

Universidad Zaragoza

Profesores:
Alfonso López Ruiz

Práctica 3. Implementación lineal en memoria dinámica de TAD genéricos

Estructuras de datos y algoritmos

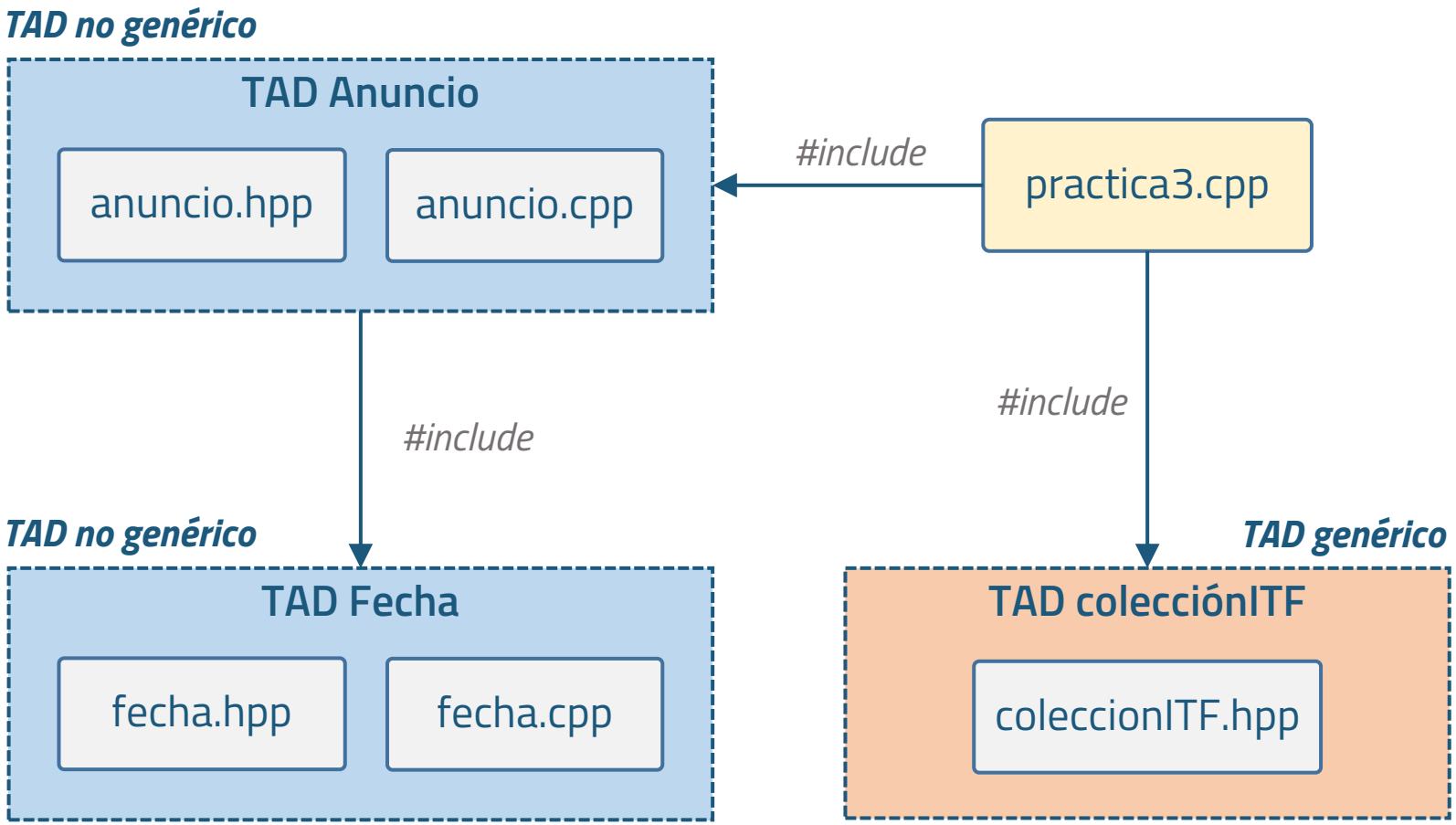
Grado en Ingeniería Informática – Curso académico 2026-2027

CC BY NC ND

OBJETIVOS DE LA PRÁCTICA

- Implementar un **TAD genérico en memoria dinámica con estructura lineal**.
  - Colección de ítems temáticos fechados (colecciónITF).
  - **Lista enlazada ordenada**.
- Implementar un **TAD no genérico**.
  - **Anuncio**.
- Comprobar de manera automática el funcionamiento de los TAD.
  - **Leer instrucciones y datos** de fichero.
  - **Construir un fichero de salida** con los resultados.

