UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica etsinf

Tema 5. Tipos de Datos Lineales

Programación (PRG)

Jorge González Mollá

Departamento de Sistemas Informáticos y Computación

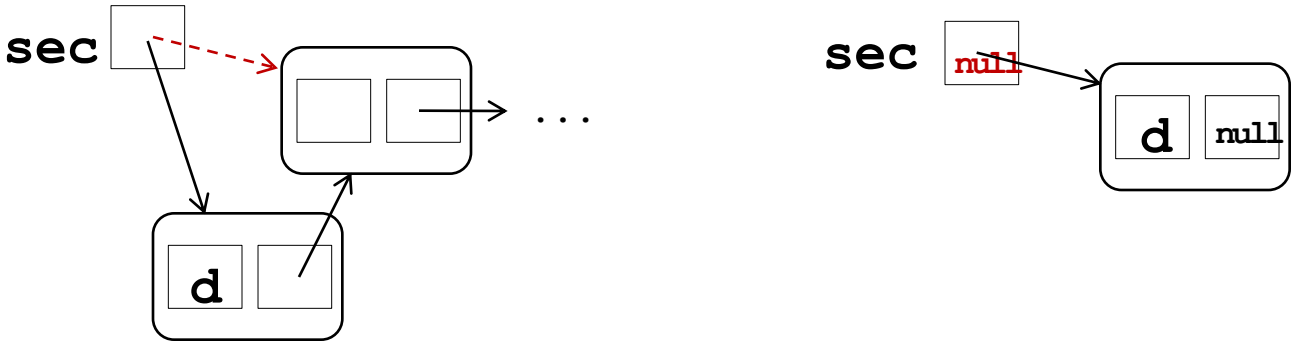
Índice

Inserción en cabeza

- Gracias a los enlaces, añadir un dato nuevo a una secuencia dada no implica ningún movimiento en memoria de los datos existentes.

- Según dónde se deba realizar la inserción, se pueden dar 2 casos:
  a) El nuevo nodo se inserta delante de toda la secuencia (incluye la situación de insertar sobre una secuencia vacía):

```
sec = new NodeInt(d, sec);
```

