

Estructuras de Datos y Algoritmos

Tablas multidimensionales

LECCIÓN 20

© All wrongs reversed – bajo licencia [CC-BY-NC-SA 4.0](#)



Universidad
Zaragoza

Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España

Grado en Ingeniería Informática
UNIVERSIDAD DE ZARAGOZA



Adaptadas de diapositivas de Javier Campos

Índice

1 Conceptos

2 Especificación

3 Implementación

Índice

1 Conceptos

2 Especificación

3 Implementación

Tablas funcionales

Conceptos

- Tipos funcionales que representan funciones:
 - Cuyo dominio es el producto cartesiano de varios géneros de claves
 - Cuyo rango es el género de los valores

$$f : \mathcal{D}_{clave1} \times \mathcal{D}_{clave2} \dots \mathcal{D}_{claveN} \mapsto \mathcal{D}_{valores}$$

- f suele ser parcial (i.e., existen tuplas de claves sin valor asociado)

Tablas funcionales

Ejemplo

- Relación entre los estudiantes y las asignaturas de la EINA y sus notas
- Tabla bidimensional: conjunto (o colección) de ternas $\langle clave_{alumno}, clave_{asignatura}, notas \rangle$, donde:
 - No podrá haber dos ternas que compartan simultáneamente la misma $clave_{alumno}$ y $clave_{asignatura}$
 - El valor asociado a un par $\langle clave_{alumno}, clave_{asignatura} \rangle$ son las notas de ese alumno en esa asignatura
- Las proyecciones de la tabla bidimensional sobre cada una de sus claves definen sendas tablas unidimensionales
 - Dada una clave de alumno, $clave_{alum1}$: tabla con los datos de asignaturas en las que está matriculado y sus notas en cada una de ellas
 - Dada una clave de asignatura, $clave_{asig1}$: tabla con los datos de los alumnos que están matriculados en esa asignatura y sus notas en esa asignatura

Índice

1 Conceptos

2 Especificación

3 Implementación

Tablas funcionales

Especificación

espec tablasBidimensionales

usa tablasGenéricas

parámetros formales

géneros claveA, claveB, valor

fpf

géneros

```
{concretamos las tablasGenéricas unidimensionales para las proyecciones de la tabla bidimensional:}
tablaA = tabla(claveA, valor)
tablaB = tabla(claveB, valor)
{nuevo género de tabla bidimensional:}
tablaAB {Los valores del género tablaAB representan conjuntos de ternas (claveA, claveB, valor) en los
no se permiten claves repetidas, estando las claves formadas por dos partes: (claveA, claveB)}
```

operaciones

crear: -> tablaAB

{Devuelve una tabla bidimensional vacía}

añadir: tablaAB t, claveA ca, claveB cb, valor v -> tablaAB

{Devuelve una tabla igual a la tabla resultante de añadir la terna (ca, cb, v) a t; si en t ya había una terna (ca, cb, v'), entonces devuelve una tabla igual a la resultante de sustituir dicha terna por (ca, cb, v) en t}

pertenece? : claveA ca, claveB cb, tablaAB t -> booleano

{Devuelve verdad si y sólo si en t hay alguna terna (ca, cb, v)}

parcial obtenerValor: claveA ca, claveB cb, tablaAB t -> valor

{Si en t existe alguna terna con el par de claves (ca, cb) devuelve el valor asociado a ellas en dicha terna;

Parcial: la operación no está definida si not(pertenece?(ca, cb, t))}

quitar: claveA ca, claveB cb, tablaAB t -> tablaAB

{Si pertenece?(ca, cb, t), devuelve una tablaAB igual a la resultante de borrar de t la terna (ca, cb, v) que tiene dichas claves; si not(pertenece?(ca, cb, t)), devuelve una tabla igual a t}

proyectarA: tablaAB t, claveA ca -> tablaB

{Devuelve una tablaB unidimensional con todos los pares (cb, v) tales que en la tabla bidimensional t existe una terna (ca, cb, v)}

proyectarB: tablaAB t, claveB cb -> tablaA

{Devuelve una tablaA unidimensional con todos los pares (ca, v) tales que en la tabla bidimensional t existe una terna (ca, cb, v)}

fespec

Índice

- 1 Conceptos
- 2 Especificación
- 3 Implementación**

Tablas funcionales

Implementación estática

Tipo matricial

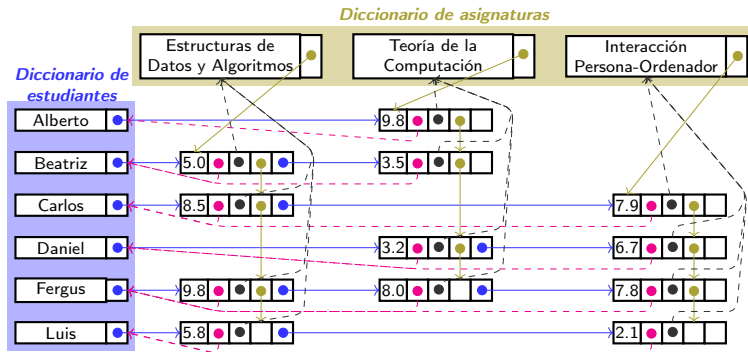
	EDA	TC	IPO
Alberto		9.8	
Beatriz	5.0	3.5	
Carlos	8.5		7.9
Daniel		3.2	6.7
Fergus	9.8	8.0	7.8
Luis	5.8		2.1

