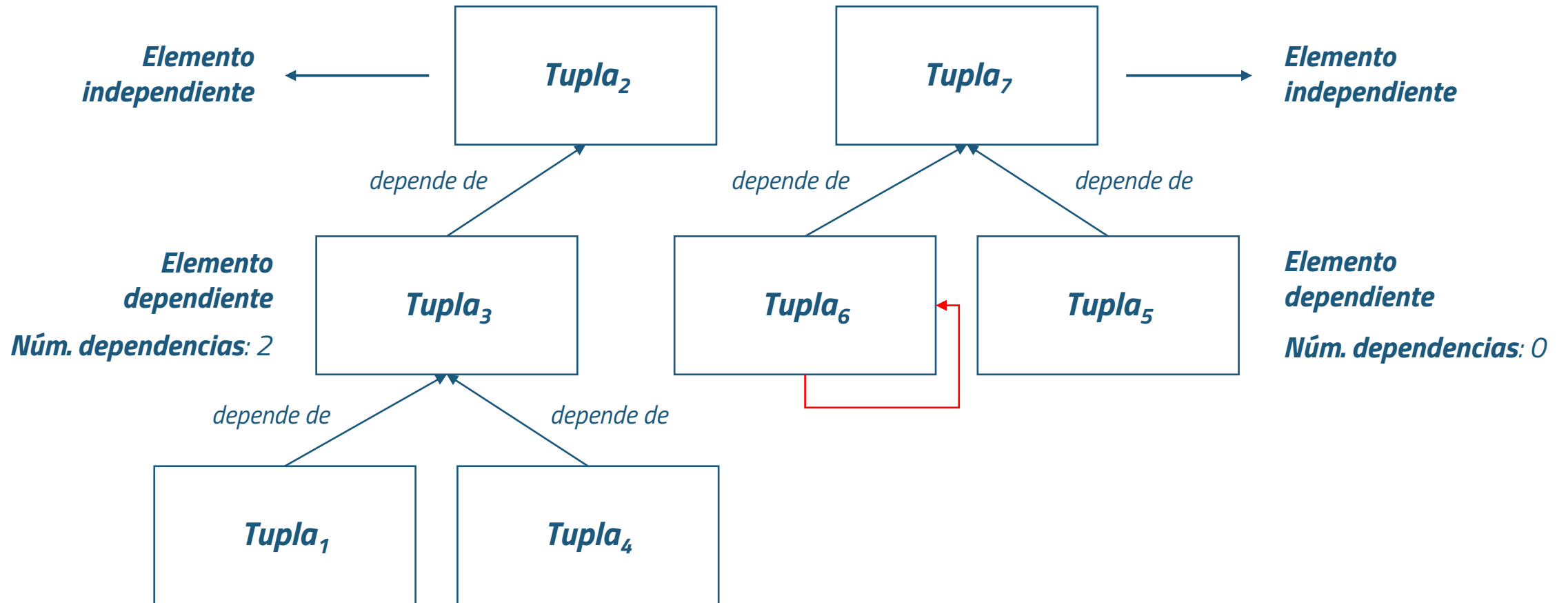
Nodo₁