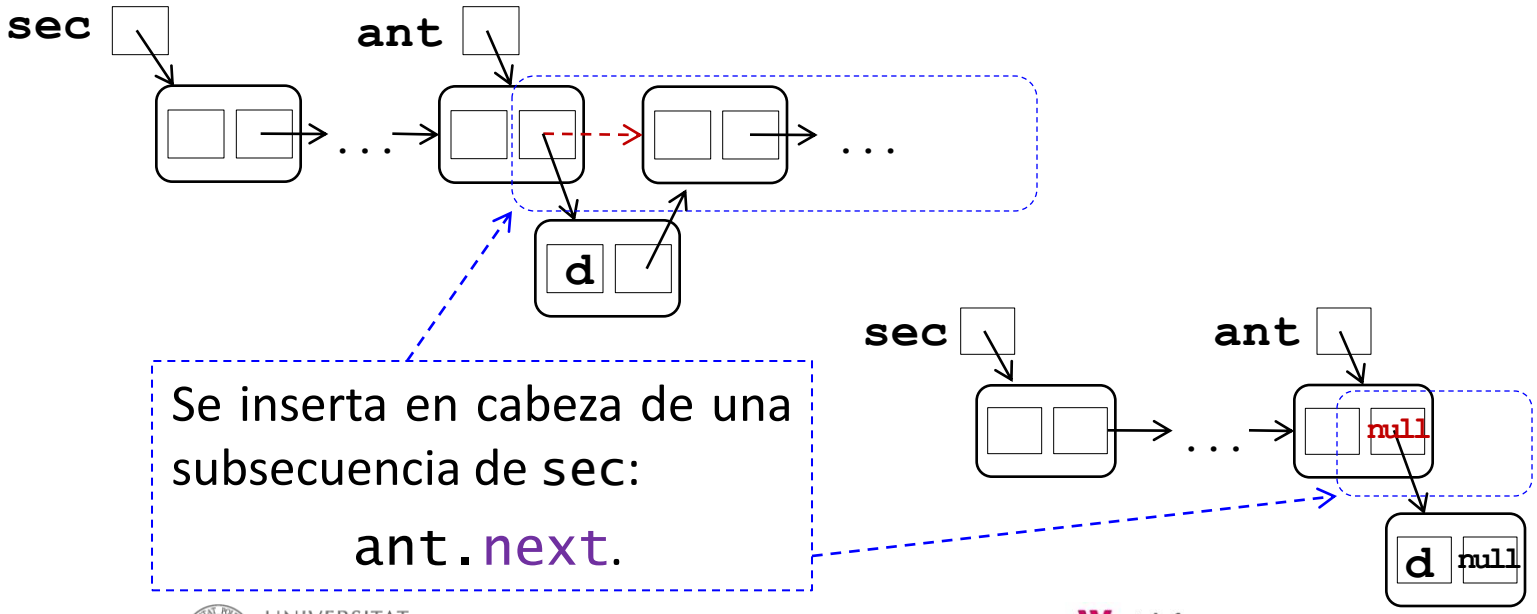


Inserción en cualquier otra posición

b) El nuevo nodo se inserta en una posición distinta de la cabeza, es decir, detrás de algún otro nodo existente de la secuencia (incluye la situación de insertar al final, detrás del último nodo):

```
ant.next = new NodeInt(d, ant.next);
```

donde `ant` es un enlace al nodo tras el que realizar la inserción.



Inserción en general

- **Ejemplo**. Dada una secuencia de un cierto número de nodos n, insertar el dato d en la posición i, siempre que $0 \leq i \leq n$.

```
if (i == 0) { sec = new NodeInt(d, sec); }
else {
        NodeInt ant = sec; int k = 0;
        while (ant != null && k < i - 1) {
              ant = ant.next; k++;
        }
         if (ant != null) // Éxito en la búsqueda
             ant.next = new NodeInt(d, ant.next);
}
```

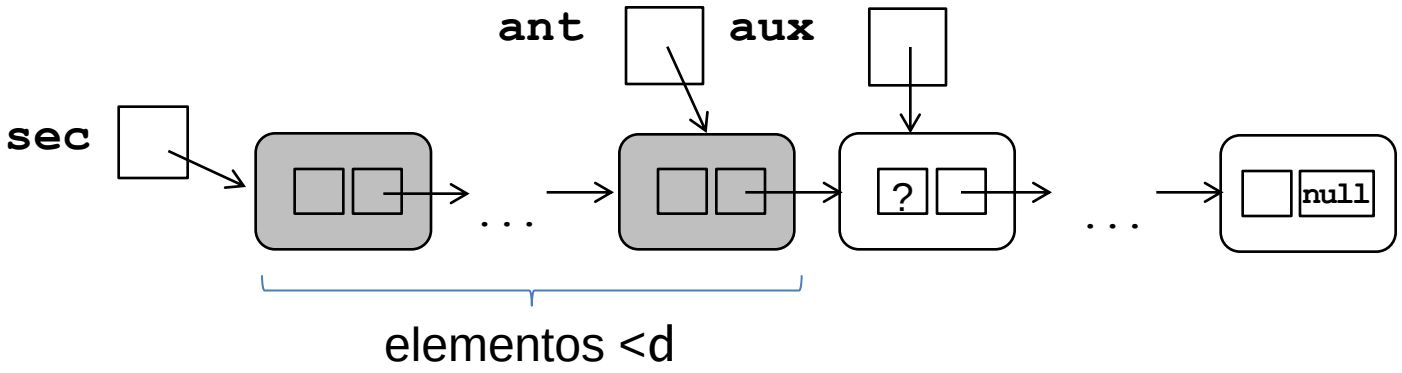
Casos:

a) i == 0: inserción en cabeza

b) i > 0: se busca el nodo i-1, y si existe, se inserta el nuevo nodo detrás

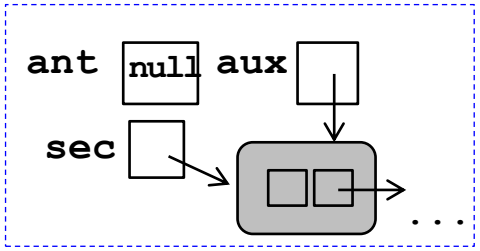
Inserción ordenada

- **Ejemplo**. Dada una secuencia cuyos datos están ordenados de menor a mayor, insertar un dato d manteniendo la ordenación.
  - Insertar delante de la subsecuencia de `sec` cuyos elementos ≥d.
  - Con la referencia `aux` se busca el primer nodo cuyo data sea ≥d. El enlace `ant` apuntará en todo momento al nodo anterior a `aux`.



Inserción ordenada

```
NodeInt aux = sec,
        ant = null; // el primer nodo no tiene
                    // anterior definido
  while (aux != null && aux.data < d) {
      ant = aux;
      aux = aux.next;
  }
  if (aux == sec) { sec = new NodeInt(d, sec); } // Caso a)
  else       { ant.next = new NodeInt(d, aux); } // Caso b)
```



Acabada la búsqueda:

Caso a): sec está vacía o todos sus datos son ≥d (inserción en cabeza).