ESQUEMA DE LA SOLUCIÓN



TAD NO GENÉRICO Y GENÉRICO EN C++

TAD no genérico: anuncio

```
// anuncio.hpp
struct anuncio;
string suTitulo(const anuncio& e);
...

struct anuncio {
    friend string suTitulo(const anuncio& e);
    ...
  private:
    string titulo;
    ...
};

// anuncio.cpp: implementación
string suTitulo(const anuncio& e) { ... }
```

TAD genérico: colecciónITF<T>

```
// coleccionITF.hpp
template<typename T> struct coleccionITF;
template<typename T> int tamanyo(const coleccionITF<T>& c);
...

template<typename T>
struct coleccionITF {
    friend int tamanyo<T>(const coleccionITF<T>& c);
    ...
  private:
    ...                          // la representación
};

template<typename T>
int tamanyo(const coleccionITF<T>& c) { ... }
```

ESPECIFICACIÓN DEL TAD NO GENÉRICO *ANUNCIO*


```
espec Anuncios
usa cadenas, fechas
género anuncio
 {Los valores del TAD anuncio representarán tuplas formadas como:
 (identificador, título, descripción, categoría, publicación, finalización), siendo
 publicación y finalización sendas fechas (válidas según el calendario gregoriano),
 y el resto sendas cadenas.}

operaciones
     ...
fespec
```

ESPECIFICACIÓN DEL TAD NO GENÉRICO *ANUNCIO*


```
espec Anuncios
...

operaciones
     crearAnuncio: cadena ID, cadena t, cadena descrip, cadena categ,
                   fecha publi, fecha fi -> anuncio
     {Devuelve un anuncio compuesto con identificador ID, título t, descripción descrip,
      categoría categ, y con fecha de publicación publi y fecha de finalización fi.}

     suIdentificador: anuncio e -> cadena
     {Dado un anuncio e formado como (I,T,D,C,P,F) devuelve I, i.e. el identificador del anuncio e.}

     suTítulo: anuncio e -> cadena
     {Dado un anuncio e formado como (I,T,D,C,P,F) devuelve T, i.e. el título del anuncio e.}

     cambiarTítulo: anuncio e, cadena n -> anuncio
     {Dado un anuncio e formado como (I,T,D,C,P,F) y una cadena n, devuelve (I,n,D,C,P,F).}
fespec
```

→ ***Constructor***

→ ***Getter***

→ ***Getter***

→ ***Setter***

ESPECIFICACIÓN DEL TAD NO GENÉRICO *ANUNCIO*


```
espec Anuncios
...

operaciones
     suDescripción: anuncio e -> cadena                                  ⟶ Getter
     {Dado un anuncio e formado como (I,T,D,C,P,F) devuelve D, i.e. la descripción del anuncio e.}

     cambiarDescripción: anuncio e, cadena n -> anuncio                  ⟶ Setter
     {Dado un anuncio e formado como (I,T,D,C,P,F) y una cadena n, devuelve (I,T,n,C,P,F).}

     suCategoría: anuncio e -> cadena                                    ⟶ Getter
     {Dado un anuncio e formado como (I,T,D,C,P,F) devuelve C, i.e. la categoría del anuncio e.}

     cambiarCategoría: anuncio e, cadena n -> anuncio                    ⟶ Setter
     {Dado un anuncio e formado como (I,T,D,C,P,F) y una cadena n, devuelve (I,T,D,n,P,F).}

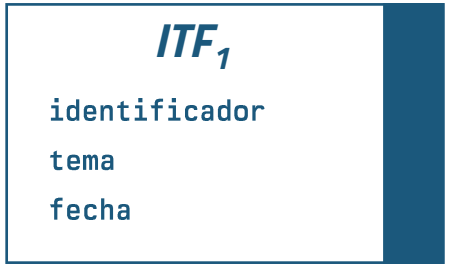
     suPublicación: anuncio e -> fecha                                   ⟶ Getter
     {Dado un anuncio e formado como (I,T,D,C,P,F) devuelve P, i.e. la fecha de publicación de e.}

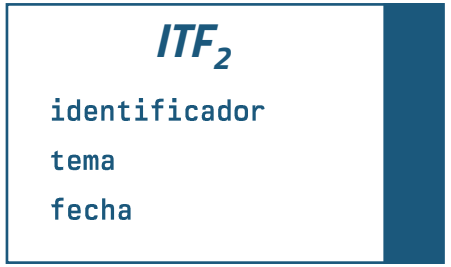
     suFinalización: anuncio e -> fecha                                  ⟶ Getter
     {Dado un anuncio e formado como (I,T,D,C,P,F) devuelve F, i.e. la fecha de finalización de e.}
fespec
```

ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN ITF*

```
espec coleccionesITF
usa booleanos, naturales, cadenas, fechas
parámetro formal
 género ITF {Ítem Temático Fechado}
 operación
 {Se requiere que el género ITF tenga definidas:}
      identificador: ITF i -> cadena
      {devuelve una cadena con el identificador del ítem i.}
      tema: ITF i -> cadena
      {devuelve una cadena con el tema del ítem i.}
      fecha: ITF i -> fecha
      {devuelve la fecha perteneciente al ítem i.}

fpf
```





DEL PARÁMETRO FORMAL A C++


```
// Especificación: el parámetro formal exige al género ITF
identificador: ITF i -> cadena
tema:          ITF i -> cadena
fecha:         ITF i -> fecha

// anuncio.hpp: anuncio las ofrece, así que puede ser un ITF
string identificador(const anuncio& a);    // su identificador
string tema(const anuncio& a);             // su categoría
Fecha  fecha(const anuncio& a);            // su fecha de publicación

// coleccionITF.hpp: el TAD genérico solo las llama, sin saber qué es T
if (identificador(i) == ident) ...
if (tema(i) == t) ...
if (anterior(fecha(i), f)) ...
```

Si anuncio no define alguna, el error aparece al compilar coleccionITF<anuncio>.

- El tipo de las fechas y la operación fecha no pueden llamarse igual.
- Clase Fecha disponible en **Material de las clases del grupo de tarde (422)**.

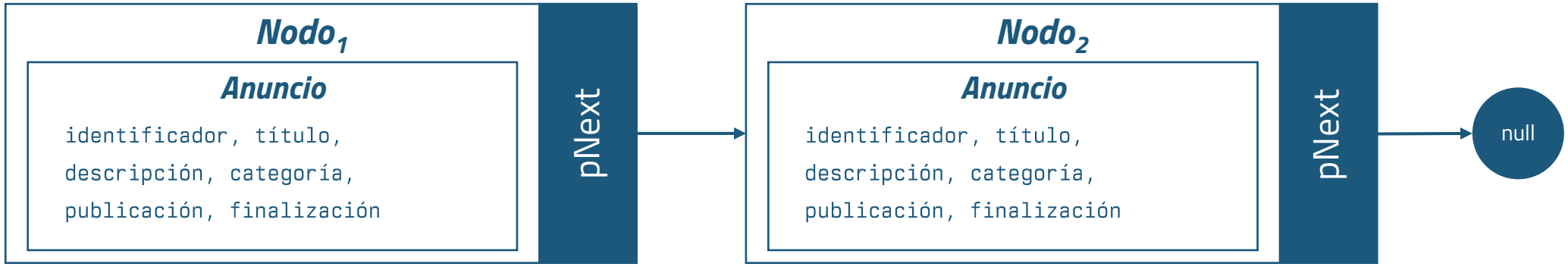
ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN ITF*



ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN ITF*

```
espec coleccionesITF
usa booleanos, naturales, cadenas, fechas
parámetro formal
 género ITF {Ítem Temático Fechado}
 operación
 {Se requiere que el género ITF tenga definidas:}
     identificador: ITF i -> cadena
     {devuelve una cadena con el identificador del ítem i.}
     tema: ITF i -> cadena
     {devuelve una cadena con el tema del ítem i.}
     fecha: ITF i -> fecha
     {devuelve la fecha perteneciente al ítem i.}

fpf
```



