ESTRUCTURA DE DATOS

Ricardo Franco Ceballos

Departamento de Sistemas de Información

Facultad de Ingenierías

AGENDA DEL DÍA

- Estructura de datos: lista doble
- Operaciones addAfter() y addBefore()
- Algoritmos usando listas dobles

Estructura de datos

Clase DoubleNode

- Atributos
  - Data: dato u objeto almacenado en el nodo
  - Next: apuntador al siguiente nodo doble
  - Prev: apuntador al nodo doble previo

DoubleNode
-data:Object -next:DoubleNode -prev:DoubleNode
+DoubleNode() +DoubleNode(Object d) +setData(Object d) +setNext(DoubleNode n) +setPrev(DoubleNode p) +getData(): Object +getNext(): DoubleNode +getPrev(): DoubleNode

Estructura de datos

Clase DoubleNode

DoubleNode
-data:Object -next:DoubleNode -prev:DoubleNode
+DoubleNode() +DoubleNode(Object d) +setData(Object d) +setNext(DoubleNode n) +setPrev(DoubleNode p) +getData(): Object +getNext(): DoubleNode +getPrev(): DoubleNode

```
DoubleNode()
    data = null
    prev = null
    next = null

DoubleNode(Object d)
    data = d
    prev = null
    next = null
```

Estructura de datos

Clase DoubleNode

DoubleNode
-data:Object -next:DoubleNode -prev:DoubleNode
+DoubleNode() +DoubleNode(Object d) +setData(Object d) +setNext(DoubleNode n) +setPrev(DoubleNode p) +getData(): Object +getNext(): DoubleNode +getPrev(): DoubleNode

```
setData(Object d)
   data = d

setNext(DoubleNode n)
   next = n

setPrev(DoubleNode p)
   prev = p
```

Estructura de datos

Clase DoubleNode

DoubleNode
-data:Object -next:DoubleNode -prev:DoubleNode
+DoubleNode() +DoubleNode(Object d) +setData(Object d) +setNext(DoubleNode n) +setPrev(DoubleNode p) +getData(): Object +getNext(): DoubleNode +getPrev(): DoubleNode

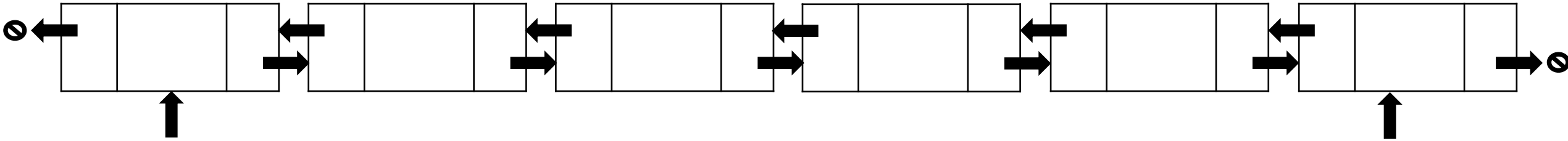
```
getData()
   return data

getNext()
   return next

getPrev()
   retunr prev
```

Estructura de datos

Lista doble o doblemente enlazada



Representación grafica de una lista doble

Estructura de datos

Clase lista doble

- La clase lista doble tiene tres atributos:
  - La cabecera (head) que permite acceder al primer nodo de la lista
  - La cola (tail) que permite acceder al ultimo nodo de la lista (atributo opcional)
  - El tamaño (size) de la lista que mantiene el número de nodos en la colección

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object

```
DoubleList()
    head = null
    tail = null
    size = 0
```

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object

```
size()
   return size

isEmpty()
   return size==0
```

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object

```
first()
   return head

last()
   return tail
```

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object

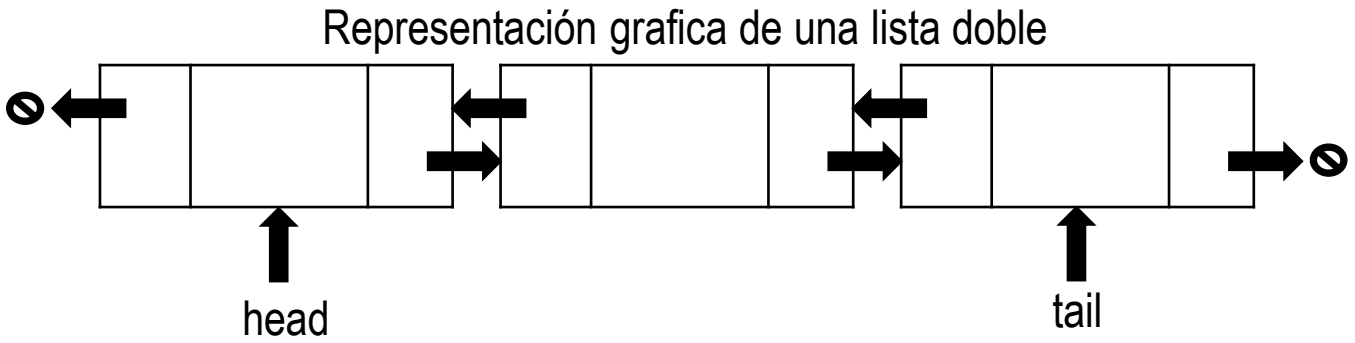
Operaciones básicas con lista doble

Agregar un dato al principio de la lista -> addFirst(Object e)

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al principio de la lista -> addFirst(Object e)

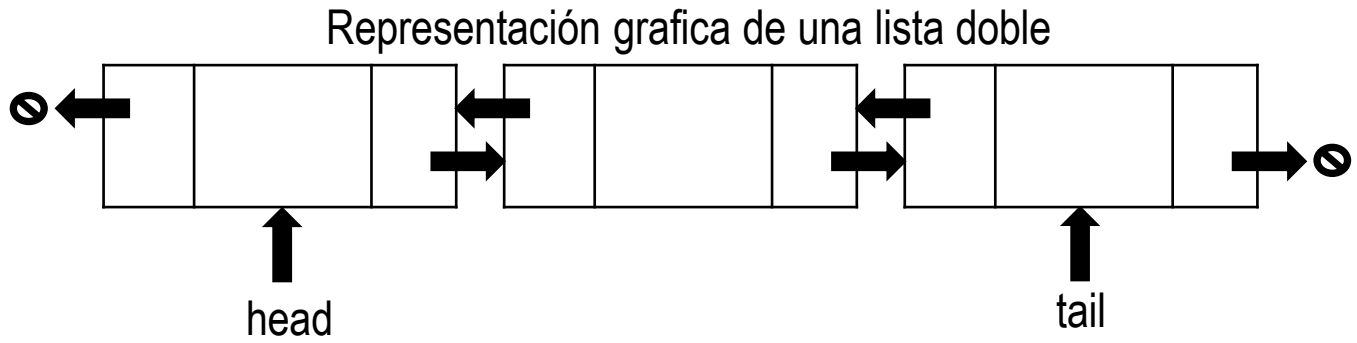


Vamos a insertar el dato **e** a la lista!!!

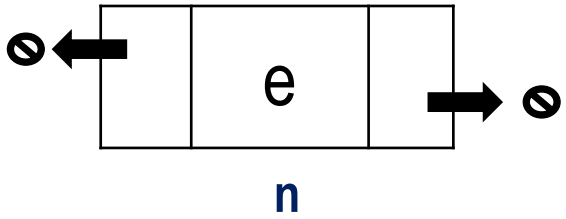
Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al principio de la lista -> addFirst(Object e)



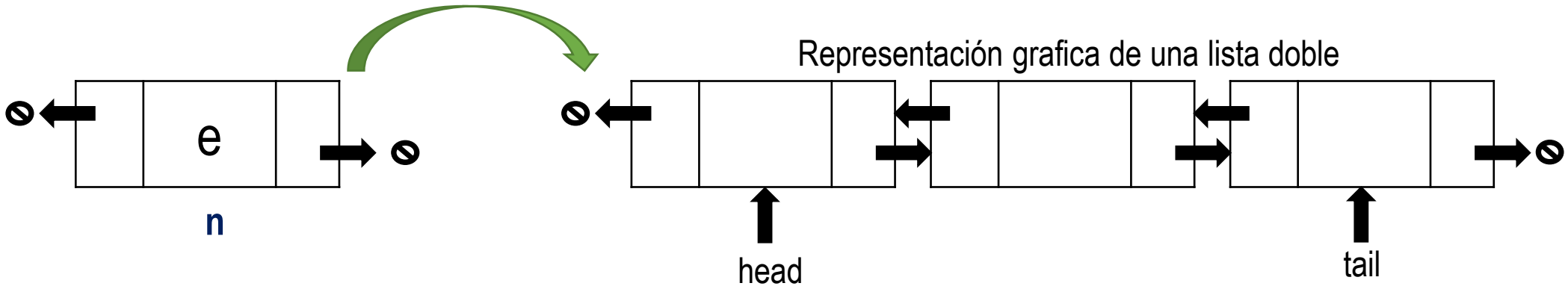
1. Creamos un nuevo nodo que contenga el dato e



Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al principio de la lista -> addFirst(Object e)

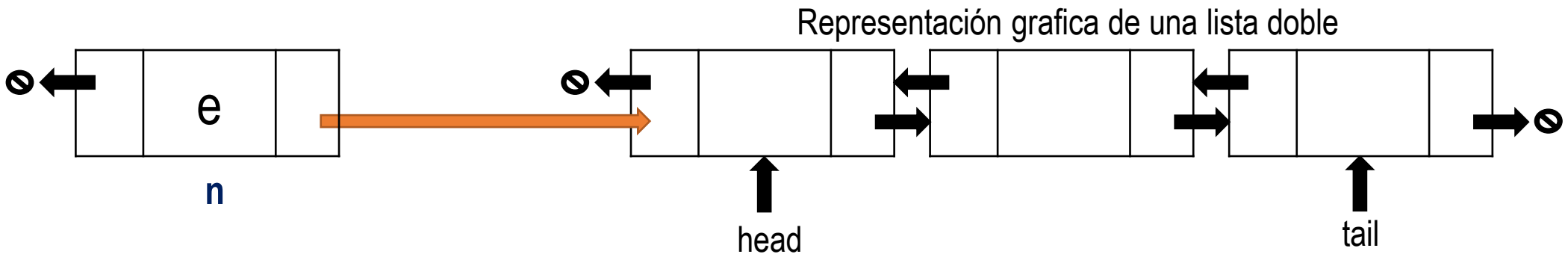


2. Realizamos la conexión del nuevo nodo

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al principio de la lista -> addFirst(Object e)

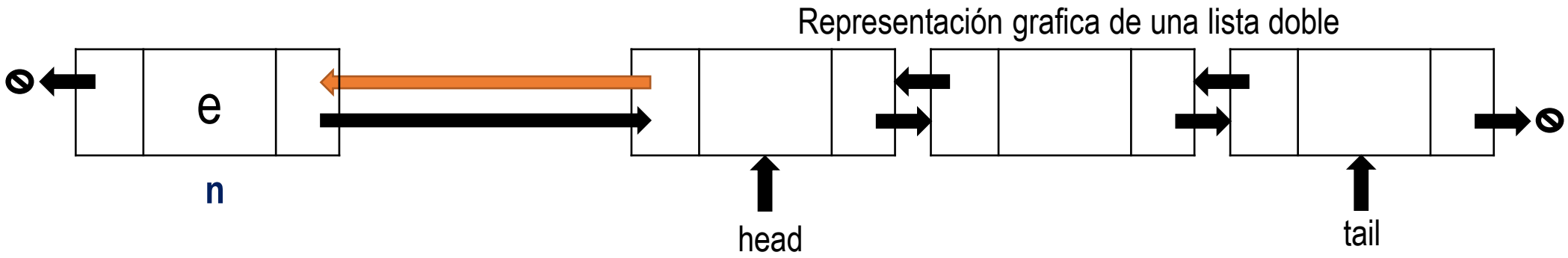


n.next => head

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al principio de la lista -> addFirst(Object e)

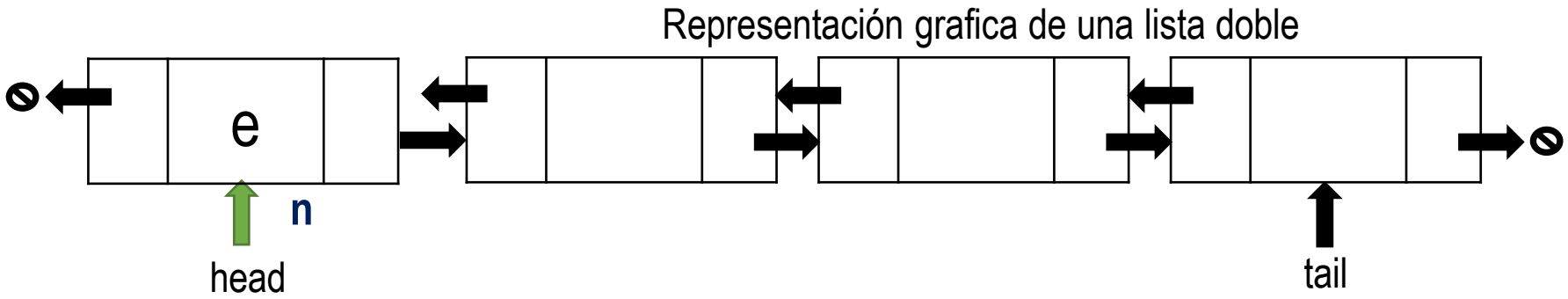


head.prev => n

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al principio de la lista -> addFirst(Object e)

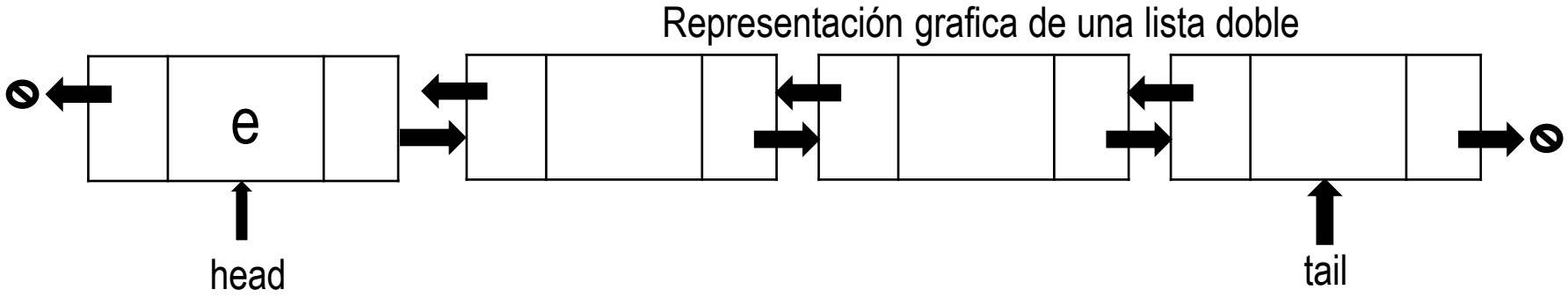


3. Actualizamos la cabecera
head => n

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al principio de la lista -> addFirst(Object e)

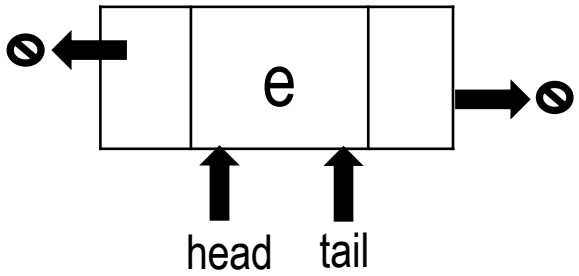


4. Actualizamos el tamaño
size++

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al principio de la lista -> addFirst(Object e)



Si la lista esta vacía, el nuevo nodo es la cabecera y la cola

Estructura de datos

Operaciones básicas con lista doble

```
addFirst(Object e)
    DoubleNode n = new DoubleNode(e)
```

Creamos un nuevo nodo que contenga el dato e

Estructura de datos

Operaciones básicas con lista doble

```
addFirst(Object e)
    DoubleNode n = new DoubleNode(e)
    if isEmpty()
       head = n
       tail = n
```

