Parte I: Programación en Ada

UNIVERSIDAD DE CANTABRIA

1. Introducción a los computadores y su programación
2. Elementos básicos del lenguaje
3. Modularidad y programación orientada a objetos
4. *Estructuras de datos dinámicas*
5. Tratamiento de errores
6. Abstracción de tipos mediante unidades genéricas
7. Entrada/salida con ficheros
8. Herencia y polimorfismo
9. Programación concurrente y de tiempo real

4.1. Relaciones entre datos

UNIVERSIDAD DE CANTABRIA

En muchas estructuras de datos es preciso establecer *relaciones* o *referencias* entre diferentes datos.

- **ahorran espacio al no repetir datos**
- **evitan inconsistencias**

Si los datos están en un array, se pueden utilizar *cursores*

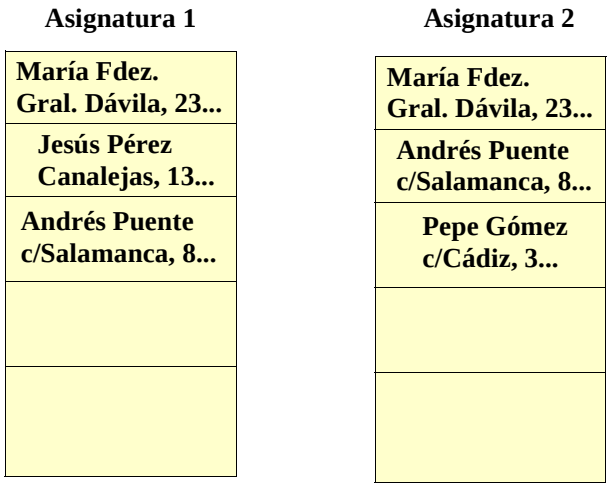
- **el cursor es un entero que indica el número de la casilla del array donde está el dato**

Si los datos no están en un array deben utilizarse *punteros* (si los soporta el lenguaje de programación)

- **son datos especiales que sirven para apuntar o referirse a otros datos**

Ejemplo: listas de alumnos de asignaturas

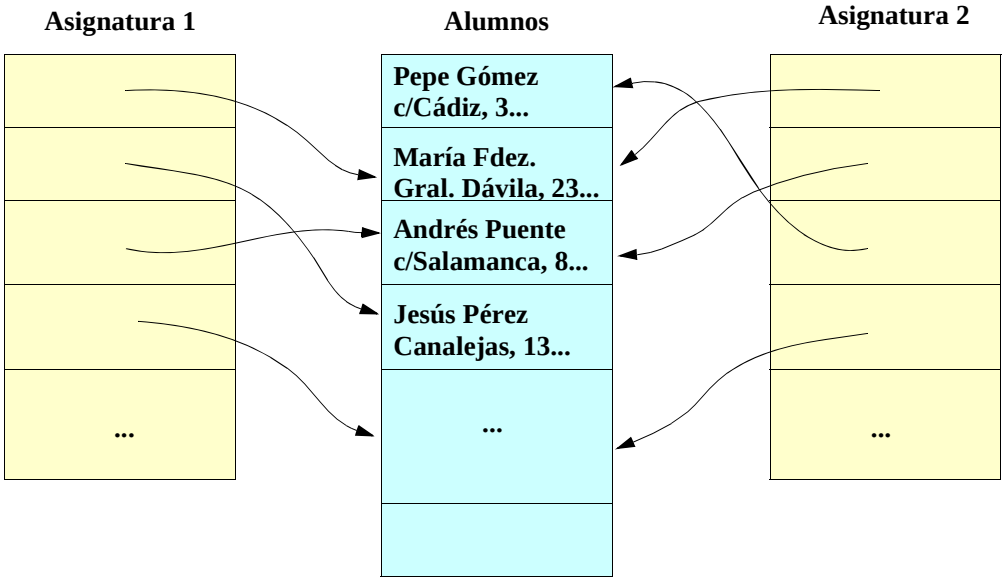
UNIVERSIDAD DE CANTABRIA



¡Hay datos repetidos!

Alternativa: Referencias entre datos

UNIVERSIDAD DE CANTABRIA



4.2. Punteros

UNIVERSIDAD DE CANTABRIA

Los punteros o tipos acceso proporcionan acceso a otros objetos de datos

Hasta ahora el acceso era sólo por el nombre o un cursor

- ahora el puntero es más flexible

Los objetos apuntados por un puntero se pueden crear o destruir de forma independiente de la estructura de bloques

Declaración:

```
type nombre is access tipo;
```

Ejemplo:

```
type P_Integer is access Integer;
P1, P2 : P_Integer;
```


Punteros (cont.)