EL NODO DENTRO DE LA COLECCIÓN

struct Nodo

```
template<typename T>
struct coleccionITF {
    friend int tamanyo<T>(const coleccionITF<T>& c);
    ...
  private:
    struct Nodo {      // tipo auxiliar
        ...
    };
    ...
    int total;         // número de ítems
};
```

Uso de Nodo

```
template<typename T>
bool esta(const string& ident, const coleccionITF<T>& c) {
    typename coleccionITF<T>::Nodo* act = c.primero;
    while (act != nullptr ...) {
        ...
    }
    ...
}

// Crear un nodo nuevo
typename coleccionITF<T>::Nodo* nuevo = new typename coleccionITF<T>::Nodo;
```

Nodo es un struct como `anuncio`, declarado dentro de `coleccionITF`.

- Está en `private`, así que solo lo usan `coleccionITF` y sus friend; `main` no lo ve.
- Fuera del struct se escribe `coleccionITF<T>::Nodo`; `typename` le dice a C++ que es un tipo.

ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN ITF*

operaciones

crear: -> colecciónITF

{Crea una colección de ITF vacía, sin ítems.}

esVacía?: colecciónITF c -> booleano

{Devuelve verdad si y sólo si c no contiene ningún ítem.}

tamaño: colecciónITF c -> natural

{Devuelve el número de ítems que hay en la colección c.}

añadir: colecciónITF c, ITF i -> colecciónITF

{Si en c no hay ningún ítem con identificador igual al de i, devuelve la colección resultante de añadir i a c. En caso contrario, devuelve una colección igual a c.}

actualizar: colecciónITF c, ITF i -> colecciónITF

{Si en c hay un ítem con identificador igual al de i, devuelve la colección resultante de sustituirlo por i. En caso contrario, devuelve una colección igual a c.}

borrar: cadena ident, colecciónITF c -> colecciónITF

{Si en c hay algún ítem con identificador igual a ident, lo borra. Si no, devuelve c.}

ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN ITF*

operaciones

está?: cadena ident, colecciónITF c -> booleano
{Devuelve verdad si y sólo si en c hay algún ítem con identificador igual a ident.}

parcial obtenerITF: cadena ident, colecciónITF c -> ITF
{Devuelve el ítem de c con identificador igual a ident.
Parcial: la operación no está definida si no está?(ident,c).}

obsoletos: fecha f, cadena separador, colecciónITF c -> cadena
{Concatena los identificadores de los ítems de c con fecha estrictamente anterior
a f, separados entre sí por la cadena separador.}

existeTema: cadena t, colecciónITF c -> booleano
{Devuelve verdad si y sólo si en c hay algún ítem i con tema igual a t.}

delTema: cadena t, cadena separador, colecciónITF c -> cadena
{Concatena los identificadores de los ítems de c con tema igual a t, separados
entre sí por la cadena separador.}

No se requiere optimizar obsoletos, purgar, existeTema, delTema ni zanjarTema.

Se podrían indexar los ítems por tema o por fecha con estructuras auxiliares, pero...

- Presenta mayor dificultad, en esta práctica y en la siguiente.
- No excusará comportamientos incorrectos o ineficientes.

DE LA ESPECIFICACIÓN A C++


```
// Devuelve una colección: void y la colección por referencia
añadir: colecciónITF c, ITF i -> colecciónITF

// Consulta: devuelve el valor; lo que no se modifica, const &
tamaño: colecciónITF c -> natural

// Parcial: bool (¿estaba definida?) y el resultado por referencia
parcial obtenerITF: cadena ident, colecciónITF c -> ITF
```