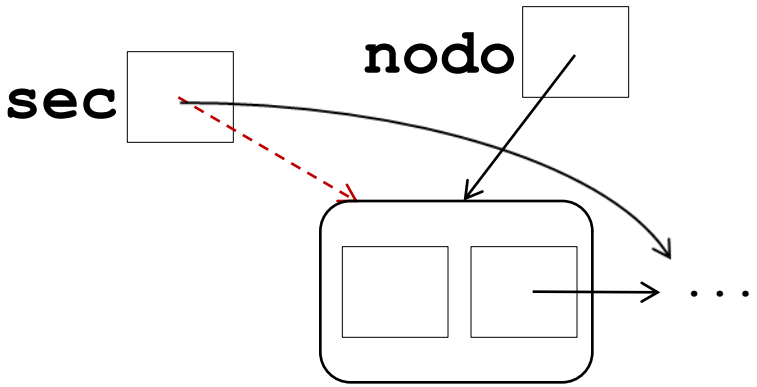
Caso b): hay datos menores que d (inserción en cualquier otra posición).

La inserción se realiza detrás del nodo ant después de todos los datos <d.

Borrado en cabeza

- Como al insertar, la eliminación de un dato de una secuencia dada se resuelve también sin ningún movimiento de los datos existentes.
- Según dónde se deba realizar el borrado, se pueden dar 2 casos:
  a) El nodo a borrar está en la primera posición de la secuencia.

```
sec = sec.next;
```

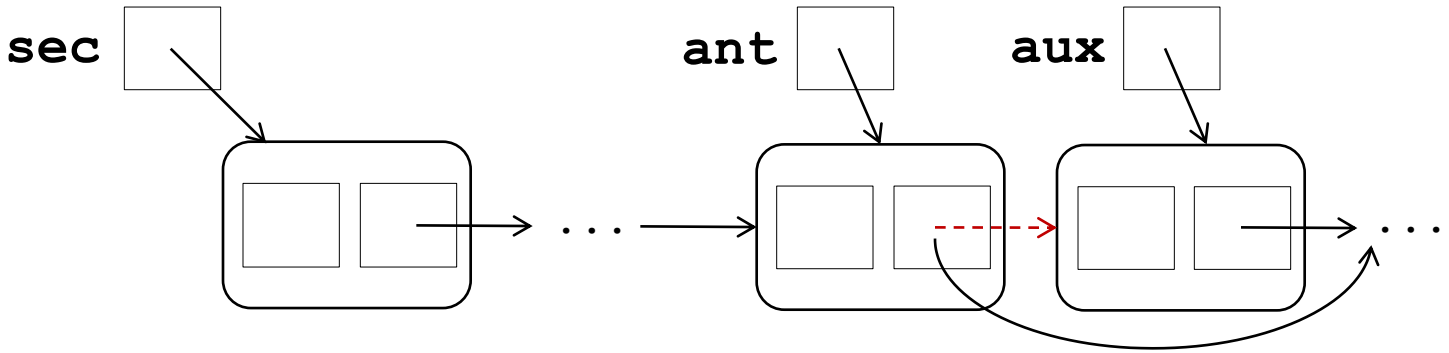


Si solo tenía 1 nodo, `sec` pasaría a ser `null` (secuencia vacía).

Borrado en cualquier otra posición

b) El nodo a eliminar no es el primero, luego tiene un anterior:

```
ant.next = aux.next;
```



donde `aux` es una referencia que apunta al nodo a eliminar, y `ant` es una referencia a su nodo inmediatamente anterior.

En el caso particular de que `aux` fuera el último nodo de `sec`, `ant.next` pasaría a valer `null` (`ant` estaría ahora el último).

Borrado en general

- **Ejemplo**. Eliminar, si existiere, la primera ocurrencia de un dato d. Si no existe, no se hace nada.

```
NodeInt aux = sec, ant = null;
while (aux != null && aux.data != d) {
    ant = aux;
    aux = aux.next;
}
if (aux != null) {          // Éxito en la búsqueda
  if (aux == sec) {            // Borrado en cabeza
      sec = sec.next;
  }
  else {   // Borrado en cualquier otra posición
      ant.next = aux.next;
  }
}
```

Inserción/Borrado (paso de parámetros)

- **Ejemplo**. Método que borra todos los datos de una secuencia que estén por debajo de un cierto `umbral`.

secuencia resultante del borrado

```
public static NodeInt borrar(Nodeint sec, int umbral){
    NodeInt aux = sec, ant = null;
    while (aux != null){
        if (aux.data < umbral) { // borrar el nodo aux
            if (aux == sec) { sec = sec.next; }
            else        { ant.next = aux.next; }
            aux = aux.next;
        }
        else { ant = aux;
               aux = aux.next;
        }
    }
    // El parámetro sec puede haber cambiado,
    return sec;
}
```