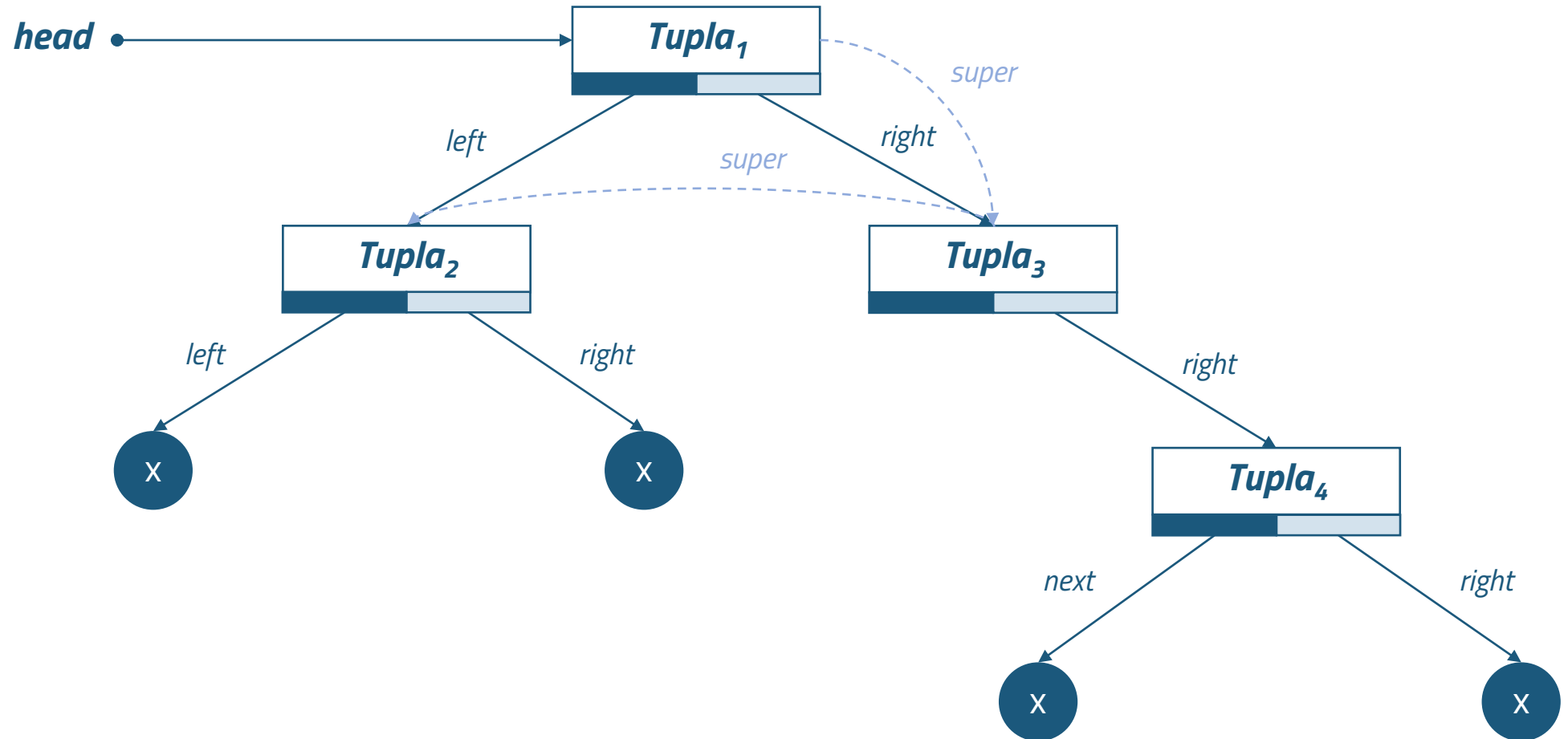
id: K
valor: V
identSup: K
numDepend: int