UNIVERSIDAD DE CANTABRIA

Hay un valor predefinido, `null`, que es el valor por omisión, y no se refiere a ningún dato

Creación dinámica de objetos:

```
P1:= new Integer;
P2:= new Integer'(37);
```

Uso del puntero: por el nombre

```
P1:=P2; -- copia sólo la referencia
```

Uso del valor al que apunta el puntero:

- completo: `nombre_puntero.all`
- campo de un registro: `nombre_puntero.campo`
- casilla de un array: `nombre_puntero(índice)`

Ejemplo

UNIVERSIDAD DE CANTABRIA

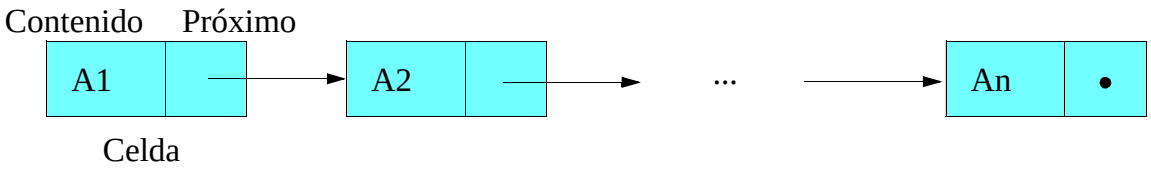
```
type Coord is record
   X,Y : Float;
end record;

type Vector is array(1..3) of Float;

type P_Coord is access Coord;
type P_Vector is access Vector;

PV1, PV2 : P_Vector;
PC1, PC2 : P_Coord;

...
PV1:=new Vector;
PV2:=new Vector'(1.0,2.0,3.0);
PV1.all:=PV2.all;    -- copia el valor completo
PV1(3):=3.0*PV2(2); -- usa casillas individuales del array

PC1:=new Coord'(2.0,3.0);
PC1.Y:=9.0;          -- usa un campo individual del registro
```


4.3. Estructuras de datos dinámicas

UNIVERSIDAD DE CANTABRIA

Son útiles cuando se usan para crear estructuras de datos con relaciones entre objetos.

Por ejemplo, una lista enlazada

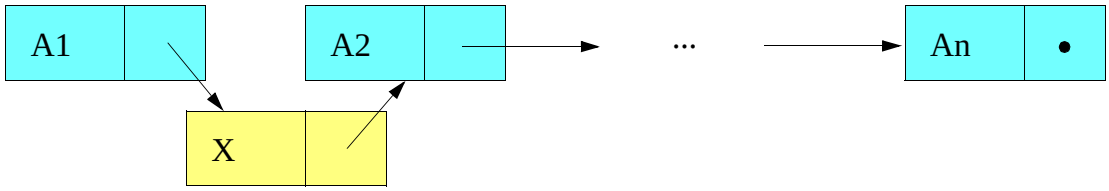


- **cada elemento tiene un contenido y un puntero al próximo**
- **el último tiene un puntero "próximo" nulo**

Flexibilidad de la estructura de datos dinámica

UNIVERSIDAD DE CANTABRIA

Se pueden insertar nuevos elementos en la posición deseada, eficientemente:



Diferencias con el array:

- **los arrays tienen tamaño fijo: ocupan lo mismo, incluso medio vacíos**
- **con estructuras de datos dinámicas se gasta sólo lo preciso**
- **pero se necesita espacio para guardar los punteros**

Definición de estructuras de datos dinámicas en Ada

UNIVERSIDAD DE CANTABRIA

Hay una dependencia circular entre el tipo puntero y el tipo celda

- **se resuelve con una *declaración incompleta de tipo***

Ejemplo

```
type Celda; -- declaración incompleta de tipo
type P_Celda is access Celda;
type Celda is record
   Contenido : ...; -- poner el tipo deseado
   Proximo : P_Celda;
end record;
```

Ejemplo de creación de una lista enlazada

UNIVERSIDAD DE CANTABRIA

```
Lista : P_Celda;

Lista:=new Celda'(1,null); -- suponemos el contenido entero
Lista.proximo:=new Celda'(2,null);
Lista.proximo.proximo:=new Celda'(3,null);
```

Para evitar extender la notación punto indefinidamente, podemos utilizar un puntero auxiliar:

```
P : P_Celda;

P:=Lista.proximo.proximo;
P.proximo:=new Celda'(4,null);
P:=P.proximo; -- para seguir utilizando P con la siguiente celda
```


Estructuras de datos dinámicas (cont.)

UNIVERSIDAD DE CANTABRIA

Veremos abundantes ejemplos en el capítulo de los tipos abstractos de datos

4.4 Punteros a objetos estáticos

UNIVERSIDAD DE CANTABRIA

Hasta ahora, todos los punteros lo eran a objetos creados dinámicamente (con `new`)

Es posible crear punteros a objetos (variables, constantes o subprogramas) estáticos. Ejemplos:

```
type Acceso_Entero is access all Integer;
type Acceso_Cte is access constant Integer;
```

Para poder crear un puntero a una variable estática, ésta se debe haber creado con la mención "`aliased`":

```
Num : aliased Integer:=5;
```

Para crear el puntero se utiliza el atributo `'Access`:

```
P : Acceso_Entero := Num'Access;
```

La principal utilidad es para llamar a funciones C

Punteros a objetos estáticos (cont.)

UNIVERSIDAD DE CANTABRIA

Este atributo sólo se puede usar si el tipo puntero está declarado en el mismo nivel de accesibilidad que el objeto, o en uno más profundo

- motivo: impedir tener un puntero a un objeto que ya no existe
- lo mejor es tener el objeto y el tipo puntero declarados en el mismo sitio

En caso de que sea imprescindible romper esta norma, se puede usar el atributo `'Unchecked_Access`, que también proporciona el puntero pero sin restricciones

- el uso de este atributo es peligroso