

TEMA 2

Estructuras de datos lineales

ESTRUCTURAS DE DATOS

Objetivos

- Conocer la especificación algebraica de las principales estructuras de datos lineales: Lista, pila y cola
- Conocer diferentes alternativas sobre el diseño e implementación de estructuras de datos lineales
- Conocer las cuestiones de rendimiento fundamentales sobre el uso de estructuras de datos lineales

Contenidos

2.0. Preliminares

2.1. Listas

2.2. Pilas

2.3 Colas

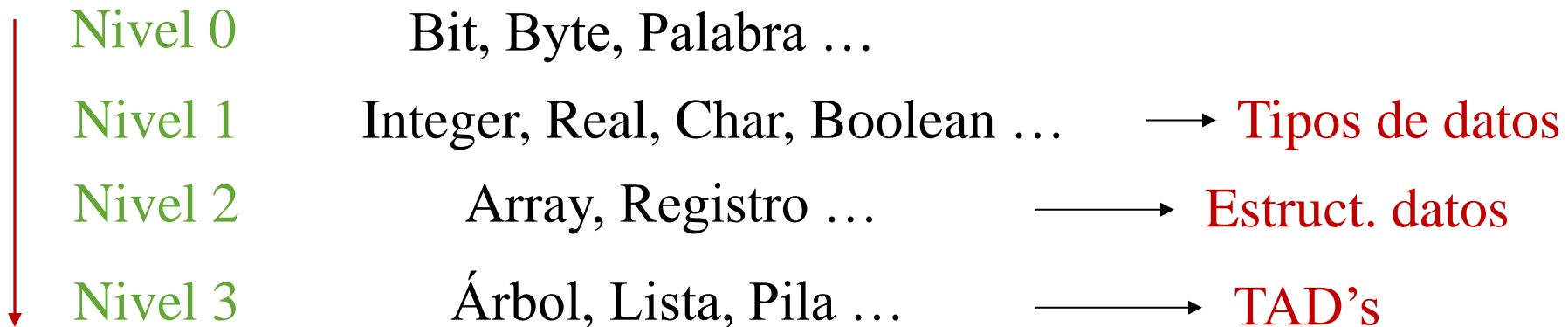
2.0 Preliminares

¿Qué es una estructura de datos?

- Agrupación de datos relacionados
- Tienen asociados ciertas operaciones
- Ejemplos de estructuras de datos son:
 - Arrays, registros, conjuntos y ficheros
- Algunas de sus operaciones:
 - Arrays: operación de selección []
 - Registro: operación de selección .
 - Conjunto: in, +, -
 - Fichero: rewrite, reset, read, write

2.0 Preliminares: Estructuras de datos

- Pascal incorpora directamente estas estructuras de datos
- Cada nivel de abstracción de datos utiliza niveles de abstracción anteriores
- Podemos definir una jerarquía de datos



2.1 Listas

- Una lista es una colección de elementos homogéneos dispuestos en orden
- En orden nos referimos a que a cualquier elemento (excepto al primero) le precede uno y le sigue otro (excepto al último)
- Cada elemento tiene asignada una posición
- Esta posición puede venir dada por una condición o no (por ejemplo dependiendo del valor del elemento, o un campo clave,...)

2.1 Listas

- Supongamos que la lista puede albergar un número ilimitado de elementos y que no hay orden condicionado de elementos
- A nivel abstracto una lista puede considerarse como una secuencia de cero o más elementos y representarse enumerando dichos elementos:

[1, 3, 5, -8, 4] o [1, [3, [5, [-8, [4, []]]]]]

['c', 'a', 'e', 'b'] o ['c', ['a', ['e', ['b', []]]]]

2.1 Listas

- La lista viene parametrizada por el tipo de elemento que alberga.
- Por ello el TipoElemento será un parámetro genérico del TipoLista
- ¿Qué operaciones definimos para el TAD Lista?
- Posibles operaciones serán:
 - CrearVacia, ListaVacia, Insertar, Suprimir, Longitud,...
- Especificación algebraica:

2.1 Listas

ESPECIFICACION Listas

PARAMETROS GENERICOS

TipoElemento

FIN PARAMETROS GENERICOS

TIPOS TipoLista

OPERACIONES

(* CONSTRUCTORAS GENERADORAS *)

CrearVacia: \rightarrow TipoLista

(* La lista vacía se suele representar por [] *)

Construir: TipoElemento x TipoLista \rightarrow TipoLista

(* OBSERVADORAS SELECTORAS *)

PARCIAL Primero : TipoLista \rightarrow TipoElemento

PARCIAL Resto : TipoLista \rightarrow TipoLista

2.1 Listas

(* OBSERVADORAS NO SELECTORAS *)

EsVacía : TipoLista → Booleano

Longitud : TipoLista → Natural

PARCIAL Ultimo : TipoLista → TipoElemento

Pertenece : TipoElemento × TipoLista → Booleano

(* CONSTRUCTORAS NO GENERADORAS *)

