Estructuras de Datos y Algoritmos

Datos puntero y estructuras dinámicas de datos

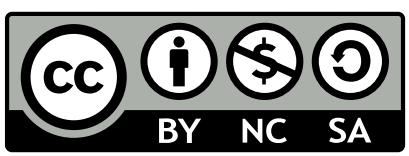
Lección 7

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

Universidad Zaragoza

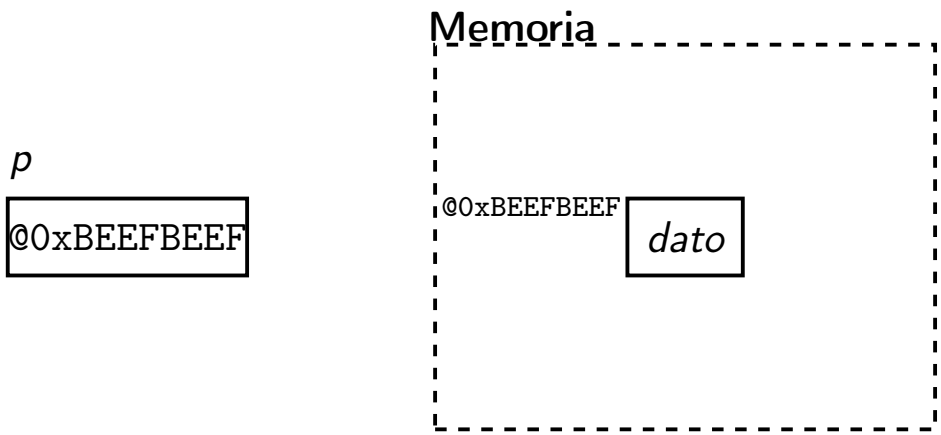
Índice

Universidad Zaragoza

Datos puntero

Puntero (o *referencia*)

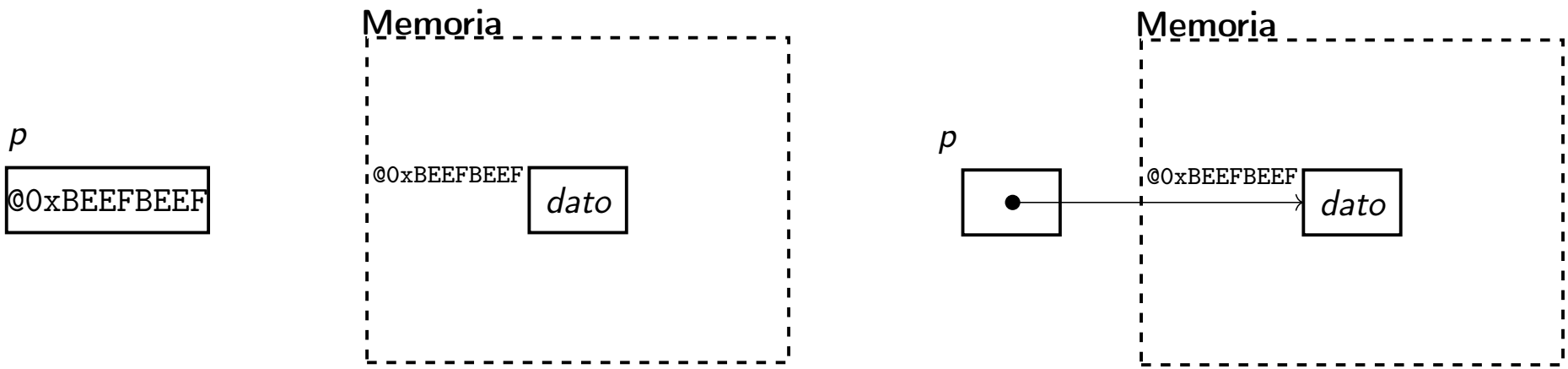
- Dato cuyo valor es la **dirección en memoria de otro determinado dato**



Datos puntero

Puntero (o *referencia*)

- Dato cuyo valor es la **dirección en memoria de otro determinado dato**
- Puede verse como un *apuntador* del dato. Gráficamente:



