

Estructuras de Datos y Algoritmos

Datos puntero y estructuras dinámicas de datos

LECCIÓN 7

© All wrongs reversed – bajo licencia CC-BY-NC-SA 4.0



Universidad
Zaragoza

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

Curso 2024/2025

Grado en Ingeniería Informática
UNIVERSIDAD DE ZARAGOZA

Aula 0.04, Edificio Agustín de Betancourt



Adaptadas de diapositivas de Javier Campos

Índice

- 1 Datos puntero
- 2 Datos dinámicos
- 3 Estructuras de datos recursivas
- 4 Implementación en C++

Índice

1 Datos puntero

2 Datos dinámicos

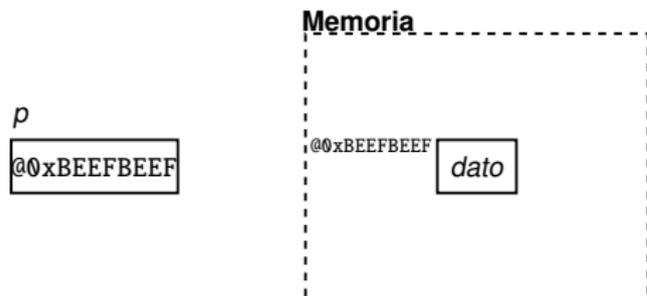
3 Estructuras de datos recursivas

4 Implementación en C++

Datos puntero

Puntero (o *referencia*)

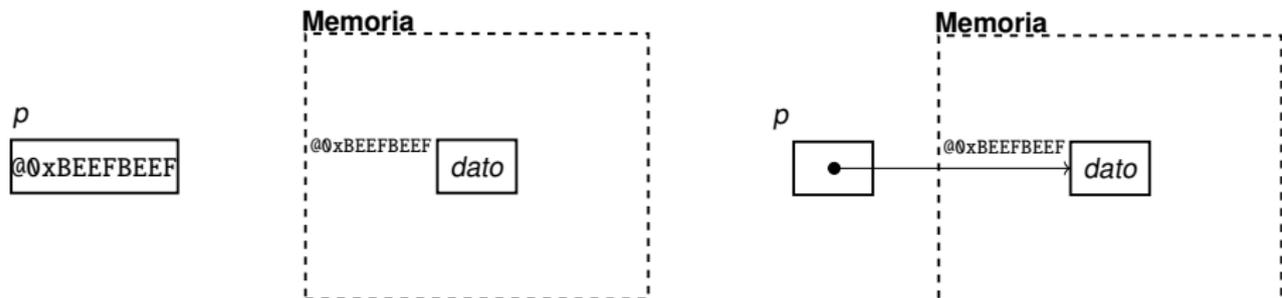
- Dato cuyo valor es la **dirección en memoria de otro determinado dato**



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

1 Datos puntero

2 Datos dinámicos

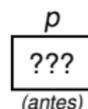
3 Estructuras de datos recursivas

4 Implementación en C++

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

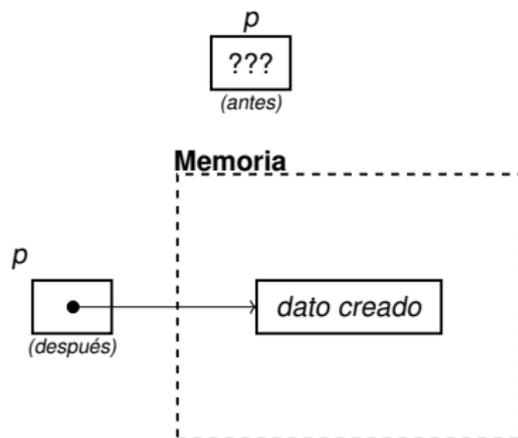
- **Creados dinámicamente** (es decir, durante la ejecución de un programa)
 - Se pueden crear y destruir en cualquier momento de la ejecución
 - Por contra, los **datos estáticos** son aquellos que se les asigna memoria al comenzar la ejecución y se libera la memoria asignada al finalizar la ejecución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

```
tipos tx = ...  
      ...  
      tp_punt_a_tx = ↑tx
```

```
...  
variables  
  p: tp_punt_a_tx
```

```
...  
implementación
```

```
...  
nuevoDato(p)
```