Concatenar : TipoLista × TipoLista → TipoLista

BorrarElemento: TipoElemento × TipoLista → TipoLista

InsertarFinal: TipoElemento × TipoLista → TipoLista

VARIABLES

lista, lista2 : TipoLista;

elemento, elem : TipoElemento;

ECUACIONES DE DEFINITUD

DEF(Primero (Construir (elemento, lista)))

DEF(Resto (Construir (elemento, lista)))

DEF(Ultimo (Construir (elemento, lista)))

2.1 Listas

ECUACIONES

(* OBSERVADORAS SELECTORAS *)

Primero (Construir (elemento, lista)) = elemento

Resto (Construir (elemento, lista)) = lista

(* OBSERVADORAS NO SELECTORAS *)

EsVacia (CrearVacia) = CIERTO

EsVacia (Construir (elemento, lista)) = FALSO

Longitud (CrearVacia) = 0

Longitud (Construir (elemento, lista)) = 1 + Longitud (lista)

Ultimo (Construir (elemento, lista)) = **SI** EsVacia (lista) →

Elemento

| Ultimo (lista)

Pertenece (elem, CrearVacia) = FALSO

Pertenece (elem, Construir (elemento, lista)) =

(elem = elemento) Ó Pertenece (elem, lista)

2.1 Listas

ECUACIONES (Cont.)

(* CONSTRUCTORAS NO GENERADORAS *)

Concatenar (CrearVacia, lista) = lista

Concatenar (Construir (elemento, lista), lista2) =

Construir (elemento, Concatenar (lista, lista2))

BorrarElemento (elem, CrearVacia) = CrearVacia

BorrarElemento (elem, Construir (elemento, lista)) = **SI** elem = elemento →

Lista

| Construir (elemento, BorrarElemento

(elem, lista))

InsertarFinal (elem, CrearVacia) = Construir (elem, CrearVacia)

InsertarFinal (elem, Construir (elemento, lista)) =

Construir (elemento, InsertarFinal (elem, lista))

FIN ESPECIFICACION

2.1 Listas

- Dos alternativas para implementar la lista: mediante arrays (lista estática) o mediante punteros (lista dinámica).
- Las 2 alternativas son posibles y tienen diferentes ventajas e inconvenientes.
- Realización estática
 - Ventajas: acceso directo
 - Inconvenientes: inserción o eliminación en posiciones concretas respetando el orden

2.1 Listas

- Realización dinámica
 - Ventajas: la inserción y la eliminación de nodos
 - Inconvenientes: acceso (secuencial)
- La forma de acceso especificada (`primero y resto`) deriva de la implementación habitual basada en punteros.

- Para acceder al tercer elemento: `lista = [1, 3, 5, -8, 4]`

`Primero (Resto (Resto (lista))) = 5`

2.1 Listas

Dentro de todas las alternativas podemos distinguir:

- Vector almacén y elemento `numero_de_elementos`
- Vector de nodos (`info + sig`) simulando una lista enlazada
- Lista enlazada simple
- Lista ordenada
- Lista enlazada con inserción al final
- Lista circular
- Lista circular con componente cabecera
- Lista doblemente enlazada
- Lista doblemente enlazada con puntero al principio y al final

2.1 Listas

- El TAD TipoListaOrdenada se puede derivar del TAD anterior.
- Gran parte del comportamiento es el mismo excepto que el usuario no podrá utilizar aquellas operaciones constructoras de listas que puedan alterar el orden
 - Construir
 - InsertarFinal
 - Concatenar

2.1 Listas

ESPECIFICACION ListaOrdenada

USA Listas

TIPO TipoListaOrdenada

OPERACIONES

(* OPERACIONES CONSTRUCTORAS NO GENERADORAS *)

InsertarOrd: TipoElemento x TipoListaOrdenada → TipoListaOrdenada

PARAMETROS GENERICOS

OPERACIÓN Menor: TipoElemento x TipoElemento → TipoElemento

FIN PARAMETROS GENERICOS

Mezclar: TipoListaOrdenada x TipoListaOrdenada → TipoListaOrdenada

VARIABLES:

e, e1: TipoElemento

lista, lista1: TipoListaOrdenada

2.1 Listas

ECUACIONES

(* operaciones constructoras no generadoras *)

InsertarOrd (e, CrearVacía) = Construir (e, CrearVacía)

InsertarOrd (e, Construir (e1, lista)) = **SI** Menor (e1, e) →
Construir (e1, InsertarOrd (e, lista))
|
Construir (e, Construir (e1, lista))

Mezclar (CrearVacía, lista) = lista

Mezclar (Construir (elem1, lista), lista1) =
InsertarOrd(elem1, Mezclar (lista, lista1))

FIN ESPECIFICACION