Datos puntero

- La dirección almacenada va a ser transparente al programador
  - Sólo le interesa conocer a qué valor referencia (no *dónde* está)
- Especializados para trabajar como referencia de datos de un tipo determinado
- Algorítmicamente tenemos un mecanismo constructor de datos puntero:

```
tipos tx = ...
          ...
          tp_punt_a_tx = ↑tx
...
variables
    p, q: tp_punt_a_tx
```

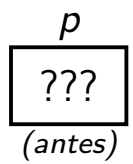
Índice

Universidad Zaragoza

Datos dinámicos

- **Creados dinámicamente** (es decir, durante la ejecución de un programa)
  - Se pueden crear y destruir en cualquier momento de la ejecución
  - Por contra, los **datos estáticos** son aquellos que se les asigna memoria al comenzar la ejecución y se libera la memoria asignada al finalizar la ejecución
- Referenciados por un puntero

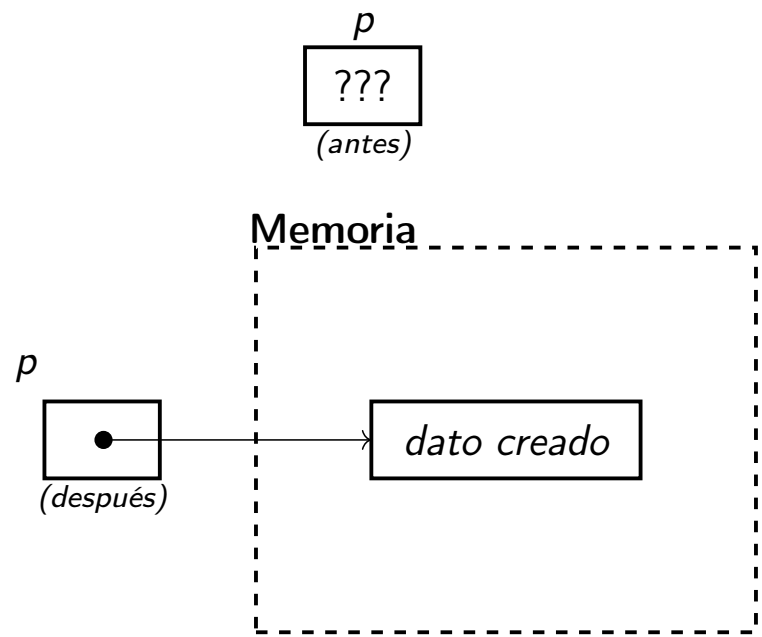
```
tipos tx = ...
          ...
          tp_punt_a_tx = ↑tx
...
variables
     p: tp_punt_a_tx
...
implementación
...
     nuevoDato(p)
```



Datos dinámicos

- **Creados dinámicamente** (es decir, durante la ejecución de un programa)
  - Se pueden crear y destruir en cualquier momento de la ejecución
  - Por contra, los **datos estáticos** son aquellos que se les asigna memoria al comenzar la ejecución y se libera la memoria asignada al finalizar la ejecución
- Referenciados por un puntero

```
tipos tx = ...
          ...
          tp_punt_a_tx = ↑tx
...
variables
     p: tp_punt_a_tx
...
implementación
...
     nuevoDato(p)
```

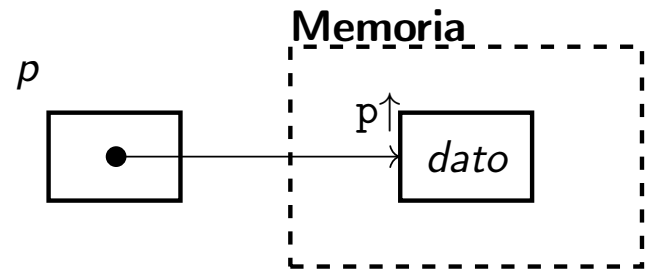


Datos dinámicos

- **NO TIENEN** ningún identificador asociado como nombre
- Accederemos al dato referenciado por un puntero `p` mediante `p↑`

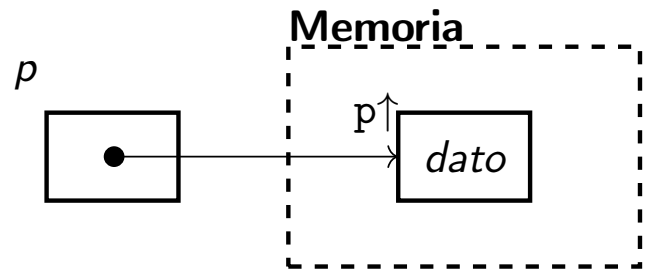
Datos dinámicos

- **NO TIENEN** ningún identificador asociado como nombre
- Accederemos al dato referenciado por un puntero `p` mediante `p↑`
  - *Significa "dato apuntado por el puntero* `p`*"*