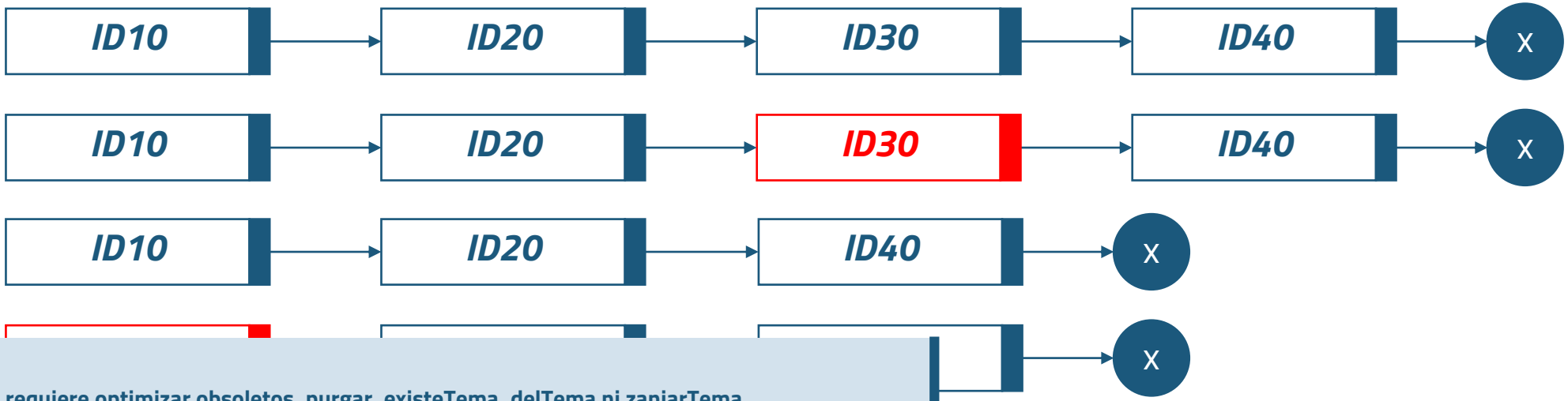
Si una operación no devuelve nada, `main` pregunta antes y después.

Por ejemplo, `tamanyo(c)` antes y después de `anyadir` dice si el anuncio ha entrado.

ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN ITF*


```
operaciones
    purgar: fecha f, coleccionITF c -> coleccionITF
    {Elimina de c todos los ítems con fecha estrictamente anterior a f.}

    zanjarTema: cadena t, coleccionITF c -> coleccionITF
    {Elimina de c todos los ítems cuyo tema es igual a t.}
```



No se requiere optimizar obsoletos, purgar, existeTema, delTema ni zanjarTema.

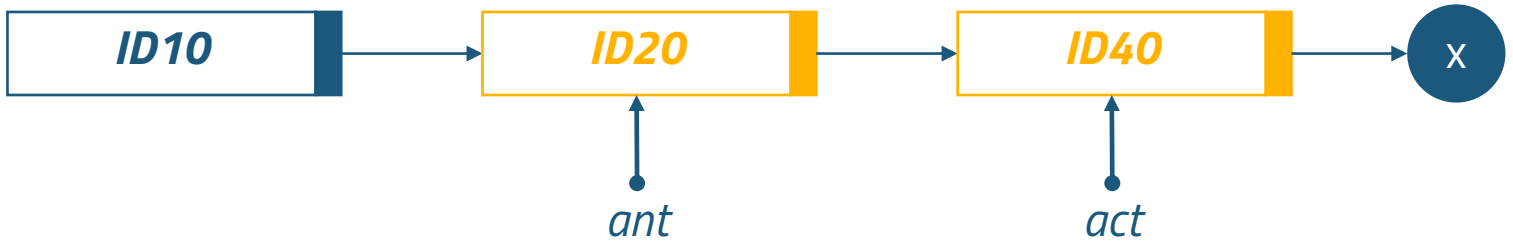
Se podrían indexar los ítems por tema o por fecha con estructuras auxiliares, pero...

- Presenta mayor dificultad, en esta práctica y en la siguiente.
- No permitirá obtener más nota.
- No excusará comportamientos incorrectos o ineficientes.

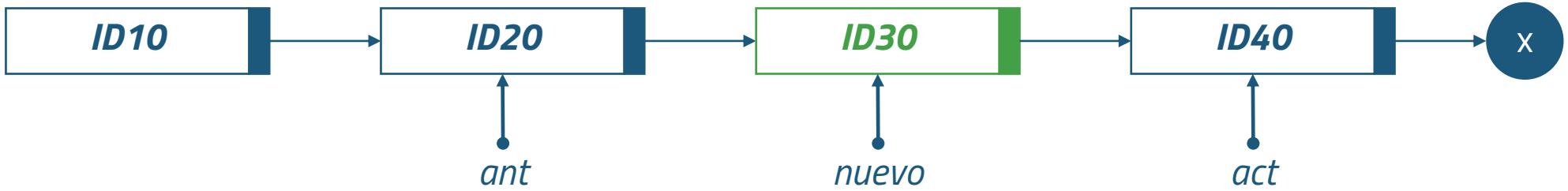
IMPLEMENTACIÓN: *INSERTAR EN ORDEN*

```
operaciones
 añadir: colecciónITF c, ITF i -> colecciónITF
 {Inserción en orden: dos punteros, ant y act, buscan el primer nodo con clave ≥ la de i.}
```