Si la lista esta vacía, el nuevo nodo es la cabecera y la cola

Estructura de datos

Operaciones básicas con lista doble

```
addFirst(Object e)
    DoubleNode n = new DoubleNode(e)
    if isEmpty()
       head = n
       tail = n
    else
       n.setNext(head)
       head.setPrev(n)
```

Realizamos la conexión del nuevo nodo

Estructura de datos

Operaciones básicas con lista doble

```
addFirst(Object e)
    DoubleNode n = new DoubleNode(e)
    if isEmpty()
       head = n
       tail = n
    else
       n.setNext(head)
       head.setPrev(n)
       head = n
```

Actualizamos la cabecera
head => n

Estructura de datos

Operaciones básicas con lista doble

```
addFirst(Object e)
    DoubleNode n = new DoubleNode(e)
    if isEmpty()
       head = n
       tail = n
    else
       n.setNext(head)
       head.setPrev(n)
       head = n
    size++
```

Actualizamos el tamaño
size++

Estructura de datos

Operaciones básicas con lista doble

```
addFirst(Object e)
    DoubleNode n = new DoubleNode(e)
    if isEmpty()
       head = n
       tail = n
    else
       n.setNext(head)
       head.setPrev(n)
       head = n
    size++
```

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object

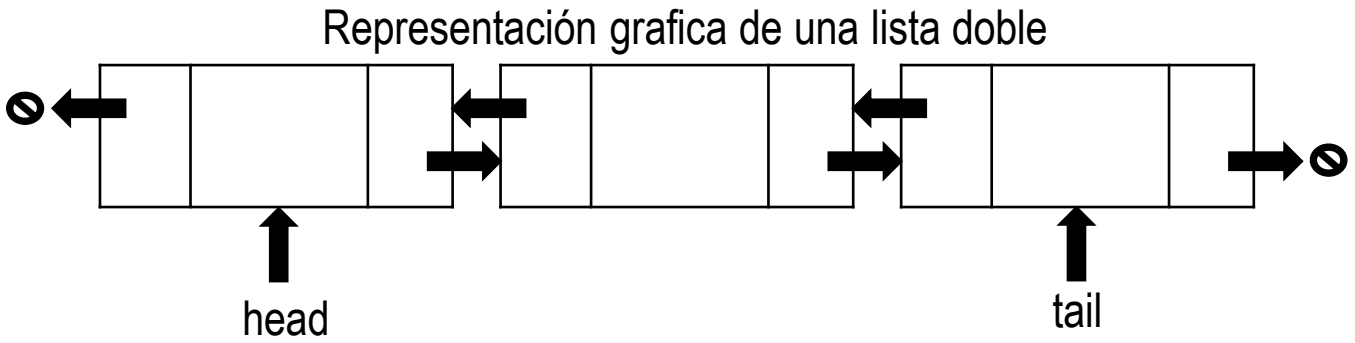
Operaciones básicas con lista doble

Agregar un dato al final de la lista -> addLast(Object e)

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al final de la lista -> addLast(Object e)

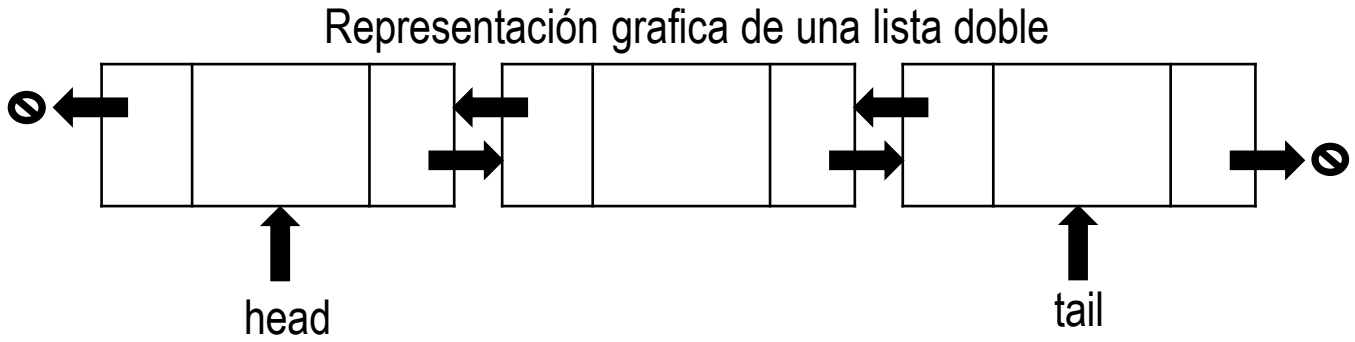


Vamos a insertar el dato **e** a la lista!!!

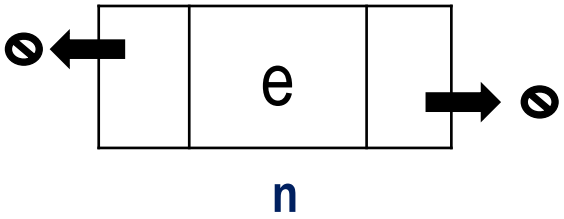
Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al final de la lista -> addLast(Object e)



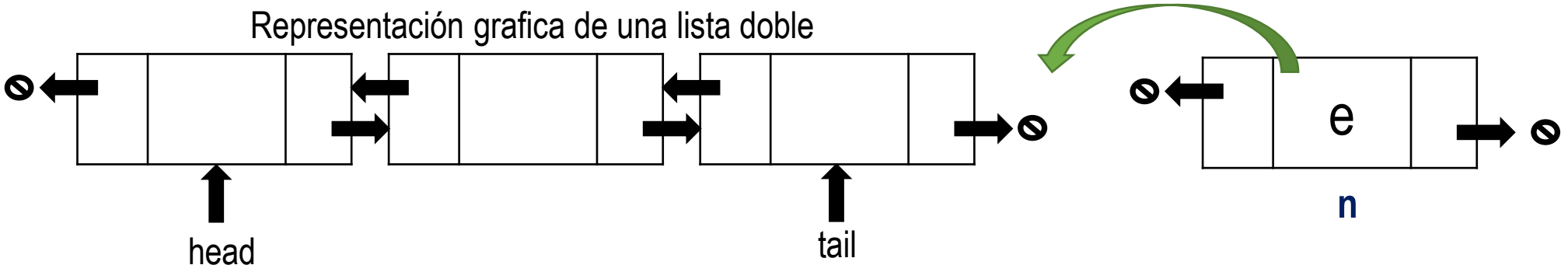
1. Creamos un nuevo nodo que contenga el dato e



Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al final de la lista -> addLast(Object e)

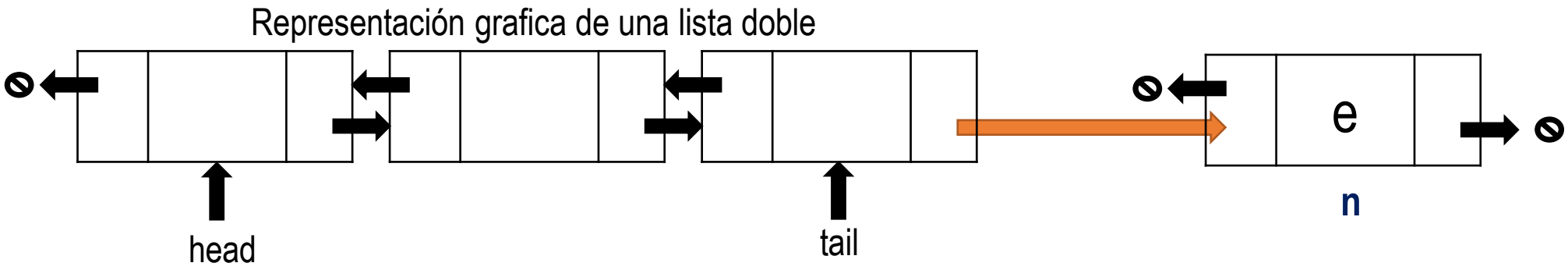


2. Realizamos la conexión del nuevo nodo

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al final de la lista -> addLast(Object e)

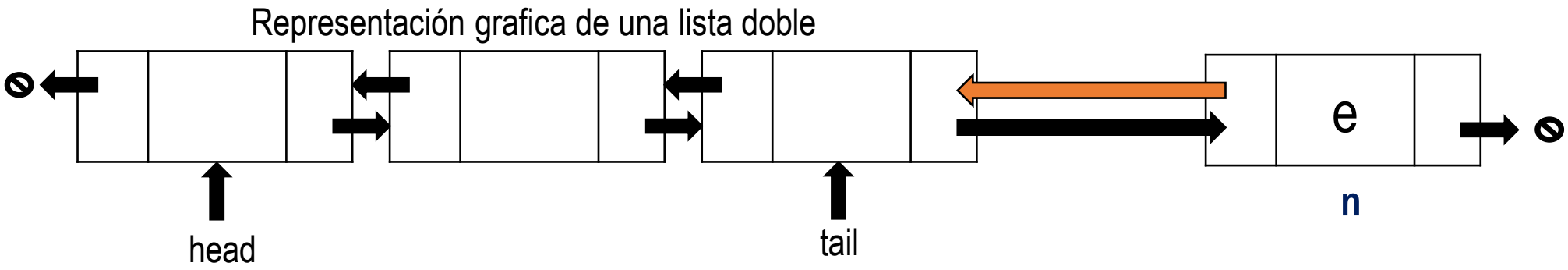


tail.next->n

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al final de la lista -> addLast(Object e)

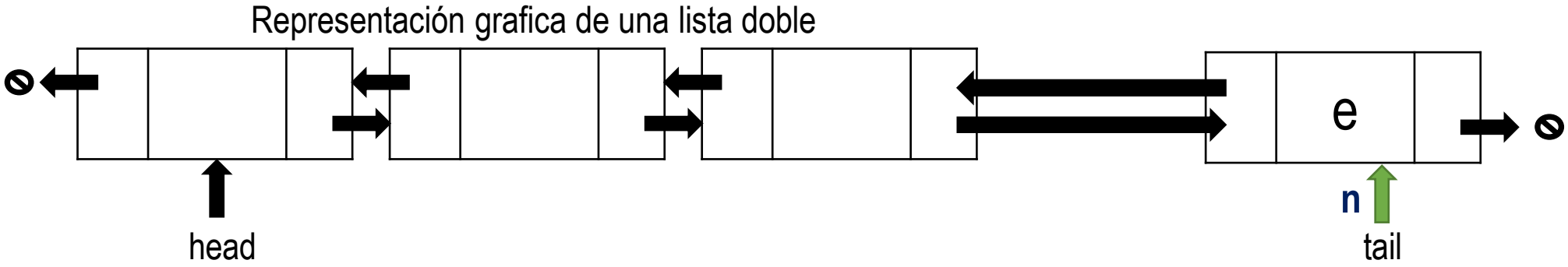


n.prev->tail

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al final de la lista -> addLast(Object e)

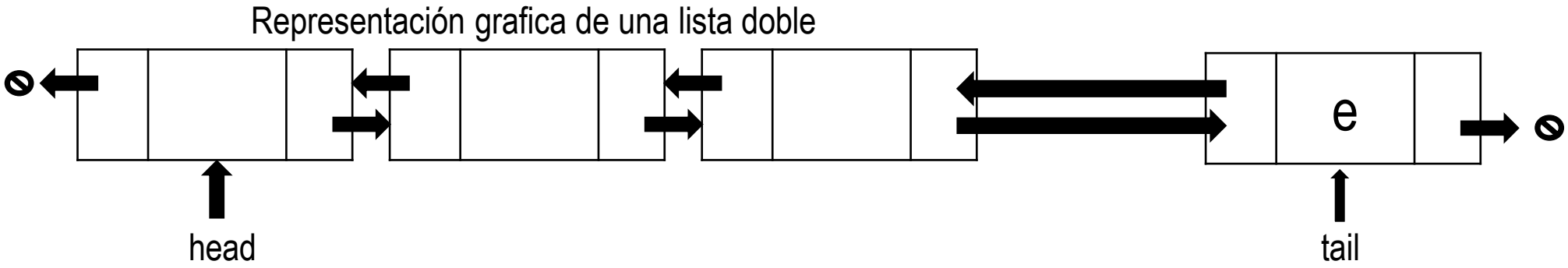


3. Actualizamos la cola
tail => n

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al final de la lista -> addLast(Object e)

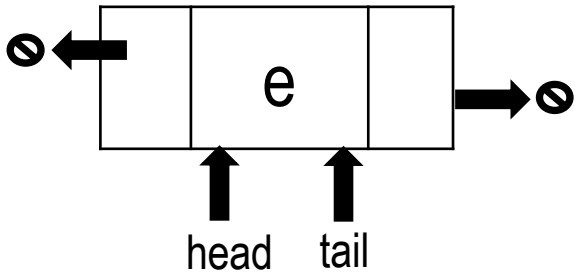


4. Actualizamos el tamaño
size++

Estructura de datos

Operaciones básicas con lista doble

Agregar un dato al final de la lista -> addLast(Object e)



Si la lista esta vacía, el nuevo nodo es la cabecera y la cola

Estructura de datos

Operaciones básicas con lista doble

```
addLast(Object e)
   DoubleNode n = new DoubleNode(e)
```

Creamos un nuevo nodo que contenga el dato e

Estructura de datos

Operaciones básicas con lista doble

```
addLast(Object e)
   DoubleNode n = new DoubleNode(e)
   if isEmpty()
      head = n
      tail = n
```

Si la lista esta vacía, el nuevo nodo es la cabecera y la cola

Estructura de datos

Operaciones básicas con lista doble

```
addLast(Object e)
   DoubleNode n = new DoubleNode(e)
   if isEmpty()
      head = n
      tail = n
   else
      tail.setNext(n)
      n.setPrev(tail)
```

Realizamos la conexión del nuevo nodo

Estructura de datos

Operaciones básicas con lista doble

```
addLast(Object e)
   DoubleNode n = new DoubleNode(e)
   if isEmpty()
      head = n
      tail = n
   else
      tail.setNext(n)
      n.setPrev(tail)
      tail = n
```

Actualizamos la cola
tail => n

Estructura de datos

Operaciones básicas con lista doble

```
addLast(Object e)
   DoubleNode n = new DoubleNode(e)
   if isEmpty()
      head = n
      tail = n
   else
      tail.setNext(n)
      n.setPrev(tail)
      tail = n
   size++
```

Actualizamos el tamaño
size++

Estructura de datos

Operaciones básicas con lista doble

```
addLast(Object e)
   DoubleNode n = new DoubleNode(e)
   if isEmpty()
      head = n
      tail = n
   else
      tail.setNext(n)
      n.setPrev(tail)
      tail = n
   size++
```

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object

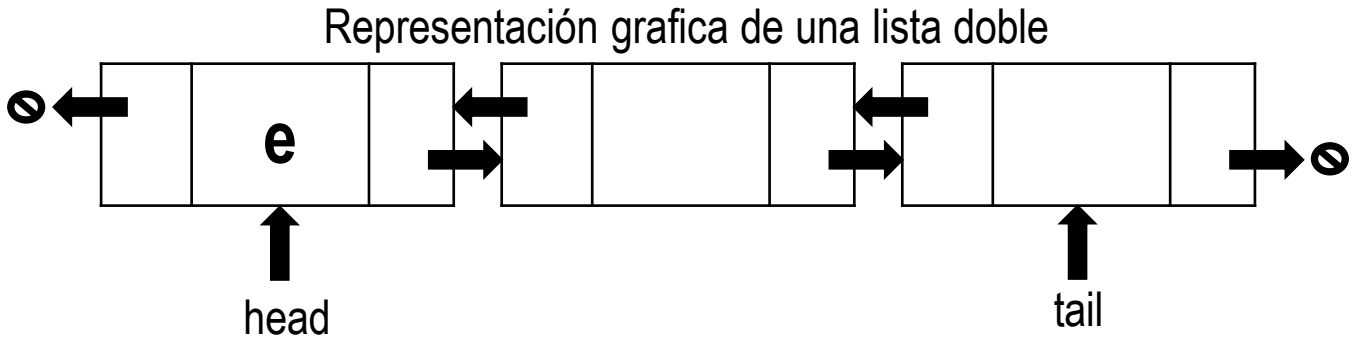
Operaciones básicas con lista doble

Eliminar y retornar el dato al principio de la lista->removeFirst()

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al principio de la lista -> removeFirst()

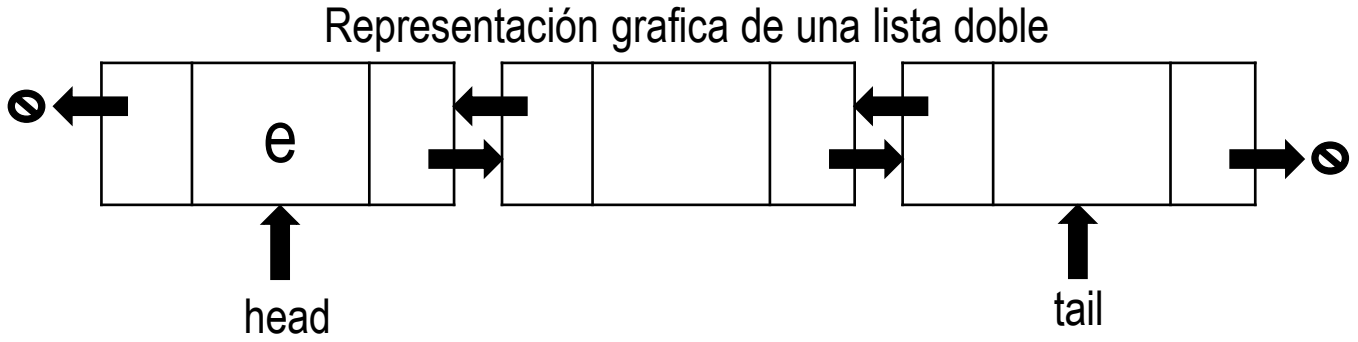


Vamos a eliminar la cabecera y retornar el dato almacenado!!!