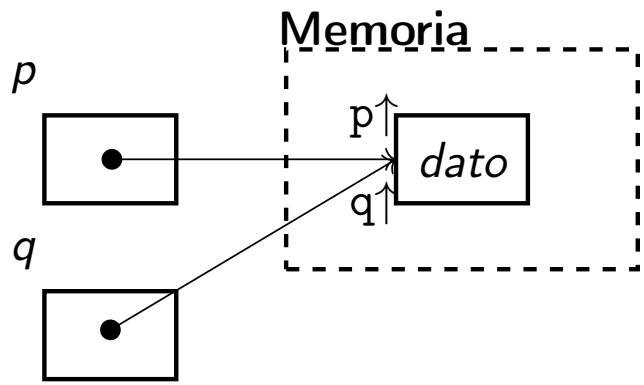
Datos dinámicos

- **NO TIENEN** ningún identificador asociado como nombre
- Accederemos al dato referenciado por un puntero `p` mediante `p↑`
  - *Significa "dato apuntado por el puntero* `p`*"*



- **La asignación entre punteros de datos del mismo tipo es válida**

`q := p` ⇒



Datos dinámicos

- **El dato creado puede ser accedido de dos formas**: `p↑` `y` `q↑`

Datos dinámicos

- **El dato creado puede ser accedido de dos formas**: `p↑` y `q↑`

- **Valor constante** `nil` para indicar que el puntero no apunta a ningún dato: (también llamado *puntero nulo*)

```
p := nil
```


Datos dinámicos

- **El dato creado puede ser accedido de dos formas**: `p↑` y `q↑`

- **Valor constante** `nil` para indicar que el puntero no apunta a ningún dato: (también llamado *puntero nulo*)

```
p := nil
```

- **Operación de destrucción de zona de memoria asignada** al dato creado: (también llamado *liberación* de memoria asignada)

```
disponer(p)
```

  - Acción opuesta a la operación `nuevoDato`

Universidad Zaragoza

Datos dinámicos

Recolección de basura

- **Gestión manual o automática de la memoria dinámica**
  - Gestión manual mediante operación `disponer` (e.g., C/C++)
  - Gestión automática (e.g., Python, Java, ...)

Universidad Zaragoza

Datos dinámicos

Recolección de basura

- **Gestión manual o automática de la memoria dinámica**
  - Gestión manual mediante operación `disponer` (e.g., C/C++)
  - Gestión automática (e.g., Python, Java, ...)

Otras operaciones disponibles con punteros

- Operador relacional de **igualdad**: $=$
- Operador relacional de **desigualdad**: $\neq$

Universidad Zaragoza

Índice

Universidad Zaragoza

Estructuras de datos recursivas

- **En su definición incluyen un dato de su mismo tipo**

```
tipo cadena =   registro
                    esVacía: booleano;
                    { y por si no es vacía ... }
                    dato: carácter; {el 1º de la cadena}
                    resto: cadena; { ¡Ojo! este tipo de
                        representación NO lo permitimos }
                freg
```

Esto tiene un problema...

¿Cómo sabe el compilador (o intérprete) cuánta memoria asignar a una variable estática de este tipo?