Tablas funcionales

Implementación dinámica

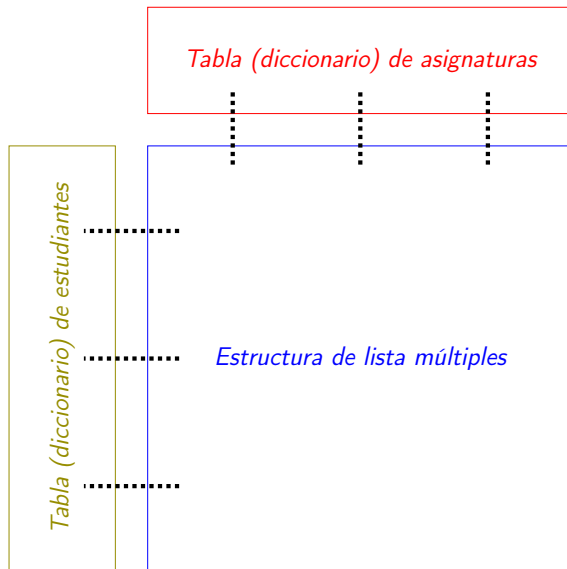
■ Implementación con estructura de listas múltiples

- *Colección de datos dinámicos enlazados mediante punteros en la que cada dato dinámico tiene más de un campo de tipo puntero*
- Cada dato puede pertenecer a más de una lista a la vez



Tablas funcionales

Implementación dinámica



Tablas funcionales

Implementación dinámica

tipos

```
estudiante = ... {información de un estudiante}
asignatura = ... {información de una asignatura}
    nota = ... {información de <estudiante, asignatura> (por ejemplo, incluye la
ptNodoEstudiante = ↑nodoEstudiante;
    nodoEstudiante = registro
        datoEstudiante: estudiante;
        primeraAsignatura: ptNodoNota
    freg;
ptNodoAsignatura = ↑nodoAsignatura;
    nodoAsignatura = registro
        datoAsignatura: asignatura;
        primerEstudiante: ptNodoNota
    freg;

ptNodoNota = ↑nodoNota;
    nodoNota = registro
        datoNota: nota;
        quéEstudiante: ptNodoEstudiante;
        quéAsignatura: ptNodoAsignatura;
        sigEstudiante, sigAsignatura: ptNodoNota
    freg
diccionarioEstudiantes = ... {representación de un diccionario unidimensional de datos de tipo
diccionarioAsignaturas = ... {representación de un diccionario unidimensional de datos de tipo
    escuela = registro
        losEstudiantes: diccionarioEstudiantes;
        lasAsignaturas: diccionarioAsignaturas
    freg
```

Tablas funcionales

Implementación dinámica

Ejemplo de operación a implementar

- *Dada una asignatura, escribir los datos y notas de los estudiantes que están matriculados en ella*