<
==

Nodo₂

id: K
valor: V
identSup: K
numDepend: int

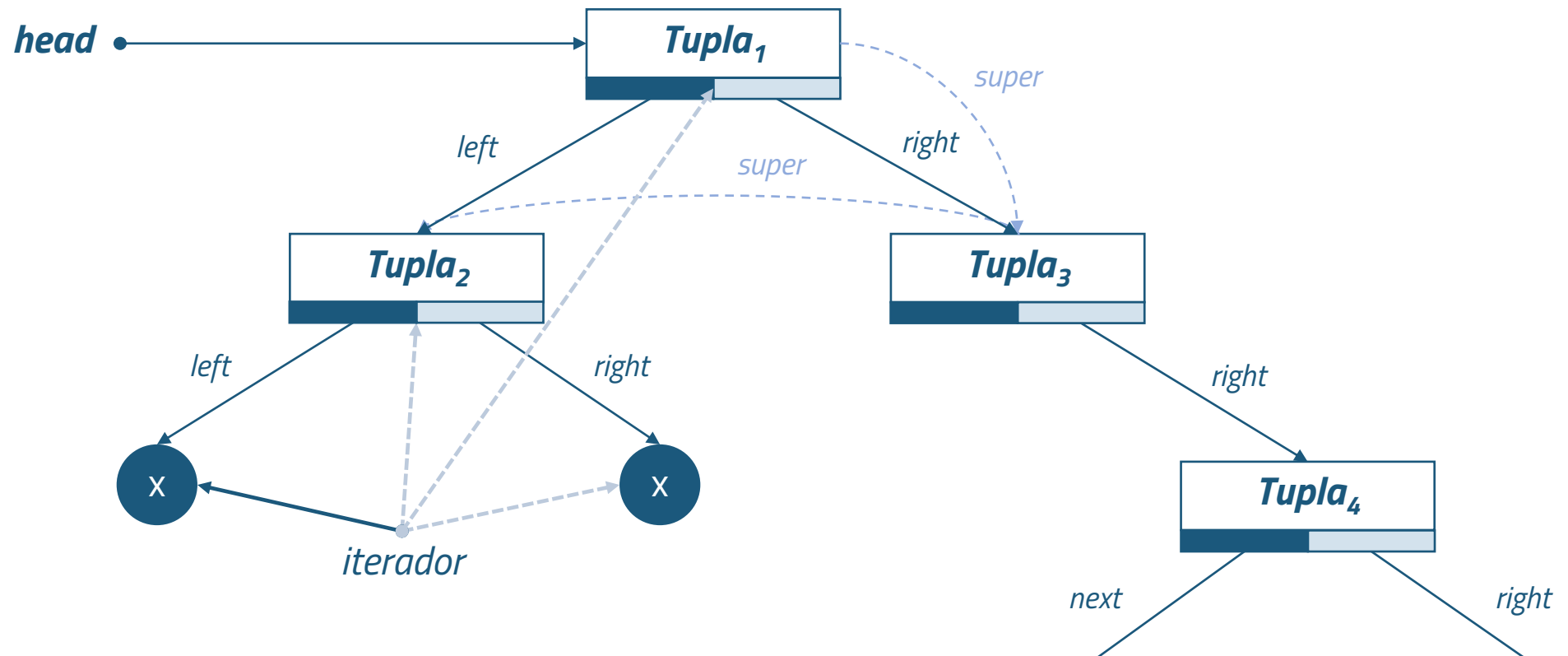
ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN INTERDEPENDIENTES*