Universidad Zaragoza

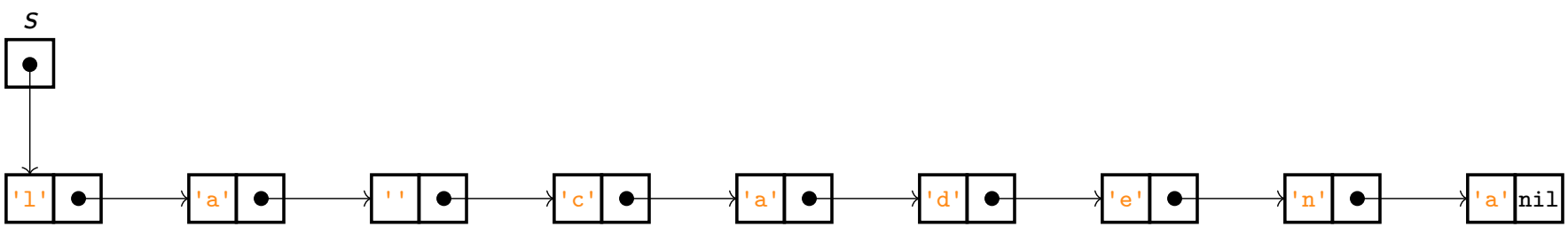
Estructuras de datos recursivas

Definición

- **Utilizando punteros** a diferentes partes de la estructura
- El tamaño y forma de la estructura puede variar en ejecución (dinámica)

```
tipos  cadena = ↑cad;
       cad    = registro
                    dato: carácter;
                    resto: cadena;
                freg
```

Supóngase una variable `s` *definida como* `s: cadena` *e inicializada con* `"la cadena"`



Estructuras de datos recursivas

Antipatrón: duplicar vs. enlazar

- Cuando una misma entidad aparece en varias estructuras (p.ej., un adulto responsable de varios niños), ¿copiamos sus datos en cada sitio?

× Así no

Copiar los datos de la entidad en cada estructura que la usa:

- memoria duplicada
- modificar ⇒ actualizar *todas* las copias
- copias inconsistentes = errores

✓ Así sí

Almacenar la entidad **una sola vez**; el resto de estructuras guardan **punteros** a ella:

- una única fuente de verdad
- modificar ⇒ un solo sitio
- coste extra: solo un puntero

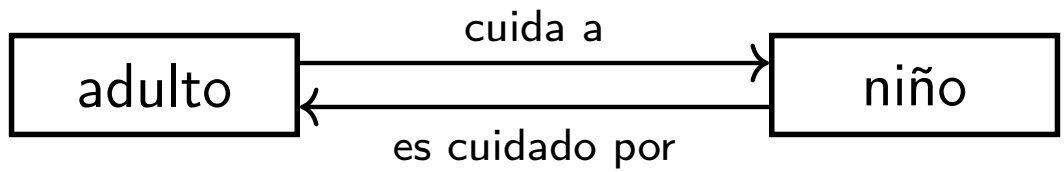
Regla: la información compartida se **enlaza**, no se **copia**

Universidad Zaragoza

Estructuras de datos recursivas

Punteros recíprocos: relaciones en ambos sentidos

- Si el sistema pregunta en los dos sentidos (*"¿qué niños cuida este adulto?"* y *"¿quién cuida a este niño?"*), la relación necesita **punteros recíprocos**: de A a B *y* de B a A
- Al **insertar** o **borrar** hay que mantener *los dos* enlaces (y los contadores o cardinales asociados): olvidar uno de ellos es el error típico de la familia F2
- Lo trabajaremos en la hoja de problemas 2 (guardería) y reaparecerá en las estructuras interrelacionadas del examen



Índice

Universidad Zaragoza

Implementación en C++

Datos puntero

Pseudocódigo:

```
tipo Persona = registro
                    nombre : cadena;
                    edad   : entero;
                freg
        Puntero_Persona = ↑Persona;
variables
    p, q : Puntero_Persona;


    nuevoDato(p);
    p↑.nombre := "Pepe";
    p↑.edad   := 19;

    disponer(p);
    q := nil;
```

C++:

```
struct Persona {
    string nombre;
    int edad;
};

Persona* p,  q;


p = new Persona;
p -> nombre = "Pepe";
// equivalente a:
// (*p).nombre = "Pepe"
p -> edad   = 19;

delete p;

q = nullptr;
```


Implementación en C++

Estructuras de datos recursivas

Pseudocódigo:

```
tipo cadena  = registro
                   dato: carácter;
                   resto:  ↑cadena;
               freg
```

C++:

```
struct Cadena {
    char dato;
    Cadena* resto;
};
```


Trabajo encargado

Leer lección 7 de los apuntes. Empezar Práctica 2

Estructuras de Datos y Algoritmos

Datos puntero y estructuras dinámicas de datos

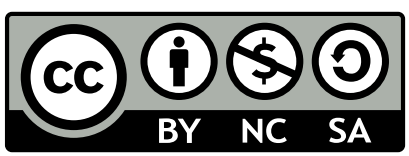
Lección 7

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