Parte I: Elementos del lenguaje 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

Asignatura 1

María Fdez. Gral. Dávila, 23...
Jesús Pérez Canalejas, 13...
Andrés Puente c/Salamanca, 8...

Asignatura 2

María Fdez. Gral. Dávila, 23...
Andrés Puente c/Salamanca, 8...
Pepe Gómez c/Cádiz, 3...

¡Hay datos repetidos!

Notas:

UNIVERSIDAD DE CANTABRIA

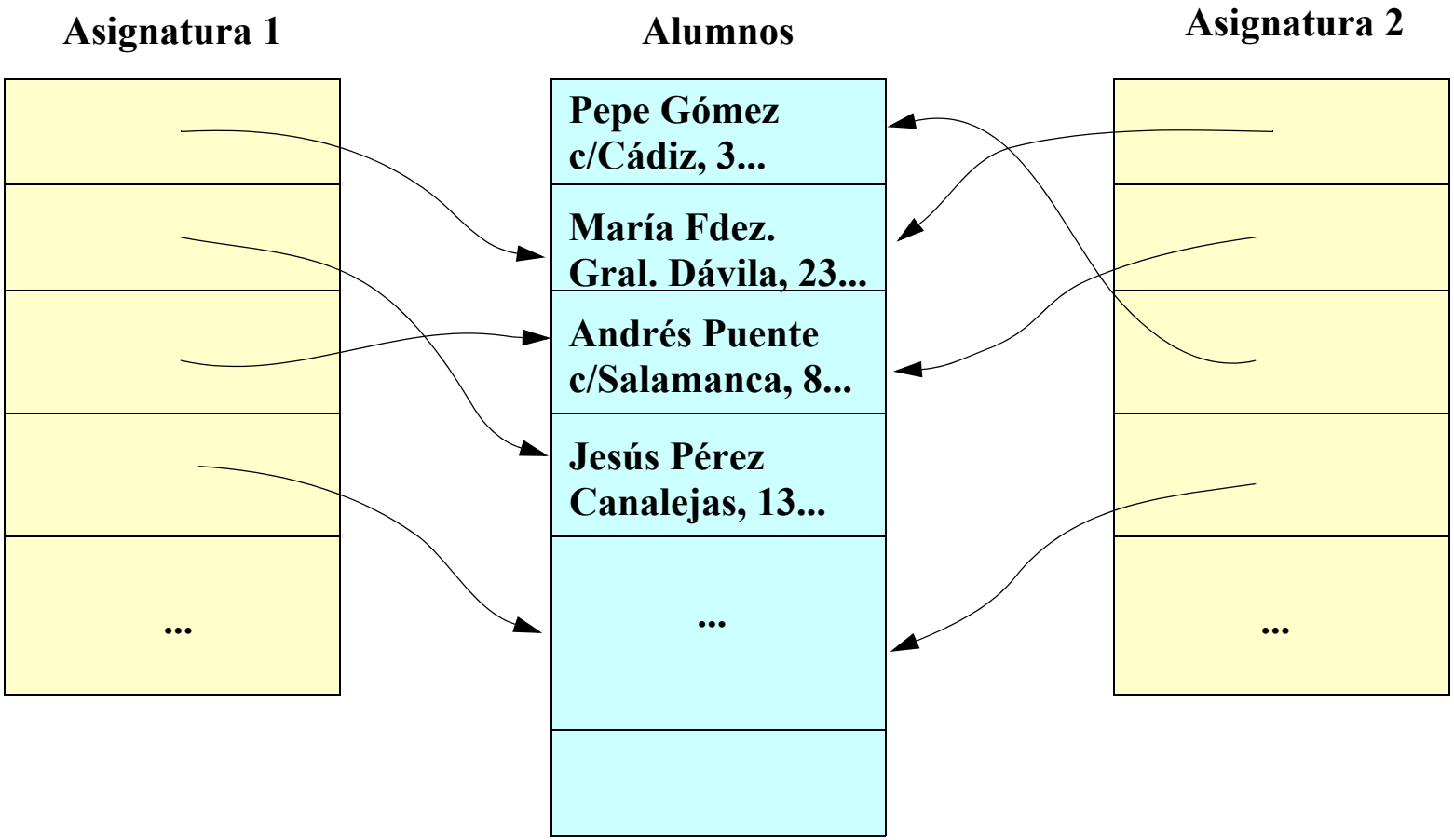
Los punteros son datos especiales que contienen direcciones de memoria de otros datos, y por lo tanto sirven para acceder a estos últimos. A través de un puntero se puede acceder a datos:

- Estáticos: cuando hacen referencia (contienen su dirección o apuntan) a datos que se han creado en la parte declarativa de un segmento de programa.
- Dinámicos: cuando hacen referencia (contienen su dirección o apuntan) a datos que no se han declarado, sino que se han creado en la ejecución del programa. En estos casos el propio puntero se utiliza normalmente en la creación de estos datos dinámicos.

En el ejemplo de la página anterior podemos ver dos listas con los datos de alumnos matriculados en dos asignaturas en las que observamos que hay datos repetidos. Estas listas se podrían implementar con sendos arrays en los que obligatoriamente se debe guardar la información completa de cada alumno, por lo que si un alumno cursa las dos asignaturas tendrá su información repetida.