operaciones

iniciarIterador: `colecInterdep c -> colecInterdep`

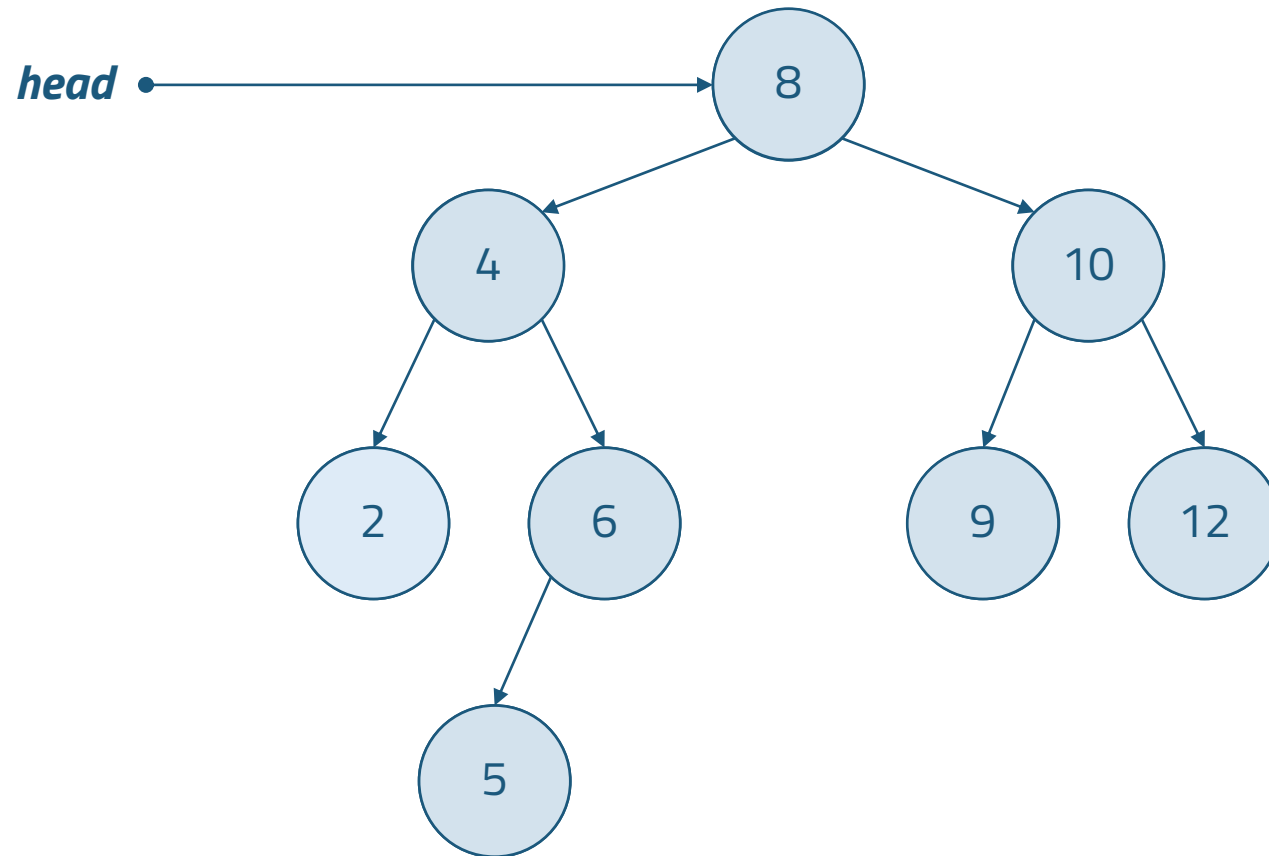
{Inicializa el iterador para recorrer los elementos de la colección *c*, de forma que el siguiente elemento a visitar sea el que tiene un *ident* anterior a los de todos los demás elementos.}



operaciones

iniciarIterador: `colecInterdep c -> colecInterdep`

{Inicializa el iterador para recorrer los elementos de la colección *c*, de forma que el siguiente elemento a visitar sea el que tiene un *ident* anterior a los de todos los demás elementos.}