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al principio de la lista -> removeFirst()



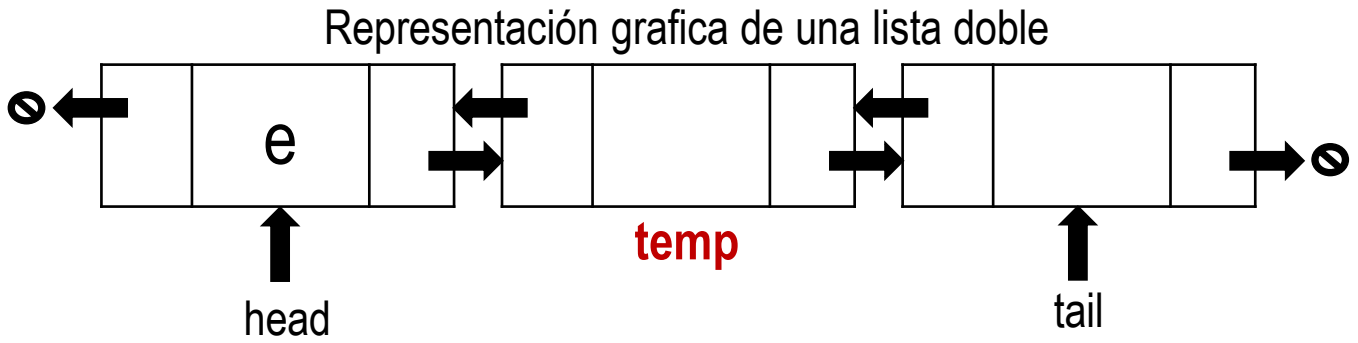
1. Creamos una variable temporal para almacenar e

temp_dato = e

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al principio de la lista -> removeFirst()



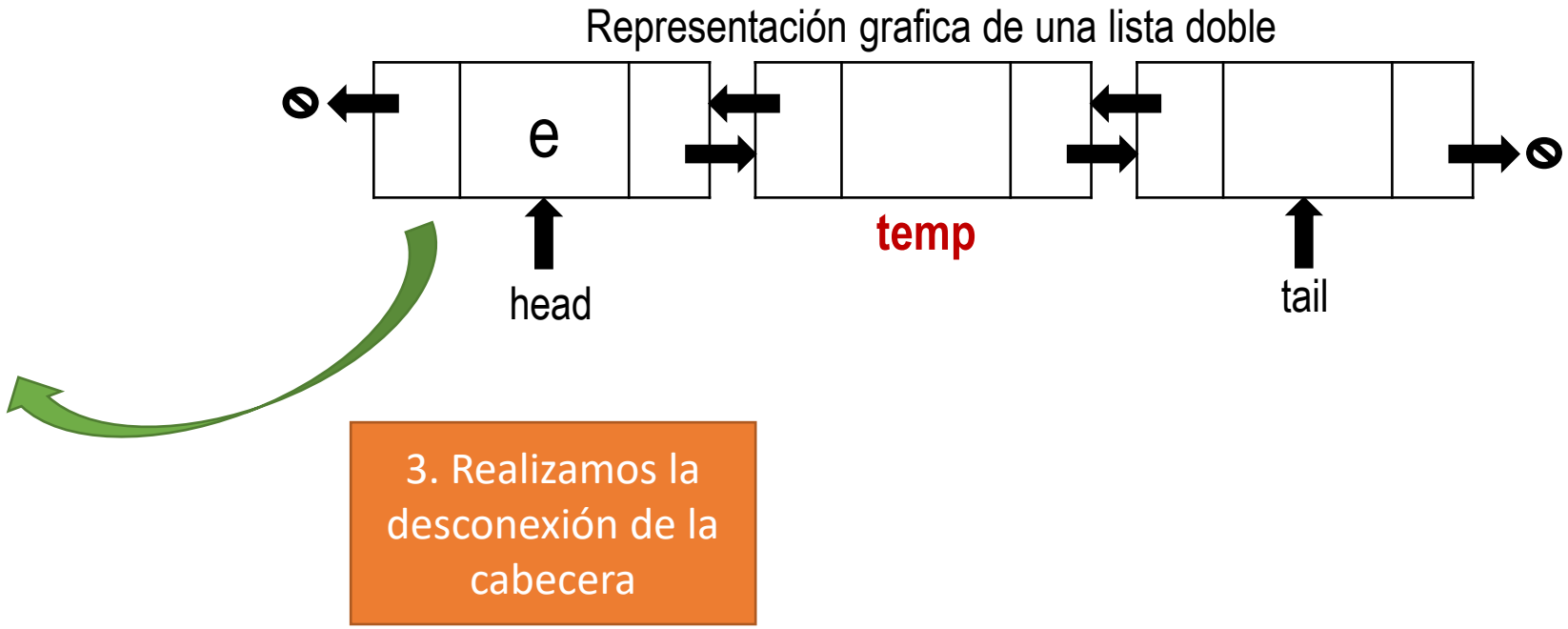
2. Creamos una variable temporal almacenar la nueva cabecera

temp = head.next

Estructura de datos

Operaciones básicas con lista doble

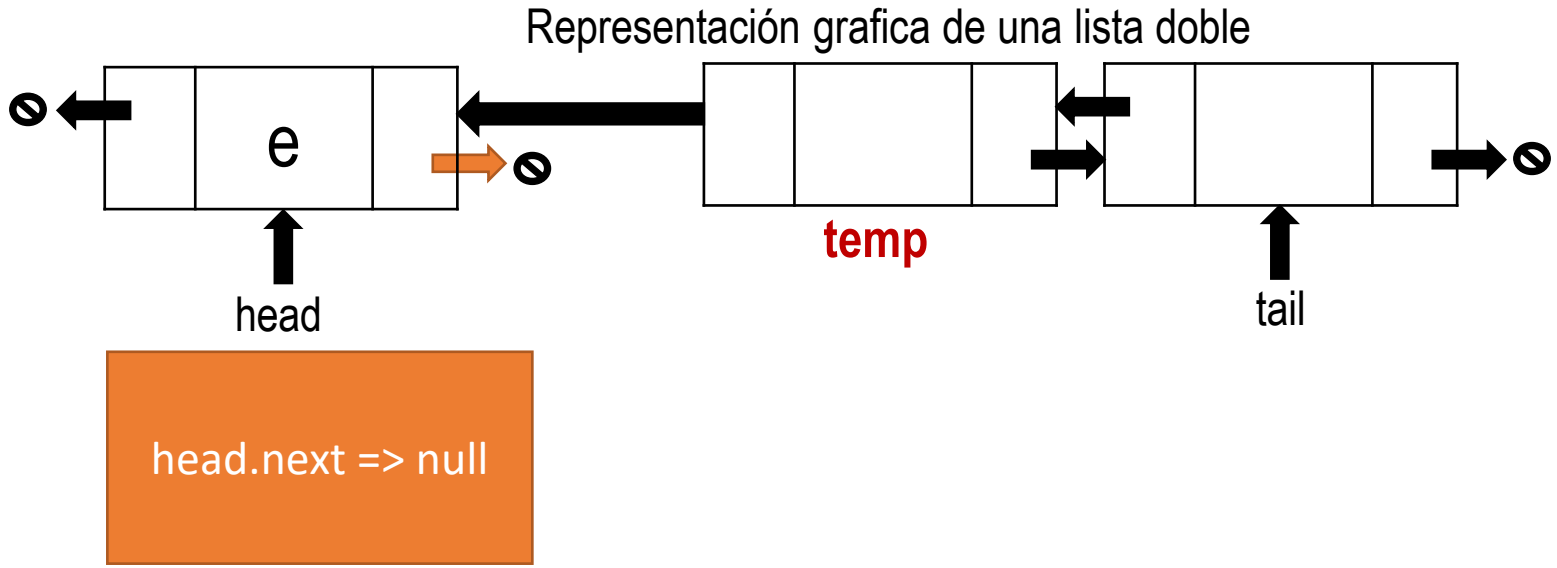
Eliminar y retornar el dato al principio de la lista -> removeFirst()



Estructura de datos

Operaciones básicas con lista doble

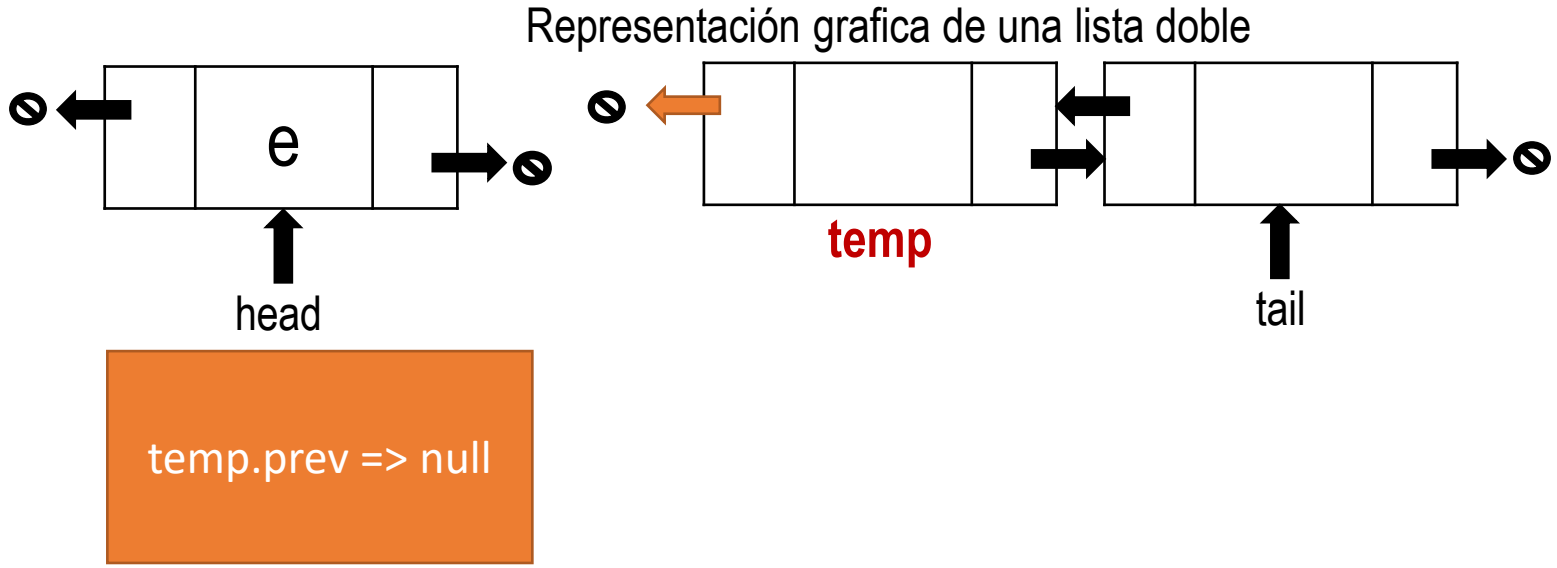
Eliminar y retornar el dato al principio de la lista -> removeFirst()



Estructura de datos

Operaciones básicas con lista doble

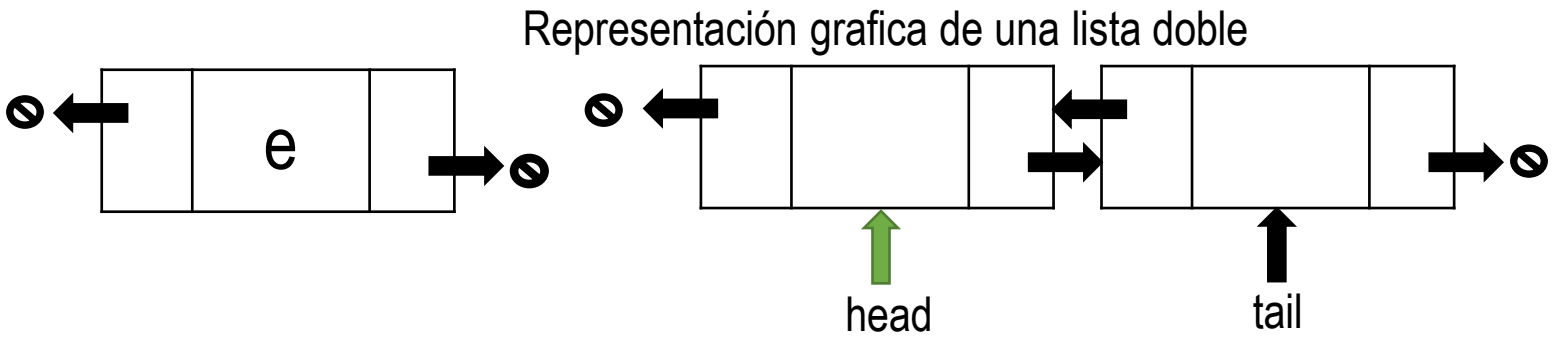
Eliminar y retornar el dato al principio de la lista -> removeFirst()



Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al principio de la lista -> removeFirst()

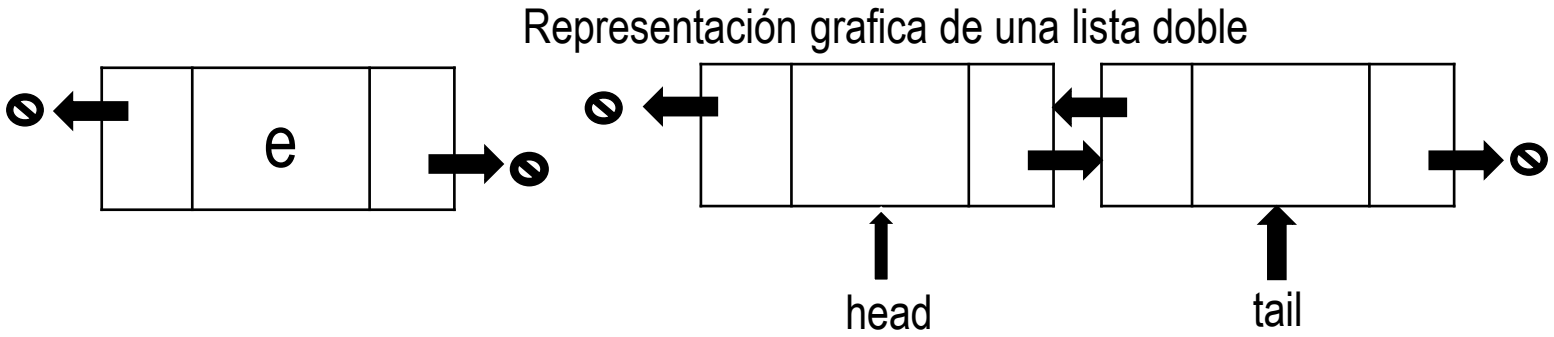


4. Actualizamos la cabecera head=>temp

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al principio de la lista -> removeFirst()