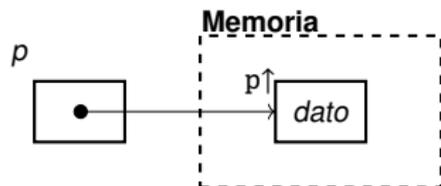


Datos dinámicos

- **NO TIENEN** ningún identificador asociado como nombre
- Accederemos al dato referenciado por un puntero p mediante $p\uparrow$

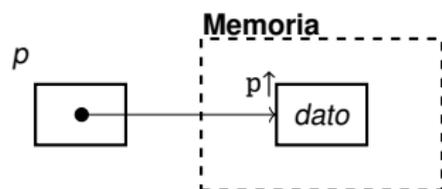
Datos dinámicos

- **NO TIENEN** ningún identificador asociado como nombre
- Accederemos al dato referenciado por un puntero p mediante $p \uparrow$
 - Significa “dato apuntado por el puntero p ”

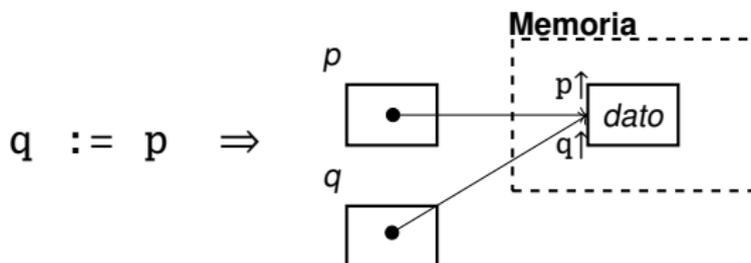


Datos dinámicos

- **NO TIENEN** ningún identificador asociado como nombre
- Accederemos al dato referenciado por un puntero p mediante $p \uparrow$
 - Significa “dato apuntado por el puntero p ”



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



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: $p \uparrow$ y $q \uparrow$
- **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:** $p \uparrow$ y $q \uparrow$
- **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`

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, ...)

Recolección de basura

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

Otras operaciones disponibles con punteros

- Operador relacional de **igualdad**: `=`
- Operador relacional de **desigualdad**: `≠`

Índice

- 1 Datos puntero
- 2 Datos dinámicos
- 3 Estructuras de datos recursivas**
- 4 Implementación en C++

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?

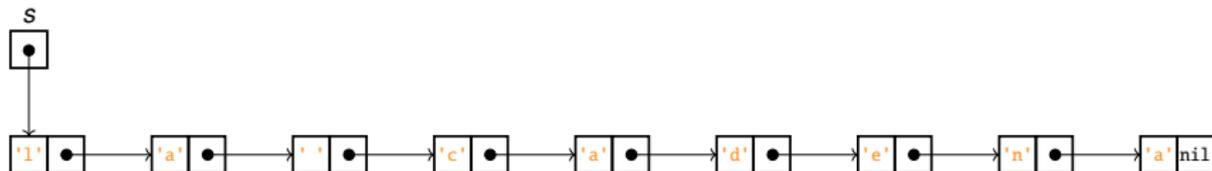
Estructuras de datos recursivas

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"



Índice

- 1 Datos puntero
- 2 Datos dinámicos
- 3 Estructuras de datos recursivas
- 4 Implementación en C++**

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;
};
```

Estructuras de Datos y Algoritmos

Datos puntero y estructuras dinámicas de datos

LECCIÓN 7

© All wrongs reversed – bajo licencia CC-BY-NC-SA 4.0



Universidad
Zaragoza

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

Curso 2024/2025

Grado en Ingeniería Informática

UNIVERSIDAD DE ZARAGOZA

Aula 0.04, Edificio Agustín de Betancourt