1. Buscar: ant y act avanzan mientras act.clave < ID30



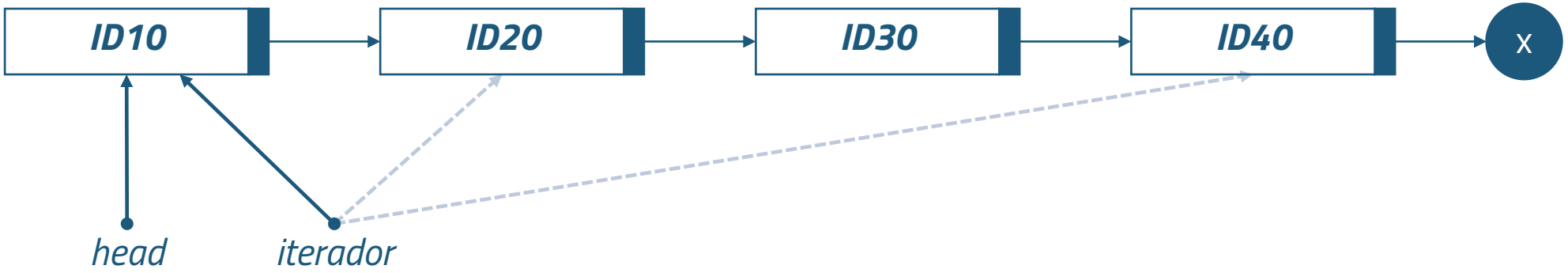
2. Enlazar: nuevo.next ← act; ant.next ← nuevo



Si act.clave = ID30, la clave ya existe y la colección no cambia.

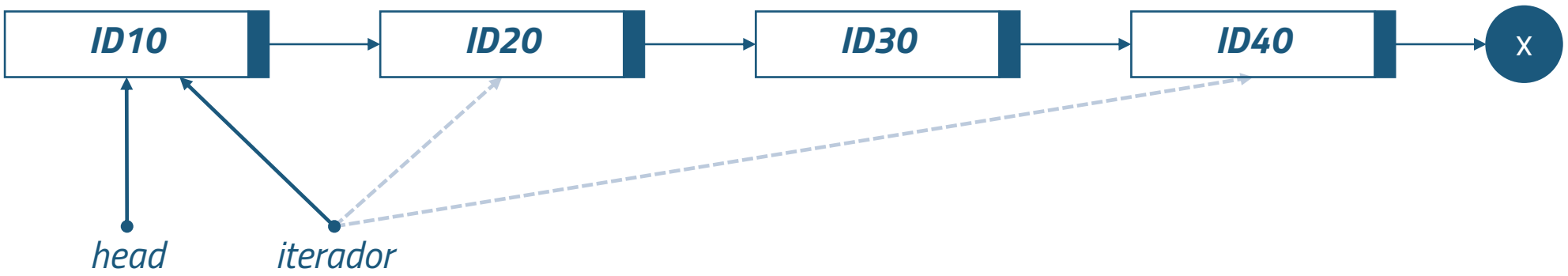
ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN ITF*

```
operaciones
 iniciarIterador: colecciónITF c -> colecciónITF
 {Inicializa el iterador para recorrer los ítems de c, de forma que el siguiente ítem
  a visitar sea el de identificador (lexicográficamente) menor.}
```



RECORRIDO CON EL *ITERADOR*


```
// LT: recorrer la colección con el iterador, de menor a mayor identificador
s << "-----LISTADO: " << tamanyo(c) << endl;
iniciarIterador(c);
while (existeSiguiente(c)) {
    siguiente(c, a);        // a: el anuncio que señala el iterador
    ...                     // escribir a con el formato de LT
    avanza(c);
}
s << "-----" << endl;
```