5. Actualizamos el tamaño
size--

Estructura de datos

Operaciones básicas con lista doble

```
removeFirst()
        if !isEmpty()
            Object temp_dato = head.getData()
```

Creamos una variable temporal para almacenar e

Estructura de datos

Operaciones básicas con lista doble

```
removeFirst()
        if !isEmpty()
            Object temp_dato = head.getData()
            if size == 1
                head=null
                tail=null
            else
                DoubleNode temp = head.getNext()
```

Creamos una variable temporal almacenar la nueva cabecera

Estructura de datos

Operaciones básicas con lista doble

```
removeFirst()
        if !isEmpty()
            Object temp_dato = head.getData()
            if size == 1
                head=null
                tail=null
            else
                DoubleNode temp = head.getNext()
                head.setNext(null)
                temp.setPrev(null)
```

Realizamos la desconexión de la cabecera

Estructura de datos

Operaciones básicas con lista doble

```
removeFirst()
        if !isEmpty()
            Object temp_dato = head.getData()
            if size == 1
                head=null
                tail=null
            else
                DoubleNode temp = head.getNext()
                head.setNext(null)
                temp.setPrev(null)
                head = temp
```

Actualizamos la cabecera head=>temp

Estructura de datos

Operaciones básicas con lista doble

```
removeFirst()
        if !isEmpty()
           Object temp_dato = head.getData()
           if size == 1
              head=null
              tail=null
           else
              DoubleNode temp = head.getNext()
              head.setNext(null)
              temp.setPrev(null)
              head = temp
           size--
           return temp_dato
        else
           return null
```

Actualizamos el tamaño
size--

Estructura de datos

Operaciones básicas con lista doble

```
removeFirst()
        if !isEmpty()
           Object temp_dato = head.getData()
           if size == 1
              head=null
              tail=null
           else
              DoubleNode temp = head.getNext()
              head.setNext(null)
              temp.setPrev(null)
              head = temp
           size--
           return temp_dato
        else
           return null
```

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object

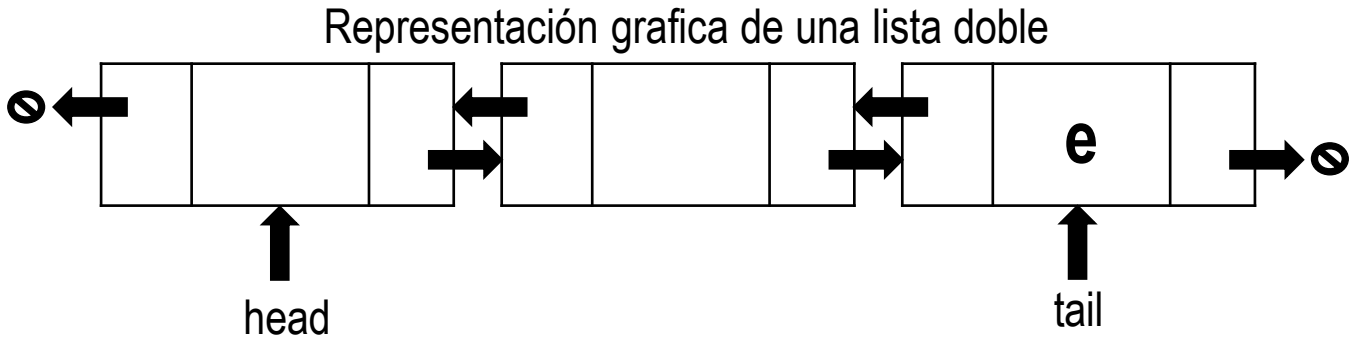
Operaciones básicas con lista doble

Eliminar y retornar el dato al final de la lista -> removeLast()

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al final de la lista -> removeLast()

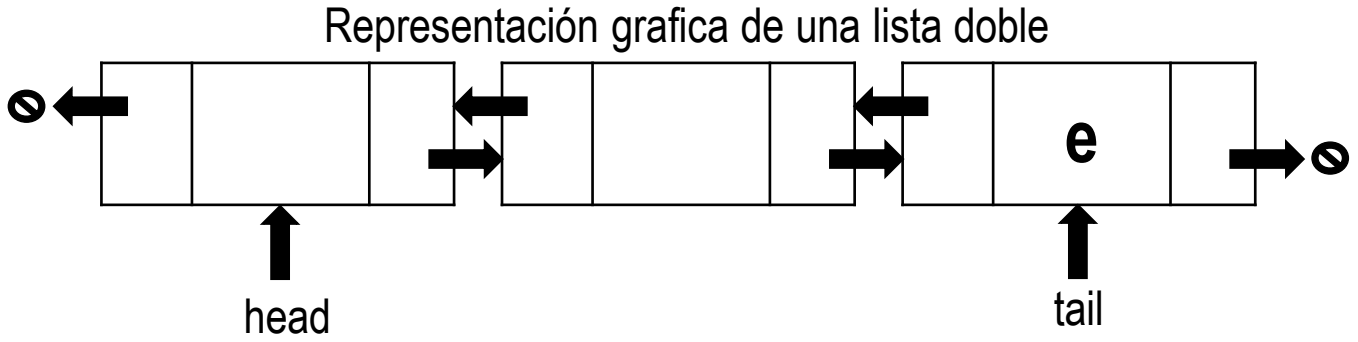


Vamos a eliminar la cola y retornar el dato almacenado!!!

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al final de la lista -> removeLast()



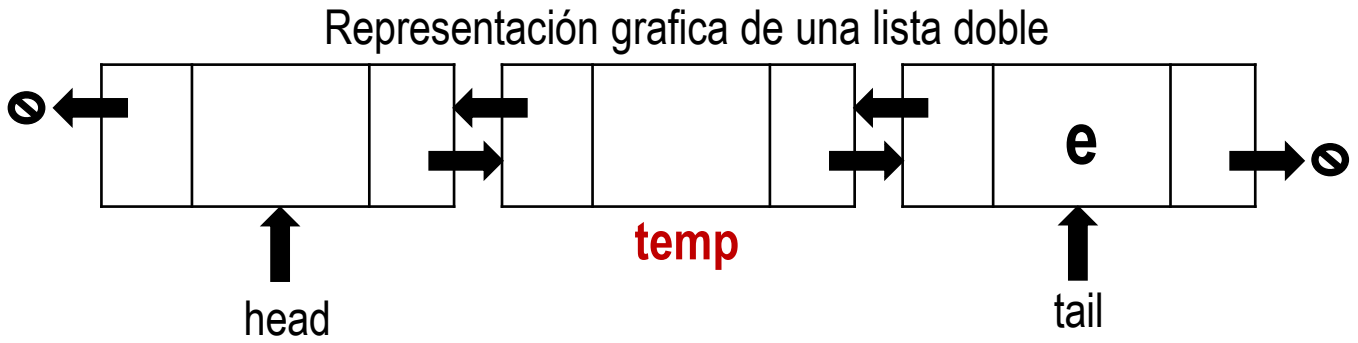
1. Creamos una variable temporal para almacenar e

temp_dato = e

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al final de la lista -> removeLast()



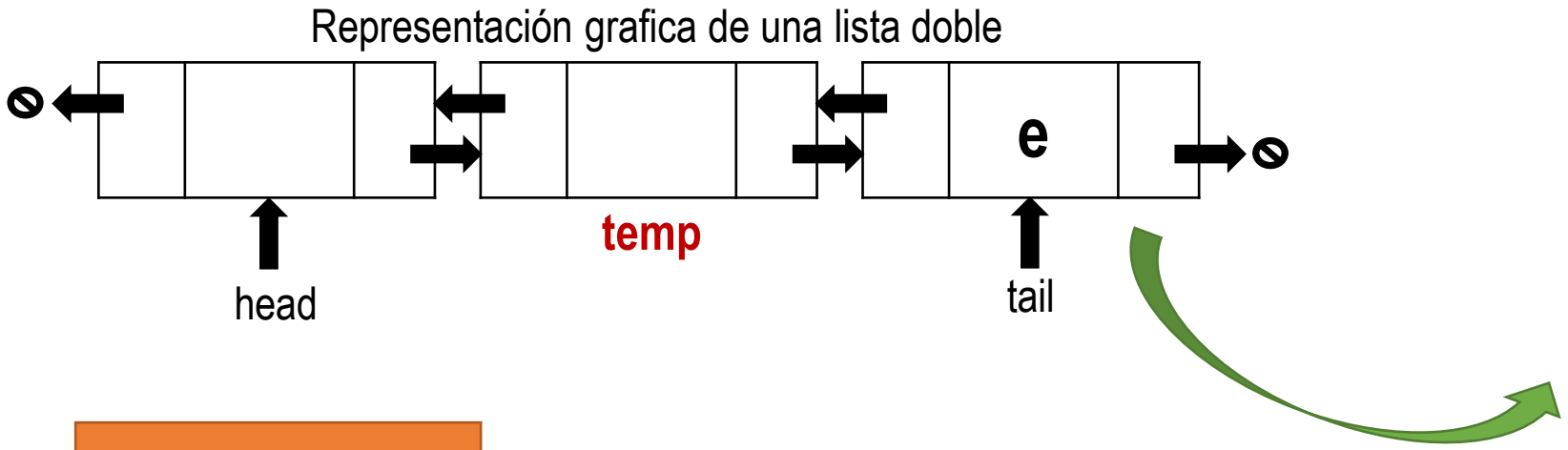
2. Creamos una variable temporal almacenar la nueva cola

temp = tail.prev

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al final de la lista -> removeLast()

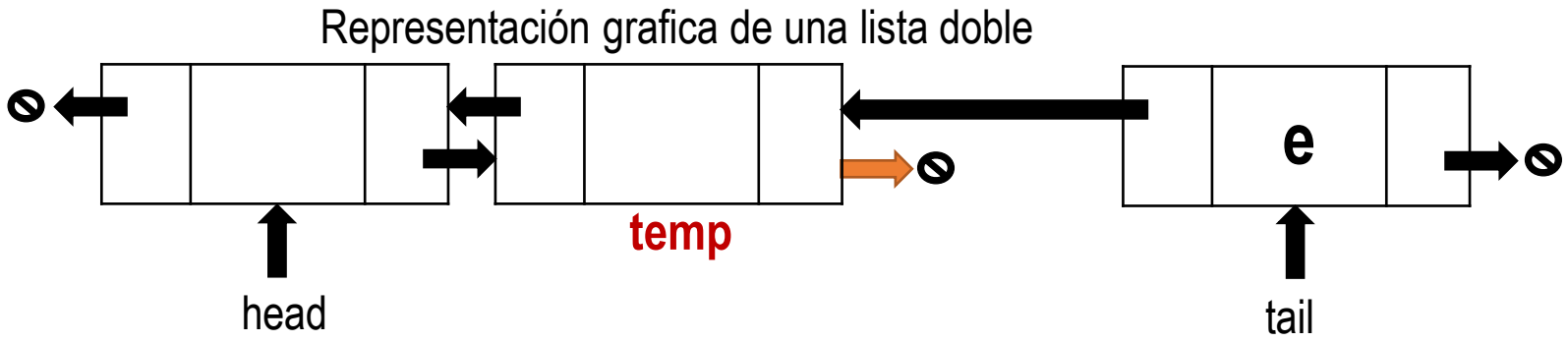


3. Realizamos la desconexión de la cola

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al final de la lista -> removeLast()

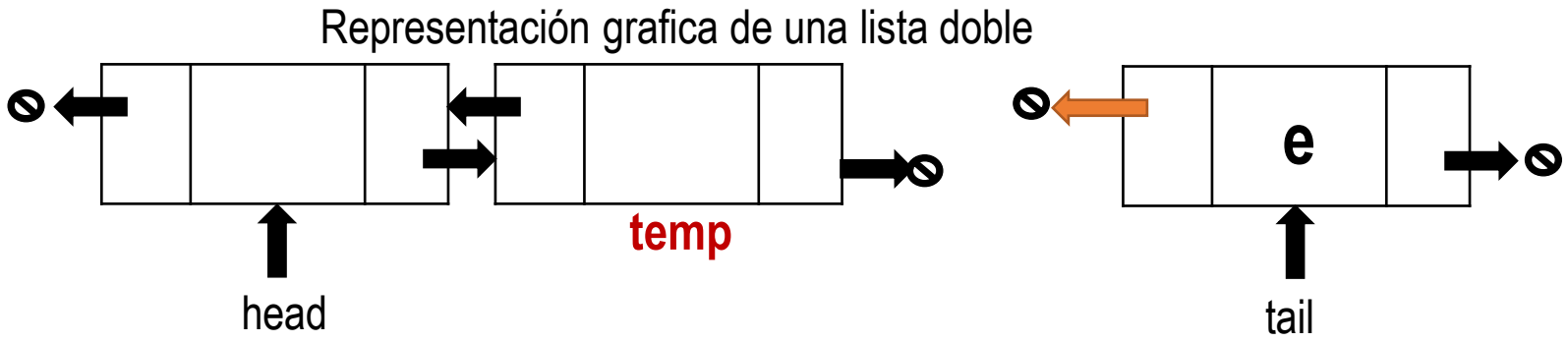


temp.next-> null

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al final de la lista -> removeLast()

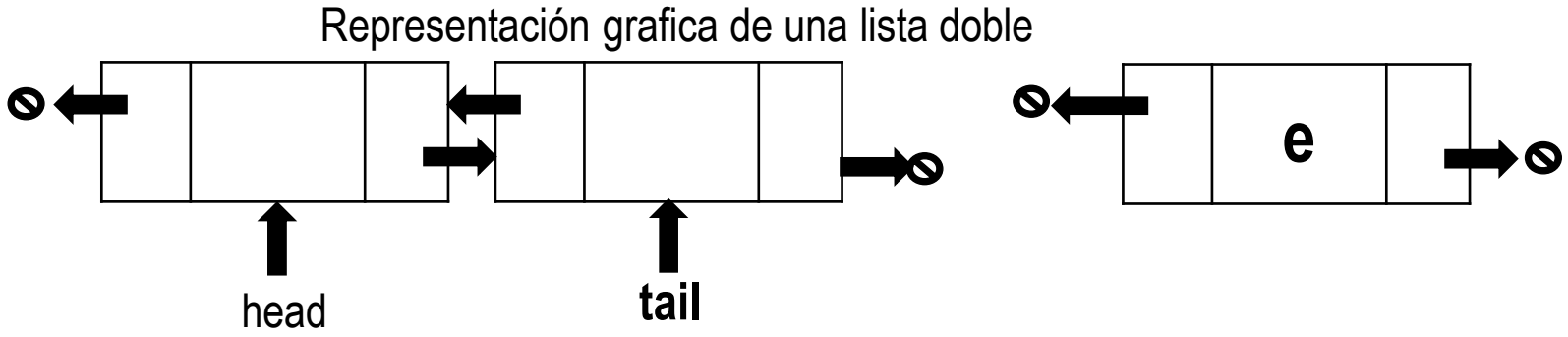


tail.prev-> null

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al final de la lista -> removeLast()

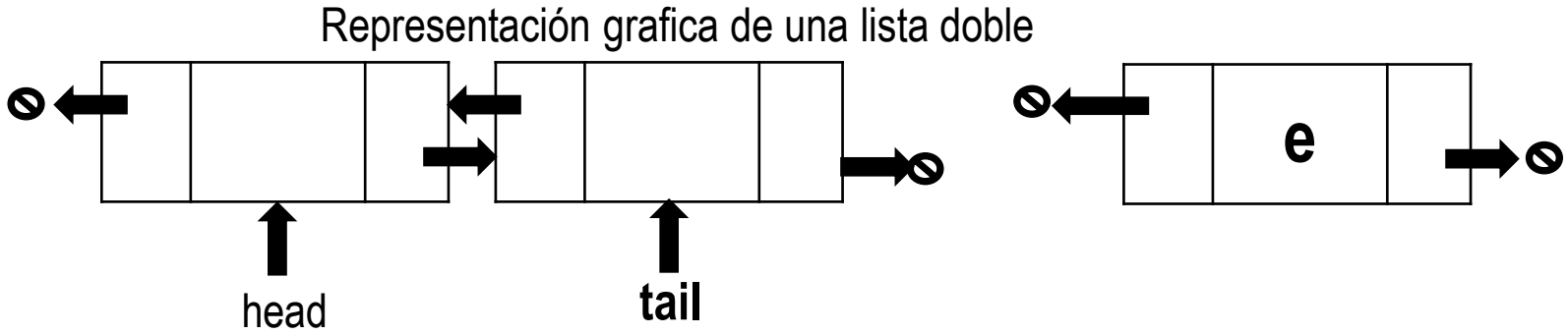


4. Actualizamos la cola
tail=>temp

Estructura de datos

Operaciones básicas con lista doble

Eliminar y retornar el dato al final de la lista -> removeLast()