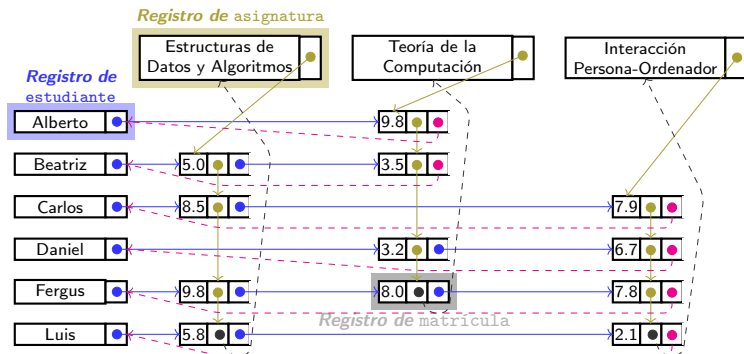
Tablas funcionales

Implementación dinámica

```
procedimiento escribeEstudiantes(ent EINA: escuela; ent c: claveAsignatura)
{Escribe en pantalla la información (notas) de los estudiantes matriculados en la
 asignatura de clave c.}
variables
    unaAsignatura: ptNodoAsignatura;
    unEstudiante: ptNodoNota
principio
    buscar(EINA, c, unaAsignatura);
    {Devuelve en unaAsignatura un puntero al registro de asignatura correspondiente
     a la asignatura de clave c; su implementación depende de la representación del
     diccionario de asignaturas}
    unEstudiante := unaAsignatura↑.primerEstudiante;
    mientrasQue unEstudiante ≠ nil hacer
        {unEstudiante apunta a un registro de nota de un estudiante en la asignatura
         de clave c}
        escribeEstudiante(unEstudiante↑.quéEstudiante);
        {escribe los datos personales del estudiante almacenados en el registro
         apuntado por ese campo}
        escribeNota(unEstudiante↑.datoNota); {escribe la nota}
        unEstudiante := unEstudiante↑.sigEstudiante
    fmq
fin
```

Tablas funcionales

Estructura de listas múltiples (otra implementación)



- El registro matrícula no contiene información de la asignatura o estudiante; vienen determinados de manera implícita por la lista de la que forman parte
- En un lenguaje en el que los punteros están especializados en apuntar a datos de un tipo, es necesario que todos los registros sean de un mismo tipo (con campos variantes, si el lenguaje lo permite)
 - En C++ se llaman *uniones etiquetadas*

Tablas funcionales

Implementación dinámica

tipos

```
estudiante = ... {información de un estudiante}
asignatura = ... {información de una asignatura}
matrícula = ... {información de <estudiante, asignatura> (por ejemplo, incluye la nota)}
claseNodo = (estud, asig, matr)

ptNodo = ↑nodo;
{Registro con campo para los tres tipos}
nodo = registro
      clase: claseNodo;
      elEstud: estudiante; primeraAsign: ptNodo;
      laAsign: asignatura; primerEstud: ptNodo;
      laMatri: matrícula; sigEstud, sigAsign: ptNodo
freg
```

Tablas funcionales

Implementación dinámica

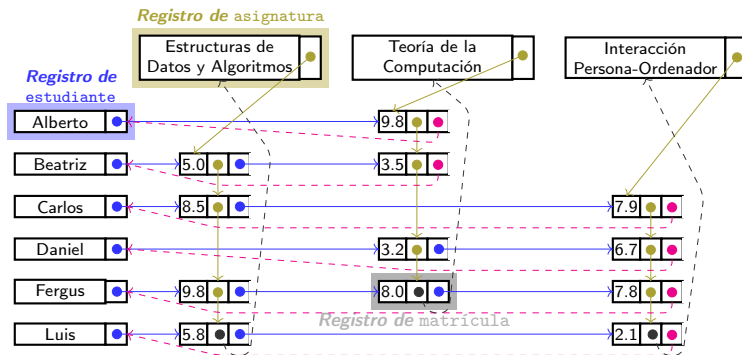
tipos

```
estudiante = ... {información de un estudiante}
asignatura = ... {información de una asignatura}
matrícula = ... {información de <estudiante, asignatura> (por ejemplo, incluye la nota)}
claseNodo = (estud, asign, matr)

ptNodo = ↑nodo;
{Registro con campos variantes}
  nodo = registro
    clase: claseNodo;
    selección
      clase = estud: (elEstud: estudiante; primeraAsign: ptNodo);
      clase = asign: (laAsign: asignatura; primerEstud: ptNodo);
      clase = matri: (laMatri: matrícula; sigEstud, sigAsign: ptNodo);
    fselección
  freg
  {en C++ esto se llama uniones etiquetadas}
```

Tablas funcionales

Estructura de listas múltiples (otra implementación)



¿Qué falta?

- ¿Cómo acceder a los datos de una asignatura concreta o de un estudiante concreto?
- ¿Representación de la colección de asignaturas?
- ¿Representación de la colección de estudiantes?