En la página siguiente podemos ver cómo se pueden usar punteros a la información de los alumnos como elementos de las listas. La información de todos los alumnos podría estar en una estructura de datos en la que la información de un alumno es única, y las listas de las asignaturas en las que se matriculan contienen sólo punteros a los datos de cada alumno.

Alternativa: referencias entre datos



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

```
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
```

Notas:

UNIVERSIDAD DE CANTABRIA

En Ada los tipos puntero se definen como `access` a algún tipo de dato.

La definición de una variable de puntero no crea el dato al que apunta, sino que se inicializa a `null`. El puntero se puede utilizar para acceder a un dato existente o para apuntar a un nuevo dato; en este último caso será necesario crearlo con `new`.

La asignación entre dos punteros hace que los dos apunten al mismo objeto de datos, no se hace una réplica del dato.

Los punteros a registros o arrays siguen la misma notación que los registros o arrays convencionales para acceder a sus elementos (el punto para el campo de un registro, el paréntesis con el índice para el elemento de un array).

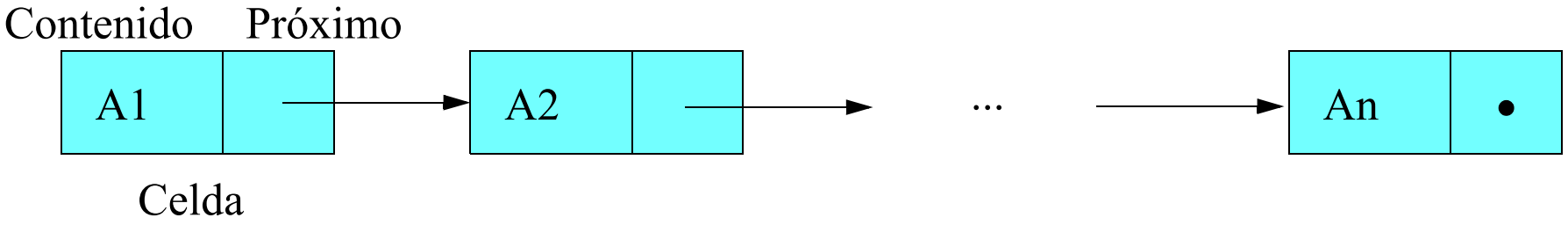
En los ejemplos se puede observar cómo se puede crear un nuevo dato asignándole un valor inicial o sin asignárselo.

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

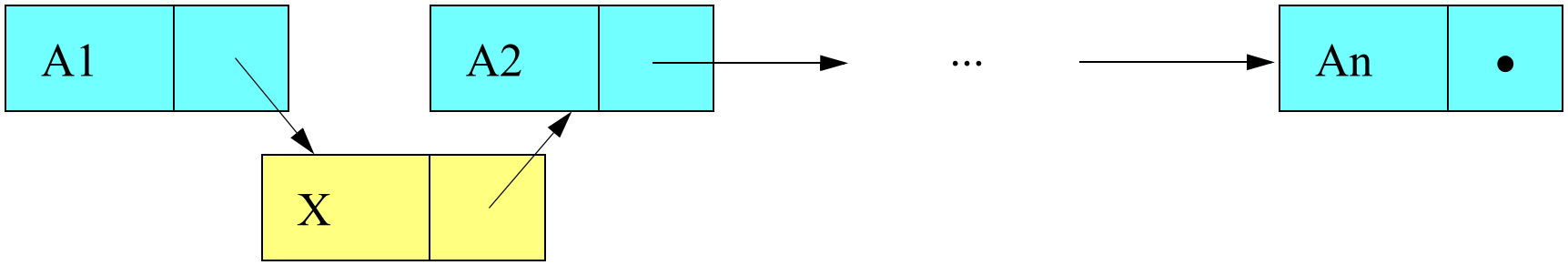


- **cada elemento tiene un contenido y un puntero al próximo**
- **el último tiene un puntero "próximo" nulo**

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

Notas:

UNIVERSIDAD DE CANTABRIA

Los punteros son especialmente útiles para crear estructuras de datos dinámicas, en las que el tamaño no está predeterminado y pueden incrementar o decrementar su tamaño en tiempo de ejecución.

En el ejemplo anterior se muestra una lista enlazada en la que cada elemento contiene además de la información del dato que guarda un puntero al siguiente elemento. Así, la lista es en realidad un puntero al primer elemento, y el último elemento de la lista es aquel cuyo elemento siguiente es `null`.

La inserción de un nuevo elemento únicamente requiere la creación del elemento y el enlace de los punteros para que forme parte de la lista en el lugar adecuado.

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
```

Notas:

UNIVERSIDAD DE CANTABRIA

Para la creación de tipos de datos que contienen un puntero a sí mismos se utiliza la definición incompleta de tipo. Primero se define un tipo sólo con su nombre, después el tipo puntero que lo apunta, y finalmente el tipo completo que puede ya contener el puntero previamente definido (ver el ejemplo de `Celda`).

Normalmente, se utilizan punteros auxiliares en los recorridos de la lista. Por ejemplo:

```
P : P_Celda;
...
P := Lista;
while P /= null
loop
   Haz el trabajo;
   P := P.proximo;
end loop;
```

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