5. Actualizamos el tamaño
size--

Estructura de datos

Operaciones básicas con lista doble

```
removeLast()
        if !isEmpty()
           Object temp_dato = tail.getData()
```

Creamos una variable temporal para almacenar e

Estructura de datos

Operaciones básicas con lista doble

```
removeLast()
        if !isEmpty()
           Object temp_dato = tail.getData()
            if size == 1
               head=null
               tail=null
            else
               DoubleNode temp = tail.getPrev()
```

Creamos una variable temporal almacenar la nueva cola

Estructura de datos

Operaciones básicas con lista doble

```
removeLast()
        if !isEmpty()
           Object temp_dato = tail.getData()
            if size == 1
               head=null
               tail=null
            else
               DoubleNode temp = tail.getPrev()
               tail.setPrev(null)
               temp.setNext(null)
```

Realizamos la desconexión de la cola

Estructura de datos

Operaciones básicas con lista doble

```
removeLast()
        if !isEmpty()
           Object temp_dato = tail.getData()
            if size == 1
               head=null
               tail=null
            else
               DoubleNode temp = tail.getPrev()
               tail.setPrev(null)
               temp.setNext(null)
               tail = temp
```

Actualizamos la cola
tail=>temp

Estructura de datos

Operaciones básicas con lista doble

```
removeLast()
        if !isEmpty()
           Object temp_dato = tail.getData()
            if size == 1
               head=null
               tail=null
            else
               DoubleNode temp = tail.getPrev()
               tail.setPrev(null)
               temp.setNext(null)
               tail = temp
           size--
           return temp_dato
        else
           return null
```

Actualizamos el tamaño
size--

Estructura de datos

Operaciones básicas con lista doble

```
removeLast()
        if !isEmpty()
           Object temp_dato = tail.getData()
            if size == 1
               head=null
               tail=null
            else
               DoubleNode temp = tail.getPrev()
               tail.setPrev(null)
               temp.setNext(null)
               tail = temp
           size--
           return temp_dato
        else
           return null
```

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object

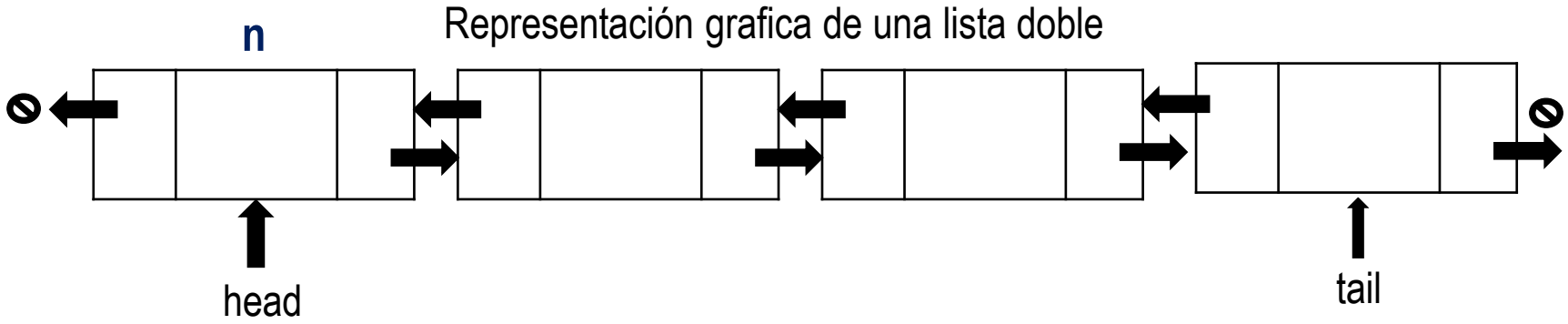
Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)

Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)

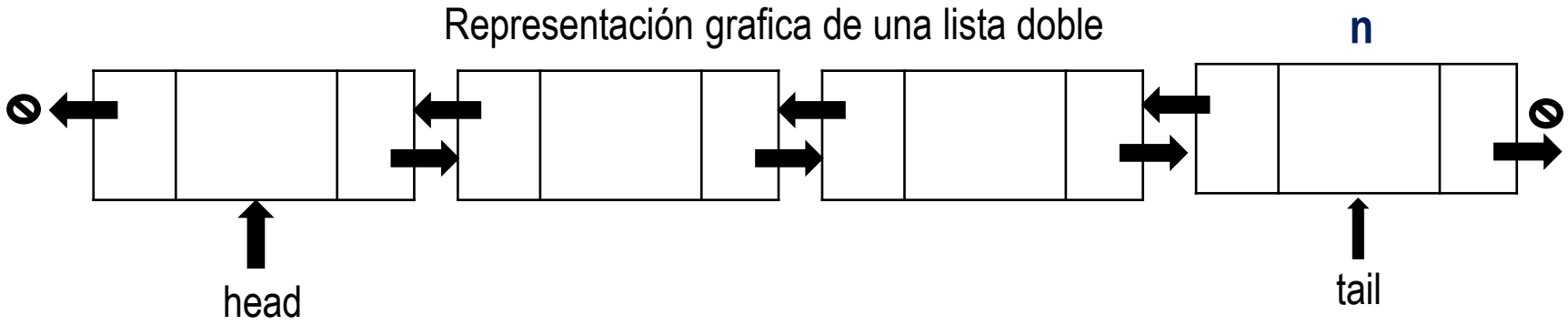


- Caso 1: n es la cabecera => usamos el método removeFirst()

Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)

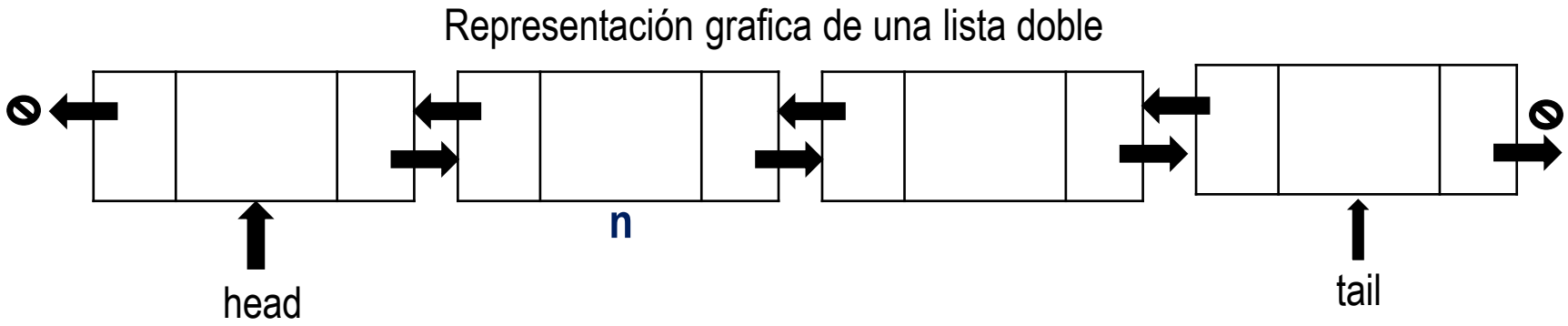


- Caso 2: n es la cola => usamos el método removeLast()

Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)

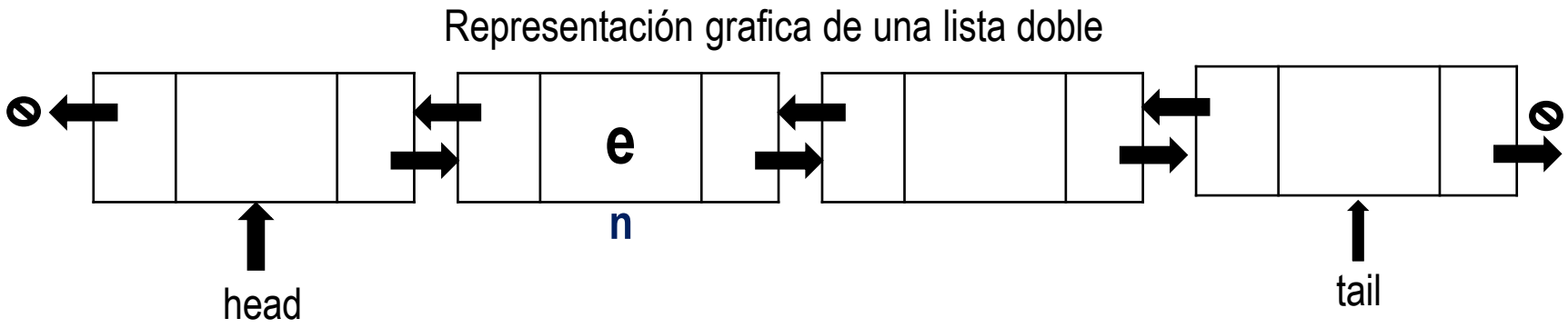


- Caso 3: n es un nodo diferente a la cabecera y la cola

Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)



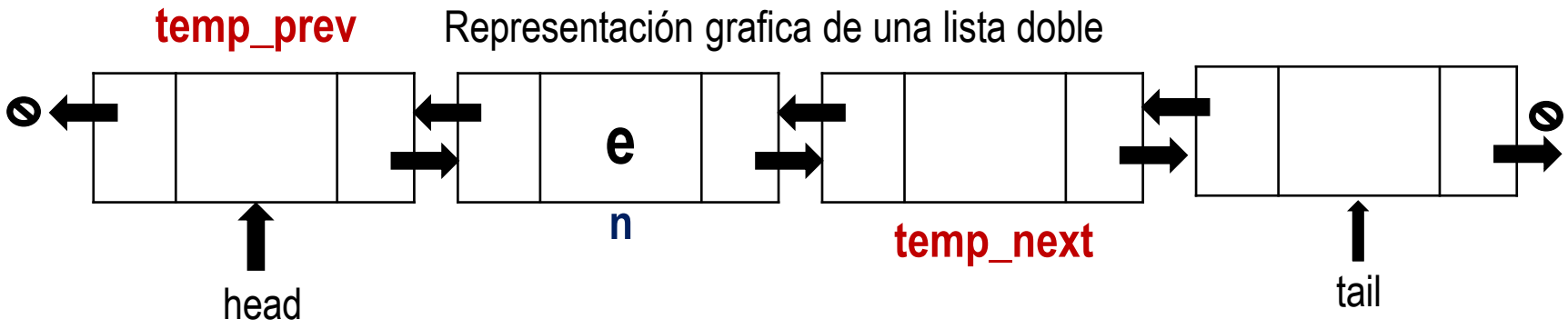
- Caso 3: n es un nodo diferente a la cabecera y la cola

1. Creamos una variable temporal para almacenar e

Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)



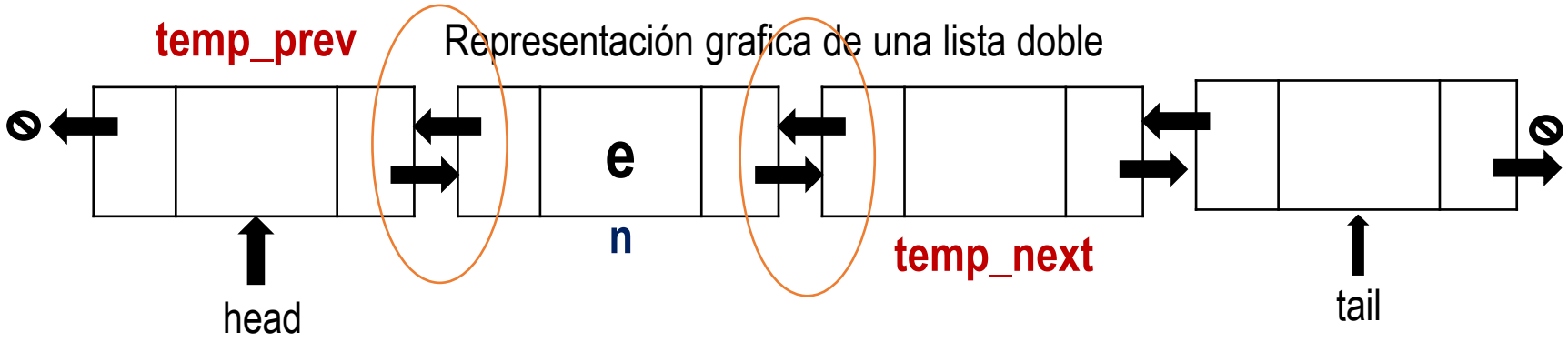
- Caso 3: n es un nodo diferente a la cabecera y la cola

2. Creamos una variable temporal almacenar el nodo siguiente y previo a n

Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)



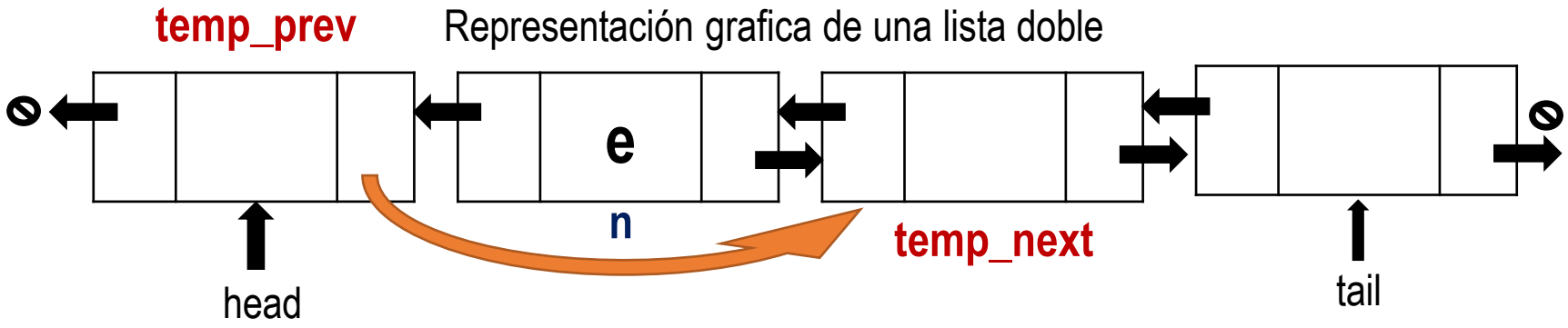
- Caso 3: n es un nodo diferente a la cabecera y la cola

3. Realizamos la desconexión del nodo n

Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)



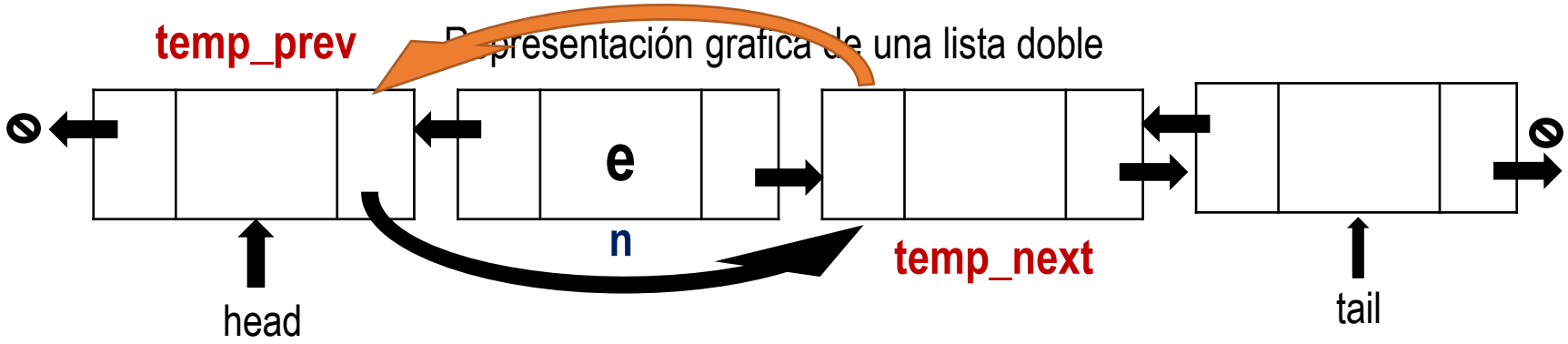
- Caso 3: n es un nodo diferente a la cabecera y la cola