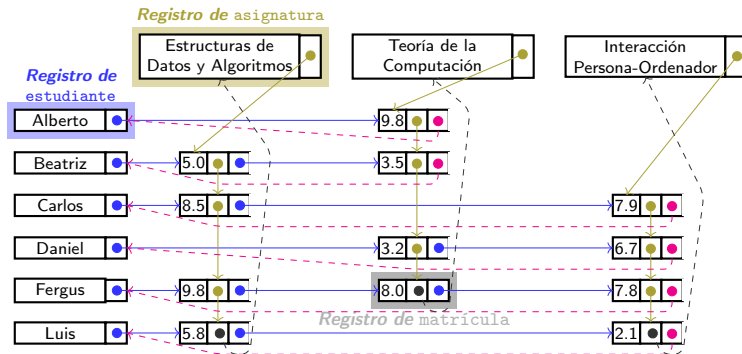
Tablas funcionales

Estructura de listas múltiples (otra implementación)

```
procedimiento escribeEstudiantes(ent c: claveAsign)
{ Escribe en pantalla la información de los estudiantes matriculados en la
  asignatura de clave c. }
variables
  unaAsign, unEstud, aux: ptNodo
principio
  observa(c, unaAsign); {devuelve en unaAsign un apuntador al registro de
    asignatura correspondiente a la asignatura de clave c; su implementación depende
    de la representación del diccionario de asignaturas}
  unEstud := unaAsign↑.primerEstud;
  mientrasQue unEstud↑.clase = matri hacer
  { unEstud apunta a registro de matrícula de un estudiante en la asignatura de clave c }
    aux := unEstud↑.sigAsign;
    mientrasQue aux↑.clase = matri hacer
      { recorrer el resto de registros de matrícula de ese estudiante }
        aux := aux↑.sigAsign
    fmq; {aux apunta a un registro de clase estud.}
    {escribe los datos del estudiante almacenados en el registro apuntado por aux}
    escribeEstudiante(aux);
    {escribe las notas almacenadas en el registro apuntado por unEstud}
    escribeNotas(unEstud);
    unEstud := unEstud↑.sigEstud
  fmq
fin
```

Tablas funcionales

Estructura de listas múltiples (otra implementación)

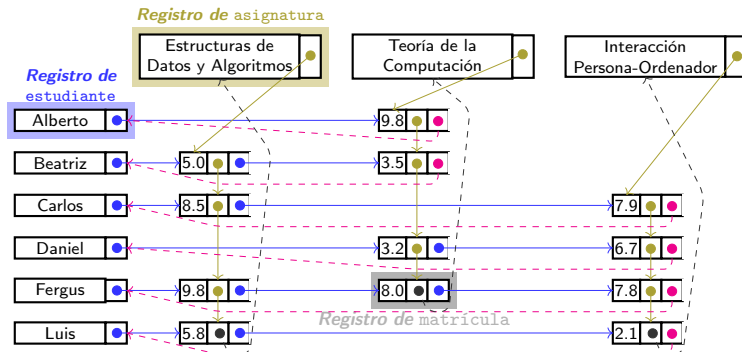


¿Cómo se podría mejorar la eficiencia?

■ ¿Alternativas?

Tablas funcionales

Estructura de listas múltiples (otra implementación)



¿Cómo se podría mejorar la eficiencia?

■ ¿Alternativas?

■ ¿Y si borramos una asignatura? ¿o un estudiante?

- Borrar todos sus datos, incluidos los registros de matrícula en los que está involucrado...

Antes de entregar

La rúbrica como *checklist*: ejercicio de especificación

Las rúbricas genéricas de especificación y de diseño de estructuras están publicadas en Moodle. Úsalas **antes de entregar**, criterio a criterio:

- ☐ **Descripción del TAD** (15 %): ¿queda claro qué representa y qué relaciones hay entre los elementos del sistema?
- ☐ **Entidades** (15 %): ¿cada entidad tiene sus características y su **identificador** (clave) definidos?
- ☐ **Relaciones** (10 %): ¿están indicadas las **cardinalidades** y dependencias?
- ☐ **Restricciones** (5 %): unicidad de identificadores, condiciones entre entidades, ...
- ☐ **Operaciones** (40 %): ¿cada una tiene dominio, rango y, si es *parcial*, sus **condiciones de no definición**? ¿Dice **qué** y no **cómo**? **F1**
- ☐ **Claridad** (5 %): sin ambigüedades ni fugas de implementación

Antes de entregar

La rúbrica como *checklist*: ejercicio de diseño de estructuras

- ☐ **Justificación de la estructura** (30 %): ¿cada estructura se elige **por un requisito** del enunciado? (tabla de decisión, lección 19) **F3**
- ☐ **Descripción de la estructura** (20 %): ¿están claras las claves y cómo se **enlazan** las entidades? (punteros recíprocos, nada duplicado) **F2**
- ☐ **Eficiencia temporal y espacial** (25 %): ¿coste asintótico de **cada** operación, con las variables definidas y el caso declarado? (6 pasos) **F4**
- ☐ **Operaciones y su impacto** (20 %): ¿se explica cómo la estructura facilita cada operación pedida y a qué coste?
- ☐ **Alternativas** (5 %): ¿se descarta razonadamente al menos una alternativa?

Esta semana: resolución del **examen de enero de 2025** con la rúbrica delante, y **co-corrección** en la clase de problemas. Calibra qué penaliza cada casilla *antes* del examen real.

Estructuras de Datos y Algoritmos

Tablas multidimensionales

LECCIÓN 20

☺ All wrongs reversed – bajo licencia [CC-BY-NC-SA 4.0](#)



Universidad
Zaragoza

Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, España

Grado en Ingeniería Informática
UNIVERSIDAD DE ZARAGOZA