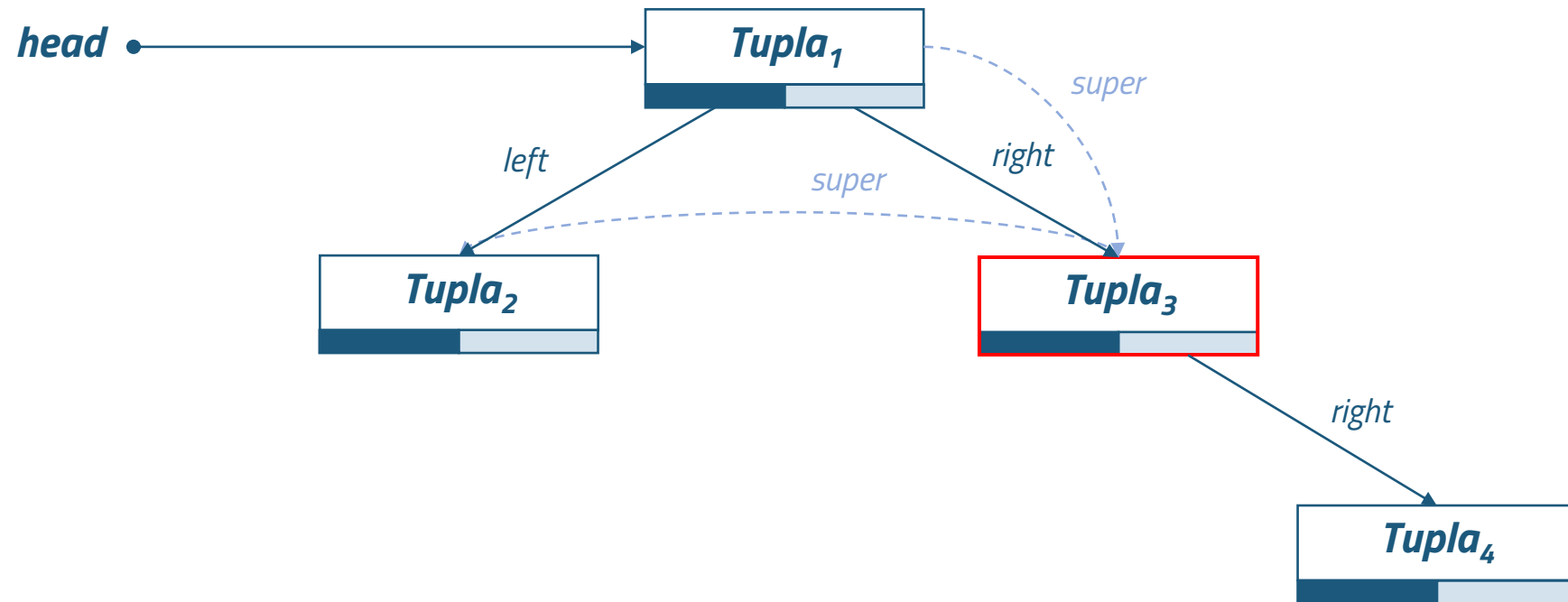


ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN INTERDEPENDIENTES*

operaciones

borrar: ident id, colecInterdep c -> colecInterdep

{Comprobar existeDependiente?(id,c).}

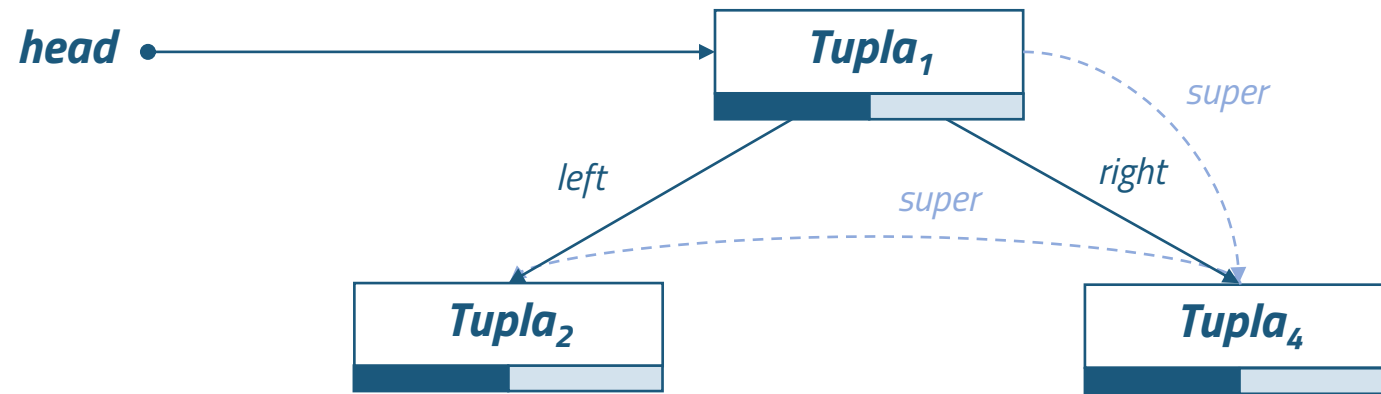


ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN INTERDEPENDIENTES*

operaciones

borrar: ident id, colecInterdep c -> colecInterdep

{Comprobar existeDependiente?(id,c).}