3.1 temp_prev.next = temp_next

Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)



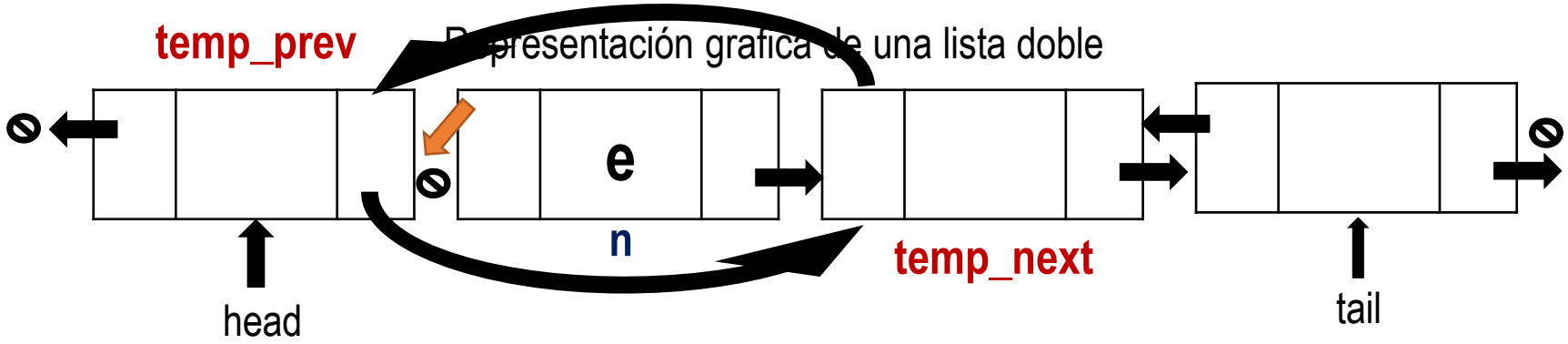
- Caso 3: n es un nodo diferente a la cabecera y la cola

3.2 temp_next.prev = temp_prev

Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)



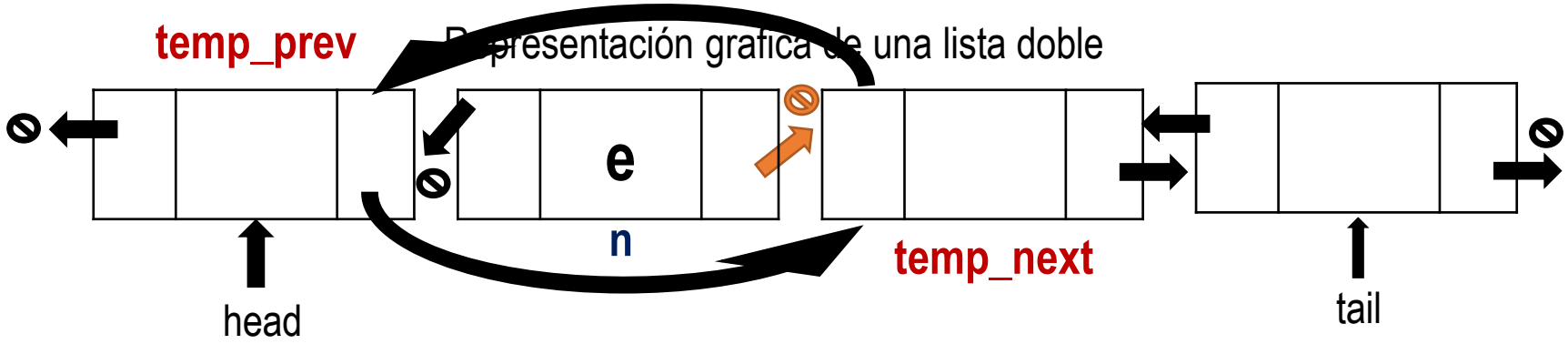
- Caso 3: n es un nodo diferente a la cabecera y la cola

3.3 n.prev = null

Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)



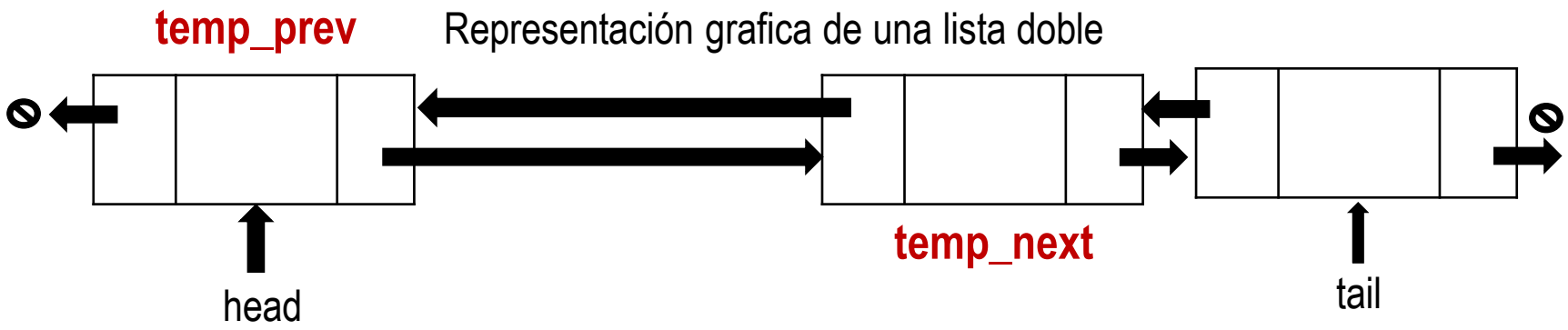
- Caso 3: n es un nodo diferente a la cabecera y la cola

3.4 n.next = null

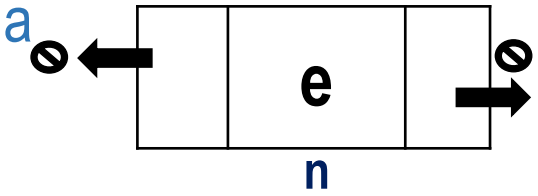
Estructura de datos

Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)



- Caso 3: n es un nodo diferente a la cabecera y la cola

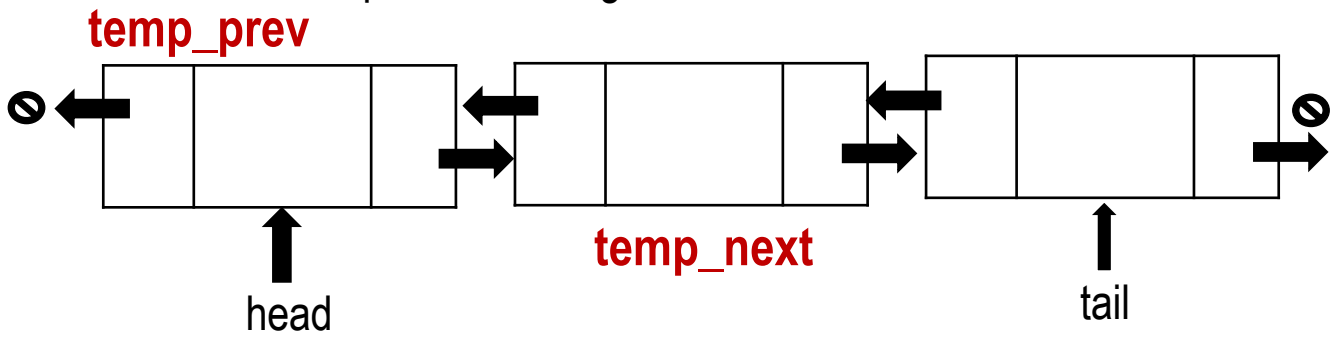


Estructura de datos

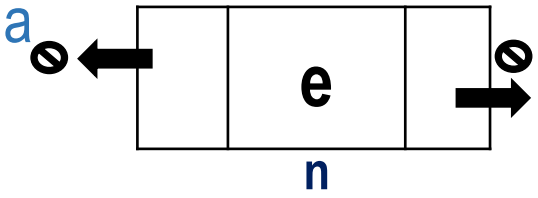
Operaciones básicas con lista doble

Eliminar un nodo de la lista -> remove(DoubleNode n)

Representación grafica de una lista doble



- Caso 3: n es un nodo diferente a la cabecera y la cola



4. Actualizamos el tamaño
size--

Estructura de datos

Operaciones básicas con lista doble

```
remove(DoubleNode n)
    if n==head
       return removeFirst()
```

- Caso 1: n es la cabecera => usamos el método removeFirst()

Estructura de datos

Operaciones básicas con lista doble

```
remove(DoubleNode n)
    if n==head
       return removeFirst()
    elseif n==tail
       return removeLast()
```

- Caso 2: n es la cola => usamos el método removeLast()

Estructura de datos

Operaciones básicas con lista doble

```
remove(DoubleNode n)
    if n==head
       return removeFirst()
    elseif n==tail
       return removeLast()
    else
```

- Caso 3: n es un nodo diferente a la cabecera y la cola

Estructura de datos

Operaciones básicas con lista doble

```
remove(DoubleNode n)
    if n==head
       return removeFirst()
    elseif n==tail
       return removeLast()
    else
       Object temp_dato = n.getData()
```

Creamos una variable temporal para almacenar e

Estructura de datos

Operaciones básicas con lista doble

```
remove(DoubleNode n)
    if n==head
       return removeFirst()
    elseif n==tail
       return removeLast()
    else
       Object temp_dato = n.getData()
       DoubleNode temp_prev = n.getPrev()
       DoubleNode temp_next = n.getNext()
```

Creamos variables temporales almacenar el nodo siguiente y previo a n

Estructura de datos

Operaciones básicas con lista doble

```
remove(DoubleNode n)
    if n==head
       return removeFirst()
    elseif n==tail
       return removeLast()
    else
       Object temp_dato = n.getData()
       DoubleNode temp_prev = n.getPrev()
       DoubleNode temp_next = n.getNext()
       temp_prev.setNext(temp_next)
       temp_next.setPrev(temp_prev)
       n.setNext(null)
       n.setPrev(null)
```

Realizamos la desconexión del nodo n

Estructura de datos

Operaciones básicas con lista doble

```
remove(DoubleNode n)
    if n==head
       return removeFirst()
    elseif n==tail
       return removeLast()
    else
       Object temp_dato = n.getData()
       DoubleNode temp_prev = n.getPrev()
       DoubleNode temp_next = n.getNext()
       temp_prev.setNext(temp_next)
       temp_next.setPrev(temp_prev)
       n.setNext(null)
       n.setPrev(null)
       size --
```

Actualizamos el tamaño
size--

Estructura de datos

Operaciones básicas con lista doble

```
remove(DoubleNode n)
    if n==head
       return removeFirst()
    elseif n==tail
       return removeLast()
    else
       Object temp_dato = n.getData()
       DoubleNode temp_prev = n.getPrev()
       DoubleNode temp_next = n.getNext()
       temp_prev.setNext(temp_next)
       temp_next.setPrev(temp_prev)
       n.setNext(null)
       n.setPrev(null)
       size --
       return temp_dato
```

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object +addAfter(DoubleNode n, Object e)

Operaciones básicas con lista doble

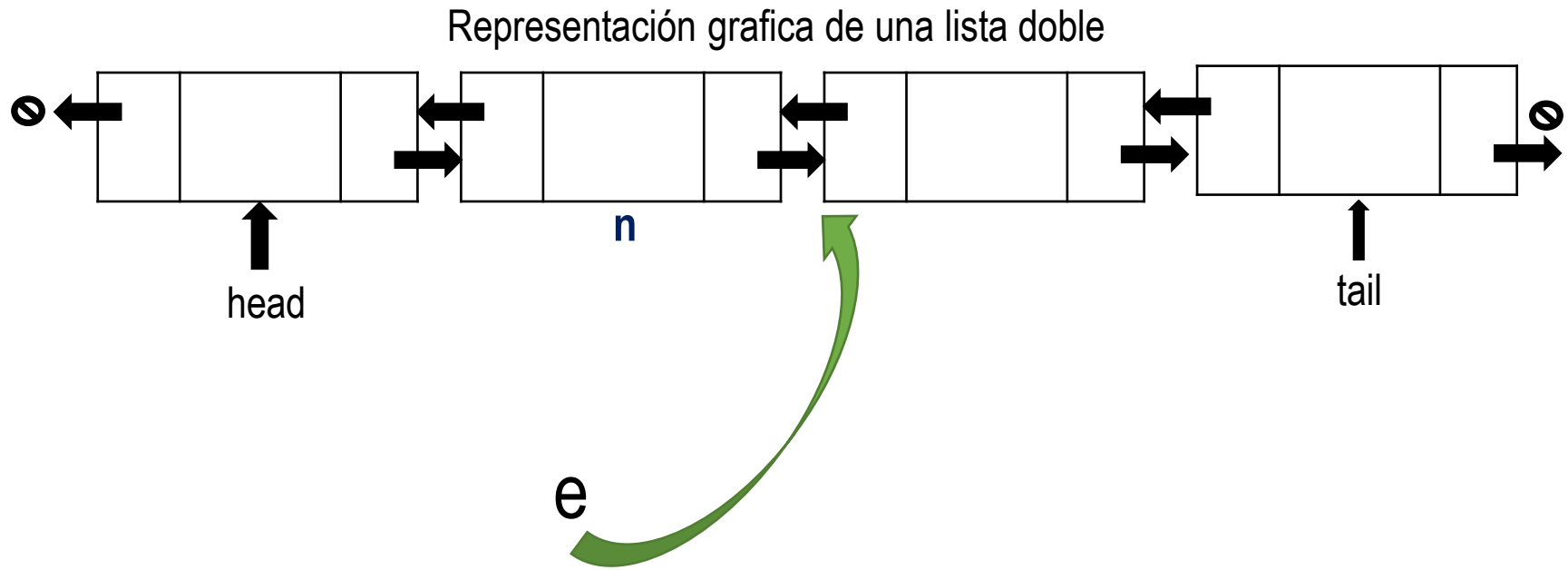
Agregar después de un nodo ->

addAfter(DoubleNode n, Object e)

Estructura de datos

Operaciones básicas con lista doble

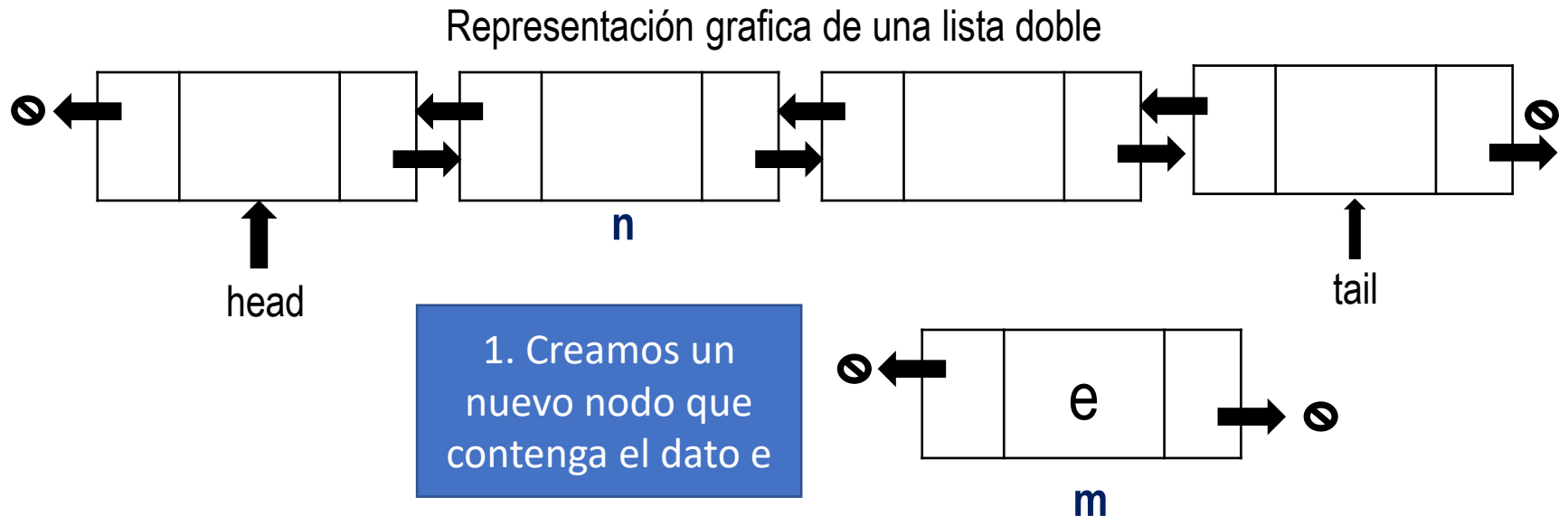
Agregar después de un nodo -> addAfter(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