ESPECIFICACIÓN DEL TAD GENÉRICO *COLECCIÓN ITF*

operaciones

existeSiguiente?: colecciónITF c -> booleano

{Devuelve verdad si queda algún ítem por visitar con el iterador de c.}

parcial siguiente: colecciónITF c -> ITF

{Devuelve el siguiente ítem a visitar con el iterador de c.
Parcial: la operación no está definida si no existeSiguiente?(c).}

parcial avanza: colecciónITF c -> colecciónITF

{Prepara el iterador para visitar el siguiente ítem, el de menor identificador entre los no visitados. Parcial: la operación no está definida si no existeSiguiente?(c).}

LECTURA DE INSTRUCCIONES

- Tipos de instrucciones:
  - **A**ñadir un anuncio a la colección.
    - Identificador, título, descripción, categoría, publicación y finalización.
  - **C**ambiar un anuncio de la colección.
    - Identificador, título, descripción, categoría, publicación y finalización.
  - **O**btener la información de un anuncio.
    - Identificador.
  - Comprobar si **E**xiste un anuncio.
    - Identificador.

LECTURA DE INSTRUCCIONES

- Tipos de instrucciones:
  - Comprobar si **E**xiste una **C**ategoría.
    - Categoría.
  - **B**orrar un anuncio.
    - Identificador.
  - **B**orrar los anuncios de una **C**ategoría.
    - Categoría.
  - **P**urgar los anuncios anteriores a una fecha.
    - Fecha (aaaa-mm-dd).

LECTURA DE INSTRUCCIONES

- Tipos de instrucciones:
  - **L**istar los anuncios **O**bsoletos.
    - Fecha (aaaa-mm-dd).
  - **L**istar una **C**ategoría.
    - Categoría.
  - **L**istar **T**odos los anuncios.
    - Sin datos.

```
ifstream f;
f.open("entrada.txt");

string instruccion, salto;
while (f >> instruccion) {
   getline(f,salto);
   if (instruccion == "A") {
   //... sigue el programa
   } else if (instruccion == "O") {
   //... sigue el programa
   }
} //... sigue el programa
```

ERRORES FRECUENTES

- En la salida, los campos van en otro orden que en la entrada: título, **categoría**, descripción.
- La fecha de un anuncio en la colección es la de **publicación**, no la de finalización.
- **" --- "** lleva un espacio a cada lado; sin resultados quedan **dos espacios** entre los guiones.
- **Mayúsculas y minúsculas** cuentan: "en venta" no es "En venta".
- Los datos pueden tener **varias palabras**: se leen con getline, no con >>.

PLANIFICACIÓN

- Semanas 1-2:
  - Implementar y probar el **TAD anuncio**.
  - De colecciónITF: ***crear, tamaño, esVacía?, añadir, está?, obtenerITF, existeTema, delTema*** e ***iterador***.
  - **A, E, O, EC, LC, LT**.
- Semanas 3-4:
  - Resto de colecciónITF: ***actualizar, borrar, obsoletos, purgar, zanjarTema***.
  - **C, B, BC, P, LO**.
  - Comprobar la lectura de ***"entrada.txt"*** y el formato de ***"salida.txt"***.

ENTREGA DE LA PRÁCTICA

- Comprobar resultados en **lab000**.
  - **g++ -std=c++11 *.cpp**.
- Utilizar siempre ***"entrada.txt"*** y ***"salida.txt"*** como rutas.
- **La salida debe coincidir exactamente** con la exigida en el guion.
  - Mayúsculas, minúsculas, espacios, caracteres, etc.
- **No** se pueden utilizar **excepciones** ni **smart pointers**.
- **No** se puede utilizar **Programación Orientada a Objetos** ni **STL**.
- **Documentación** de código.
- **Nombres completos y NIP de todos los autores al comienzo de cada fichero**.

ENTREGA DE LA PRÁCTICA

- Fecha límite: **17-11-2026** (incluido).
- Solo **entregará un miembro** del equipo.
- **practica3_NIP1_NIP2_NIP3.zip**.
- Dos ficheros ***"entrada.txt"*** y ***"salida.txt"***.
  - **Significativamente diferentes** a los propuestos.
- **Estructura plana**: **solo código fuente y los dos ficheros anteriores**.