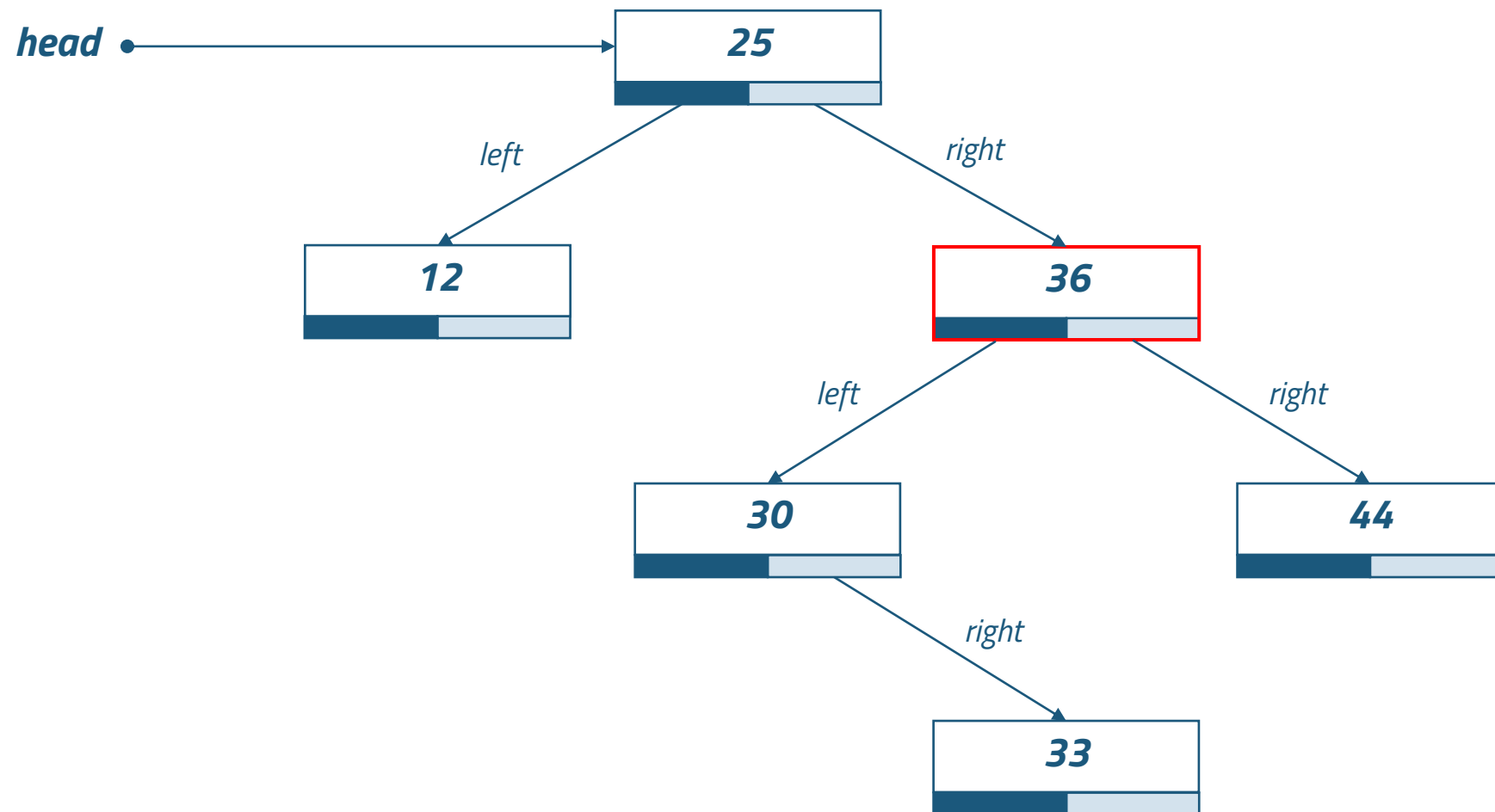


ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN INTERDEPENDIENTES*

operaciones

borrar: ident id, colecInterdep c -> colecInterdep

{Comprobar existeDependiente?(id,c).}

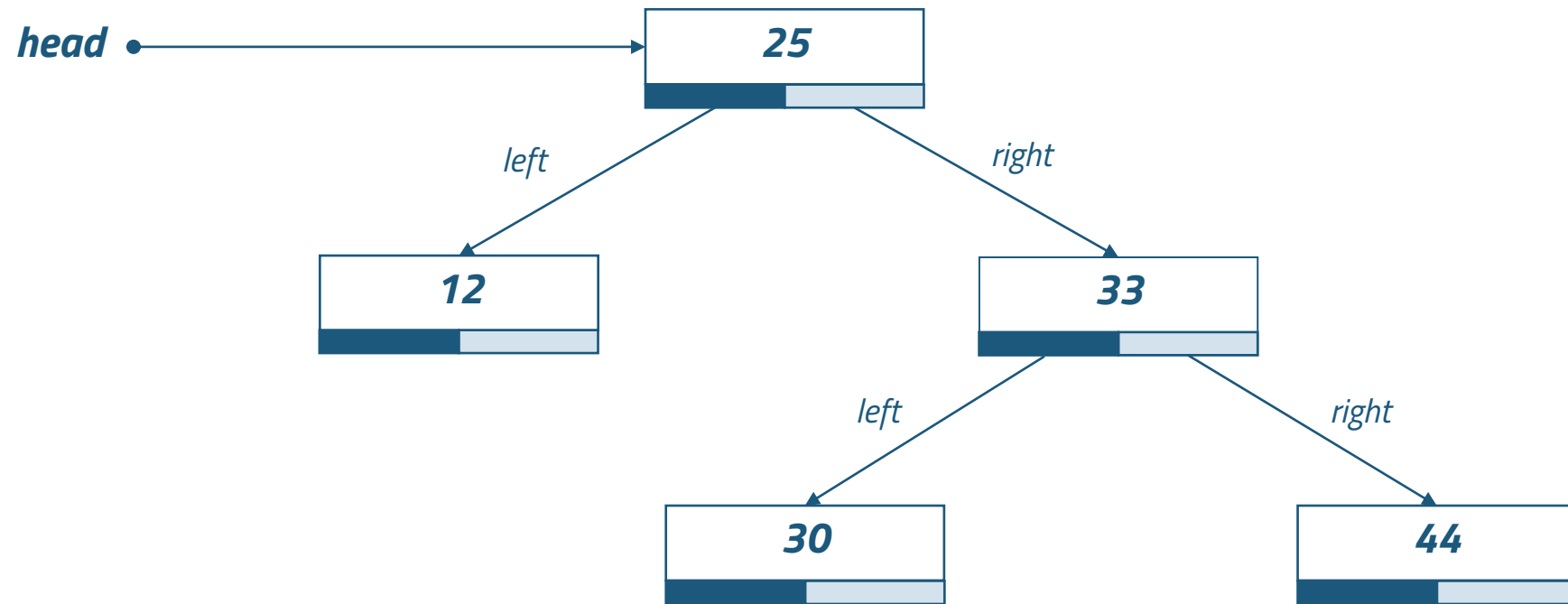


ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN INTERDEPENDIENTES*

operaciones

borrar: ident id, colecInterdep c -> colecInterdep

{Comprobar existeDependiente?(id,c).}



ENTREGA DE LA PRÁCTICA

- ▶ Comprobar resultados en **Hendrix**.
 - ▷ **C++ 11**.
- ▶ Utilizar siempre *“entrada.txt”* y *“salida.txt”* como rutas.
- ▶ **No** se pueden utilizar **excepciones**.
- ▶ **No** se pueden utilizar *smart pointers*.
- ▶ **Documentación** de código.
- ▶ **Incluir nombres completos y NIP de ambos autores al comienzo de cada fichero.**
- ▶ Solo **entregará un miembro** del equipo.
- ▶ **practica4_NIP1_NIP2.zip**.