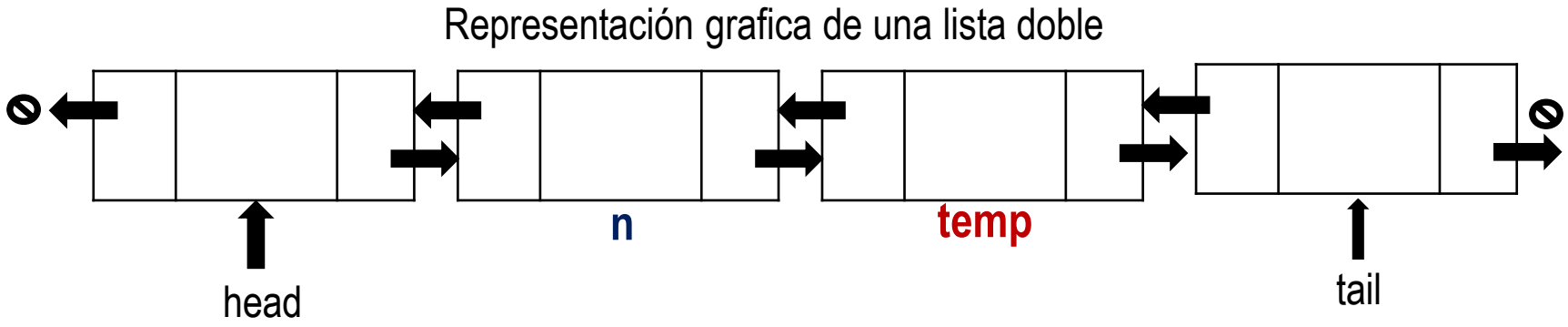
Agregar después de un nodo -> addAfter(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

Agregar después de un nodo -> addAfter(DoubleNode n, Object e)



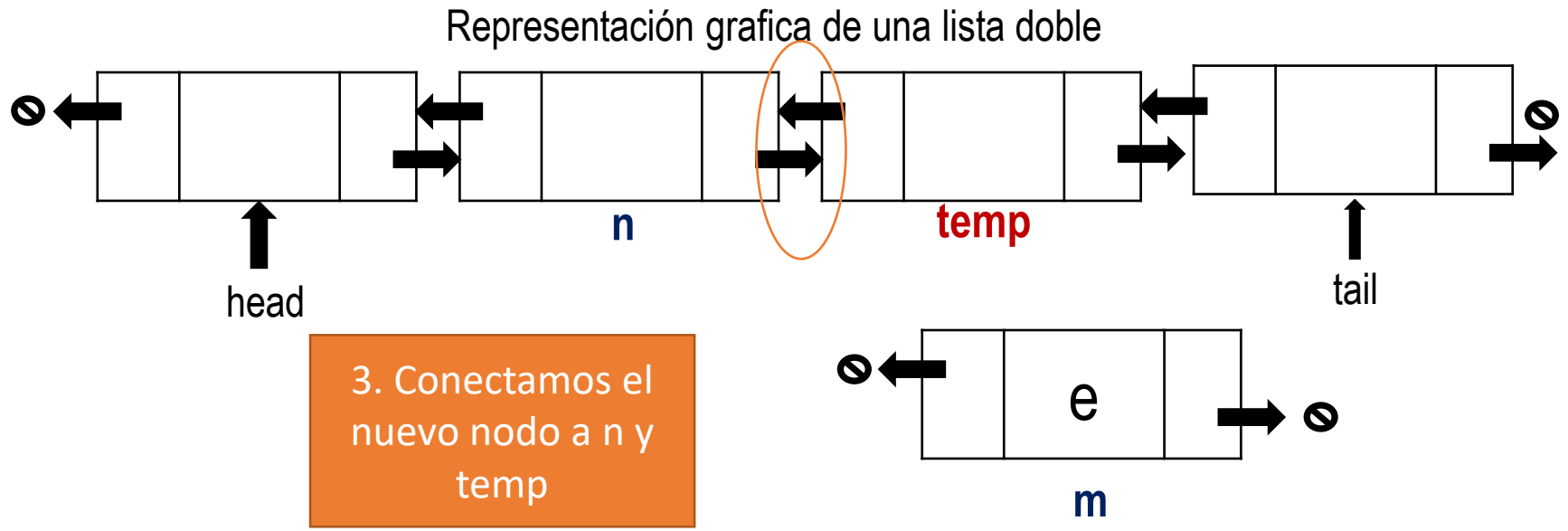
2. Creamos una variable temporal para almacenar el siguiente nodo a n

temp = n.next

Estructura de datos

Operaciones básicas con lista doble

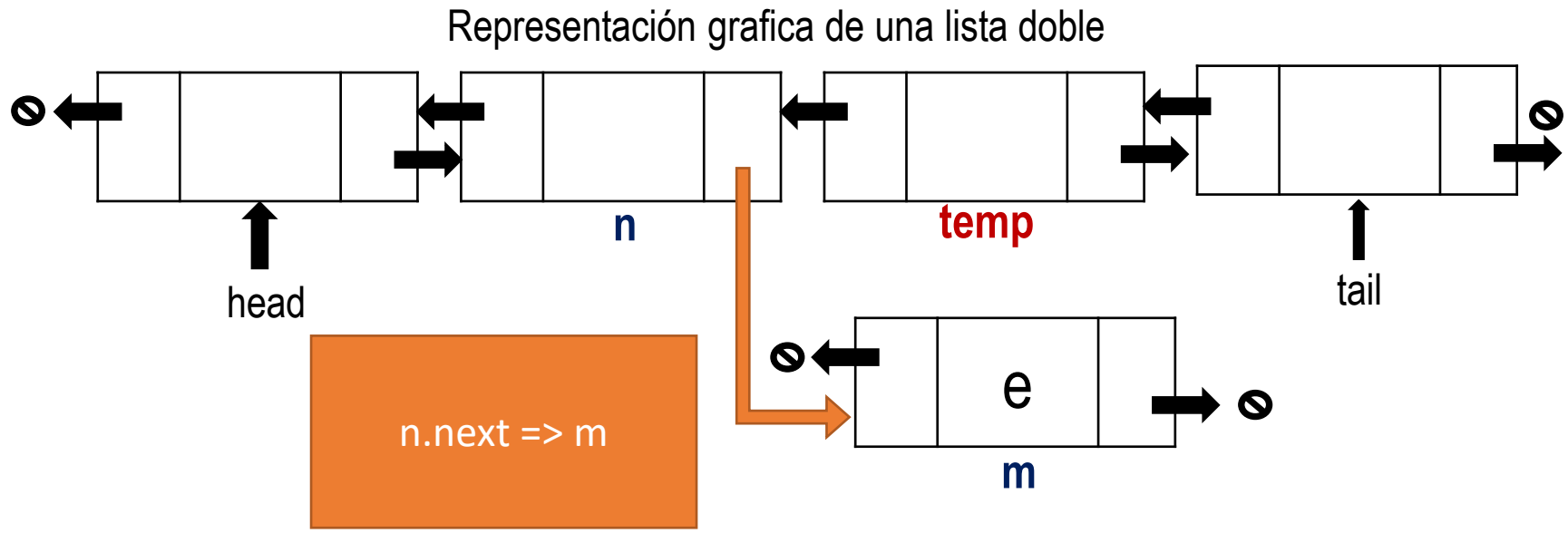
Agregar después de un nodo -> addAfter(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

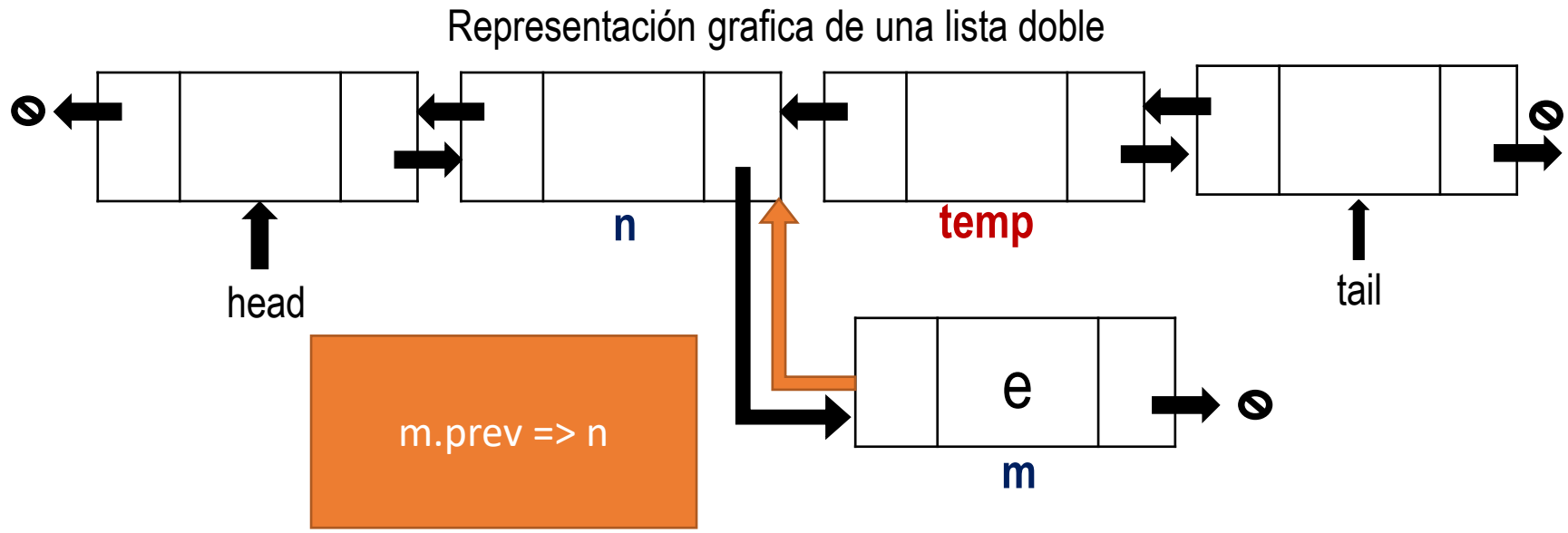
Agregar después de un nodo -> addAfter(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

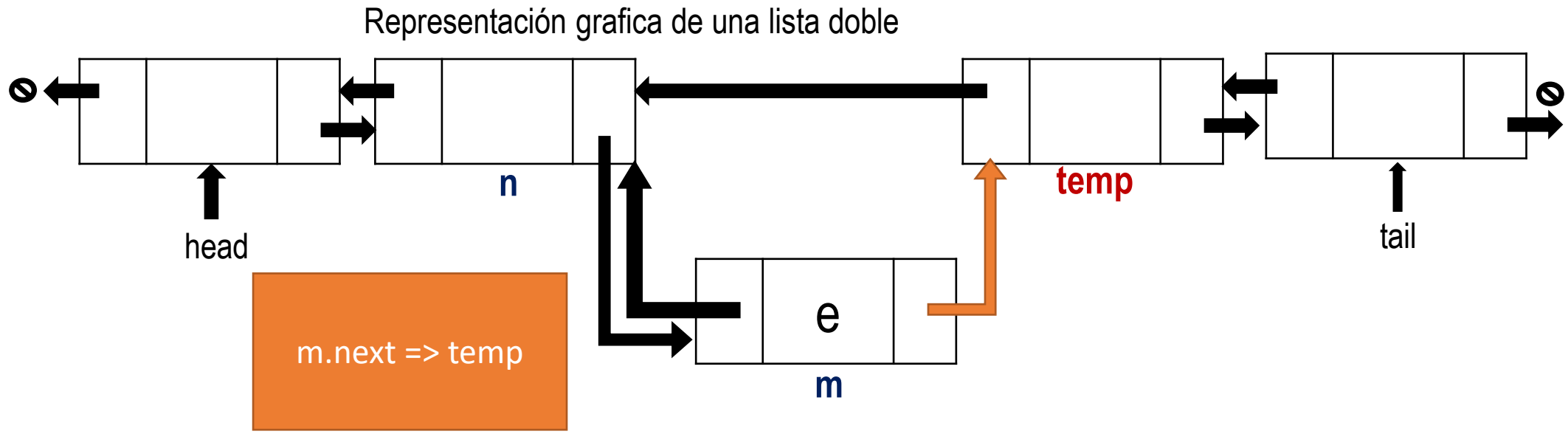
Agregar después de un nodo -> addAfter(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

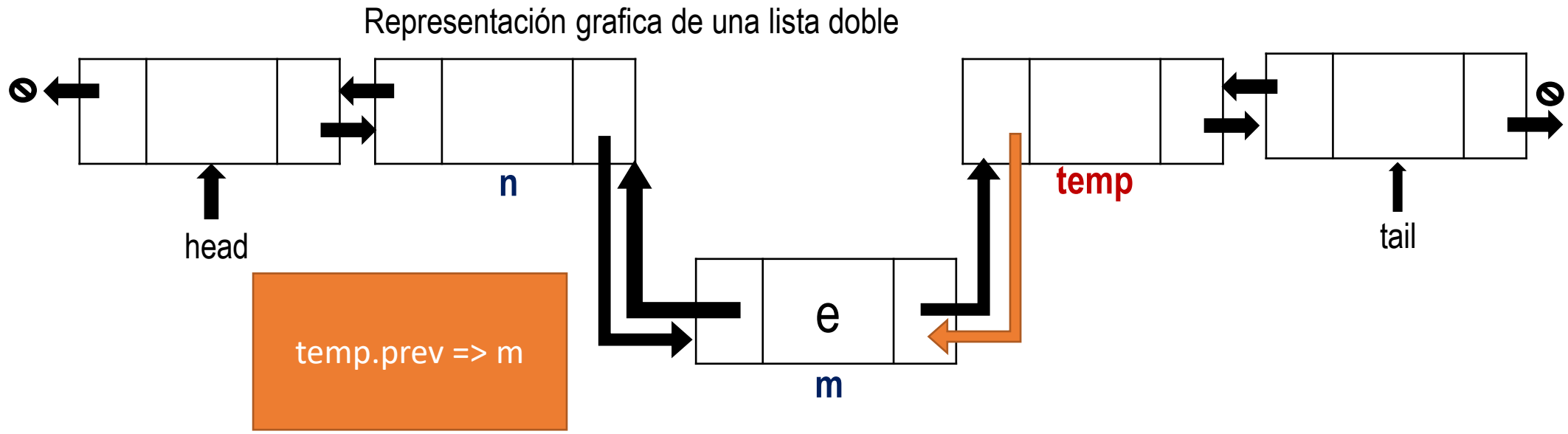
Agregar después de un nodo -> addAfter(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

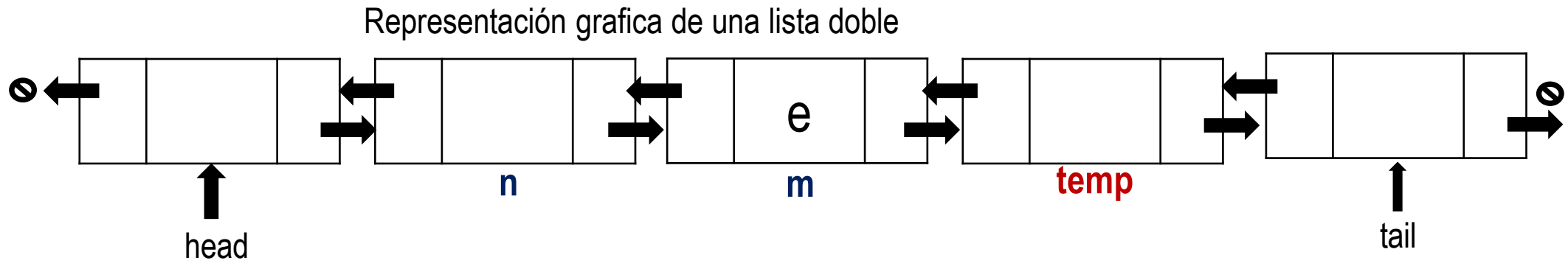
Agregar después de un nodo -> addAfter(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

Agregar después de un nodo -> addAfter(DoubleNode n, Object e)



4. Actualizamos el tamaño
Size++

Estructura de datos

Operaciones básicas con lista doble

```
addAfter(DoubleNode n, Object e)
```

Estructura de datos

Operaciones básicas con lista doble

```
addAfter(DoubleNode n, Object e)
   if n == tail
      addLast(e)
   else
      DoubleNode m = new DoubleNode(e)
```

Creamos un nuevo nodo que contenga el dato e

Estructura de datos

Operaciones básicas con lista doble

```
addAfter(DoubleNode n, Object e)
   if n == tail
      addLast(e)
   else
      DoubleNode m = new DoubleNode(e)
      DoubleNode temp = n.getNext()
```

Creamos una variable temporal para almacenar el siguiente nodo a n

Estructura de datos

Operaciones básicas con lista doble

```
addAfter(DoubleNode n, Object e)
   if n == tail
      addLast(e)
   else
      DoubleNode m = new DoubleNode(e)
      DoubleNode temp = n.getNext()
      n.setNext(m)
      m.setPrev(n)
      m.setNext(temp)
      temp.setPrev(m)
    size++
```

Conectamos el nuevo nodo a n y temp

Estructura de datos

Operaciones básicas con lista doble

```
addAfter(DoubleNode n, Object e)
   if n == tail
      addLast(e)
   else
      DoubleNode m = new DoubleNode(e)
      DoubleNode temp = n.getNext()
      n.setNext(m)
      m.setPrev(n)
      m.setNext(temp)
      temp.setPrev(m)
    size++
```

Actualizamos el tamaño
Size++

Estructura de datos

Operaciones básicas con lista doble

```
addAfter(DoubleNode n, Object e)
   if n == tail
      addLast(e)
   else
      DoubleNode m = new DoubleNode(e)
      DoubleNode temp = n.getNext()
      n.setNext(m)
      m.setPrev(n)
      m.setNext(temp)
      temp.setPrev(m)
    size++
```

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object +addAfter(DoubleNode n, Object e) +addBefore(DoubleNode n, Object e)

Operaciones básicas con lista doble

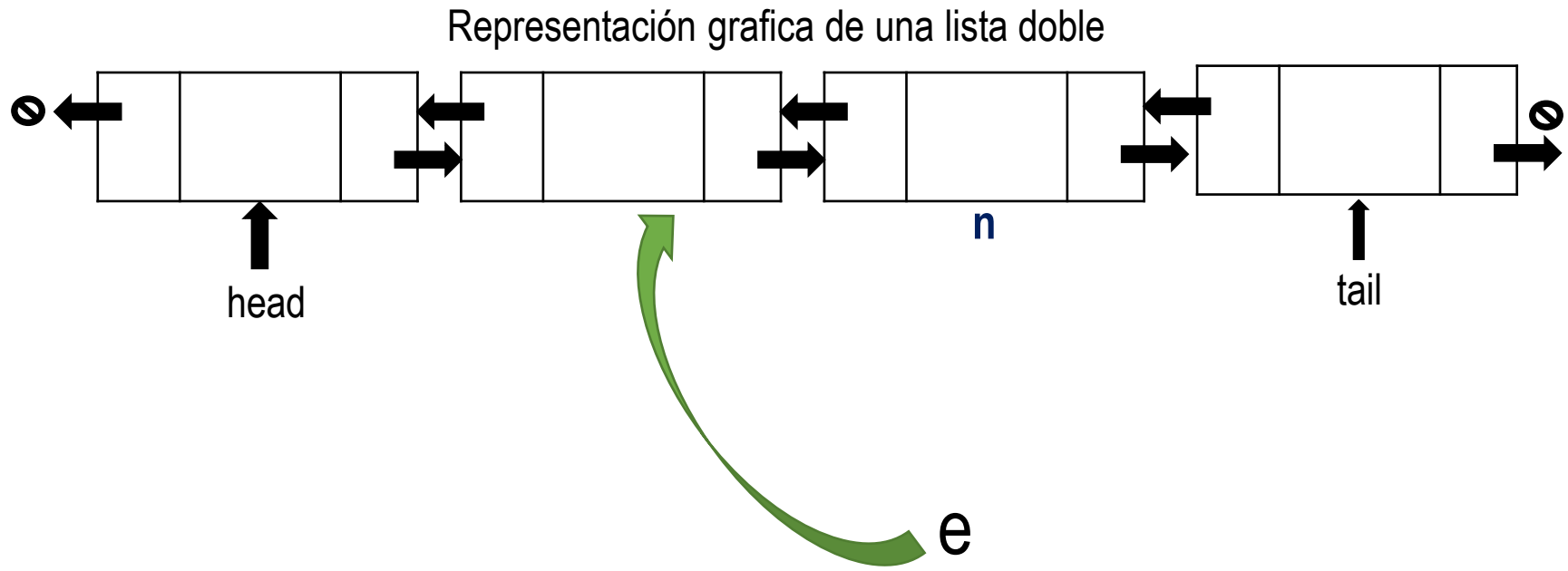
Agregar antes de un nodo ->

addBefore(DoubleNode n, Object e)

Estructura de datos

Operaciones básicas con lista doble

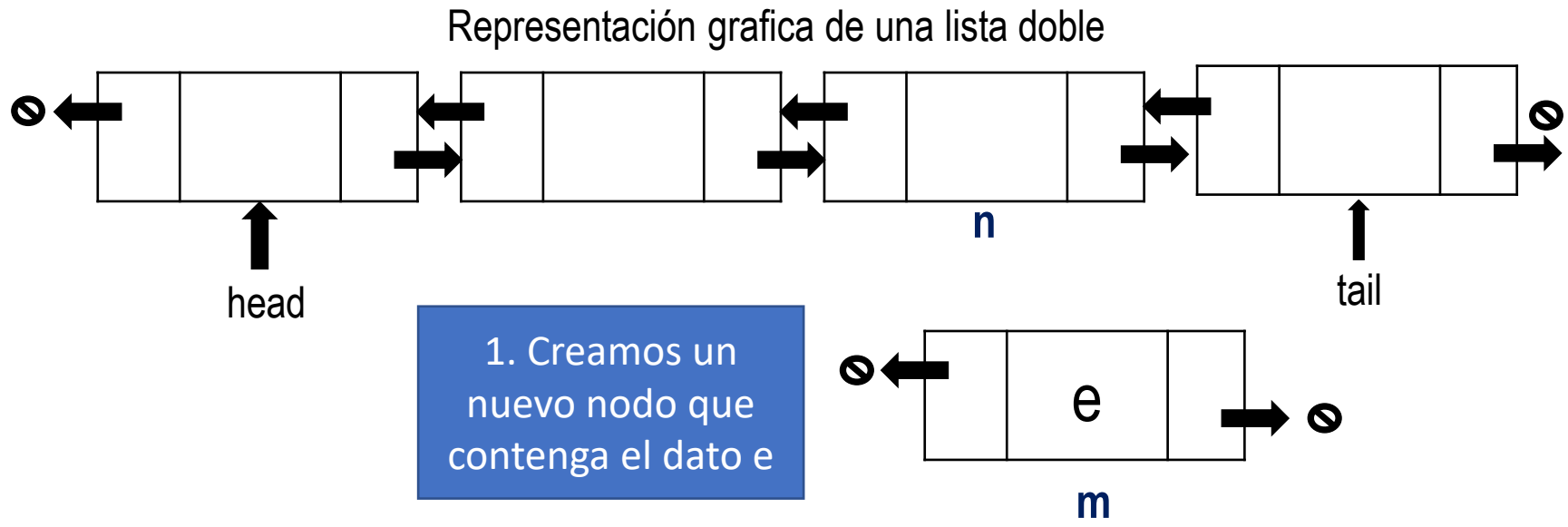
Agregar antes de un nodo -> addBefore(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

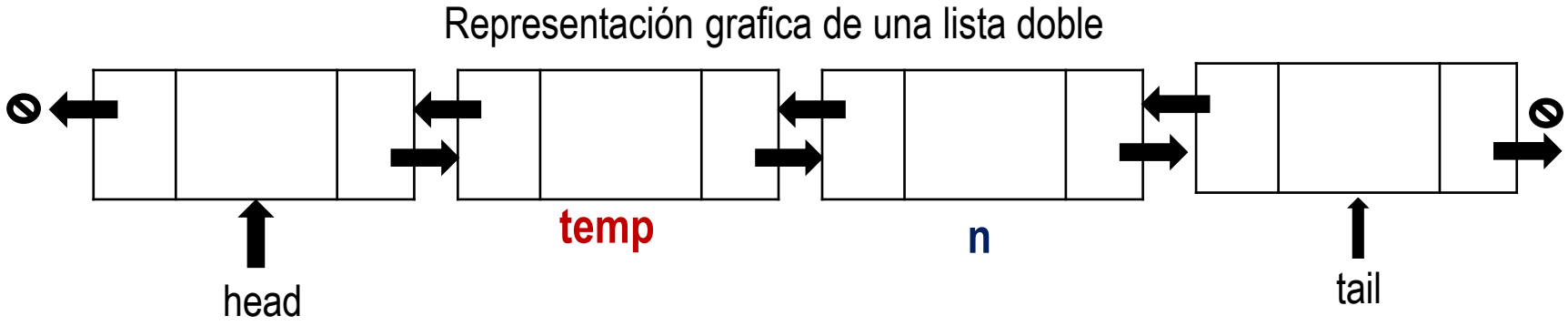
Agregar antes de un nodo -> addBefore(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

Agregar antes de un nodo -> addBefore(DoubleNode n, Object e)



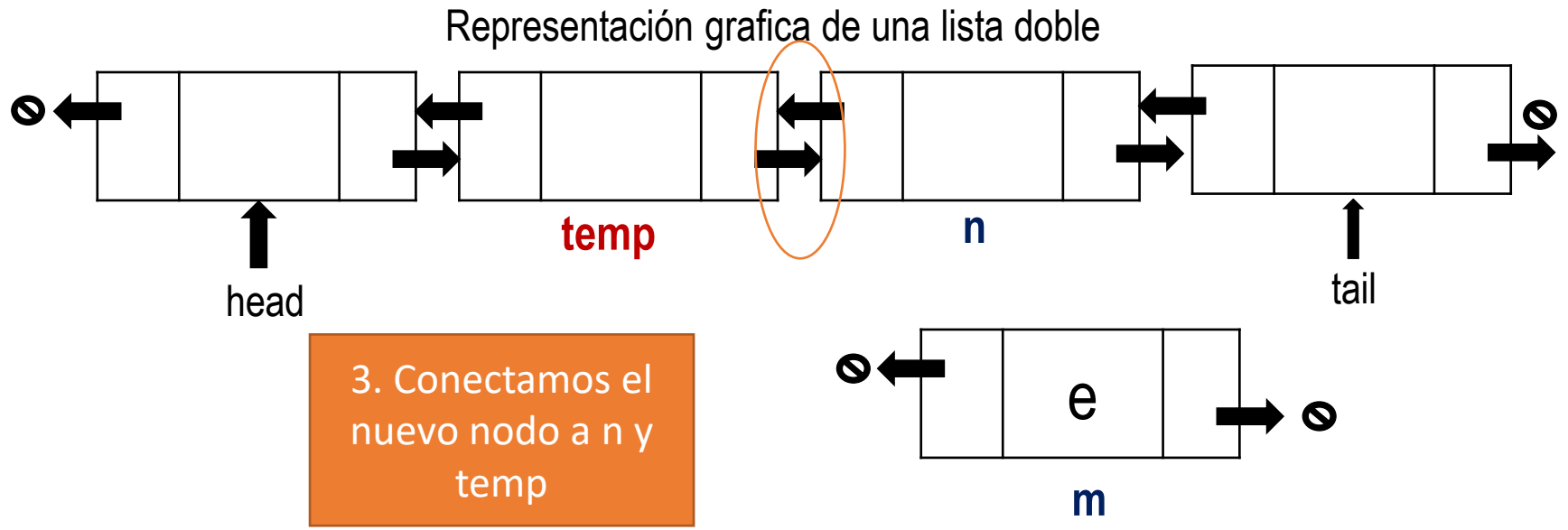
2. Creamos una variable temporal para almacenar el nodo anterior a n

temp = n.prev

Estructura de datos

Operaciones básicas con lista doble

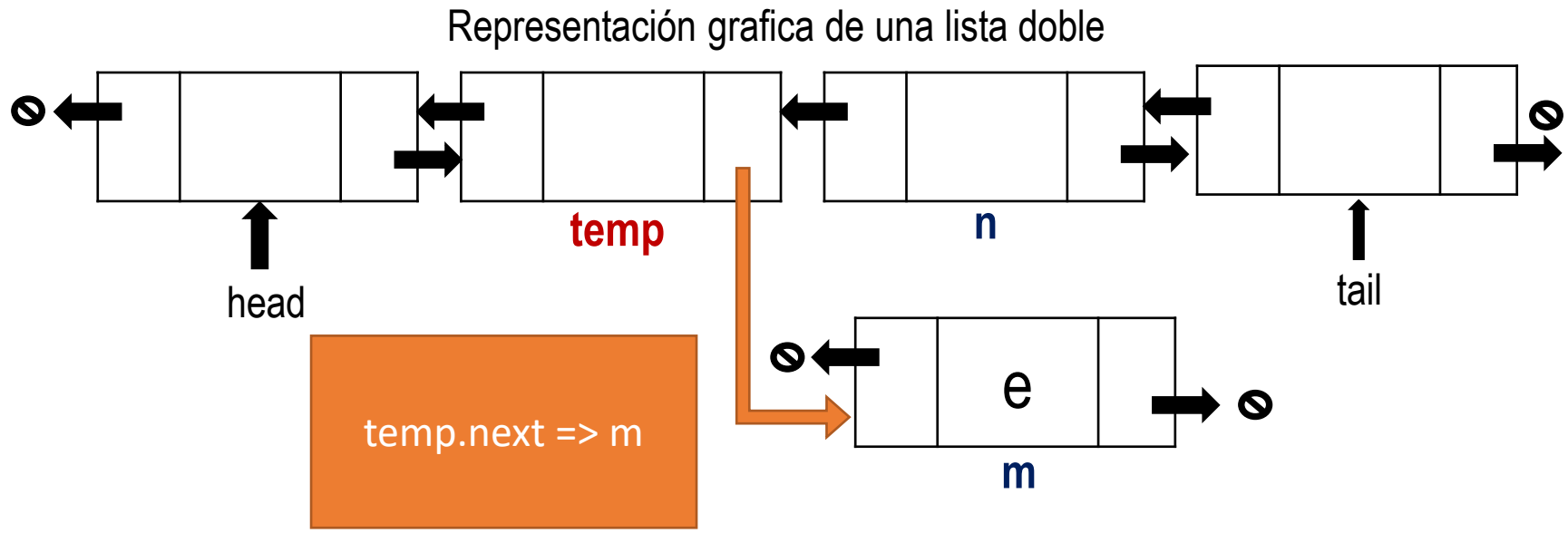
Agregar antes de un nodo -> addBefore(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

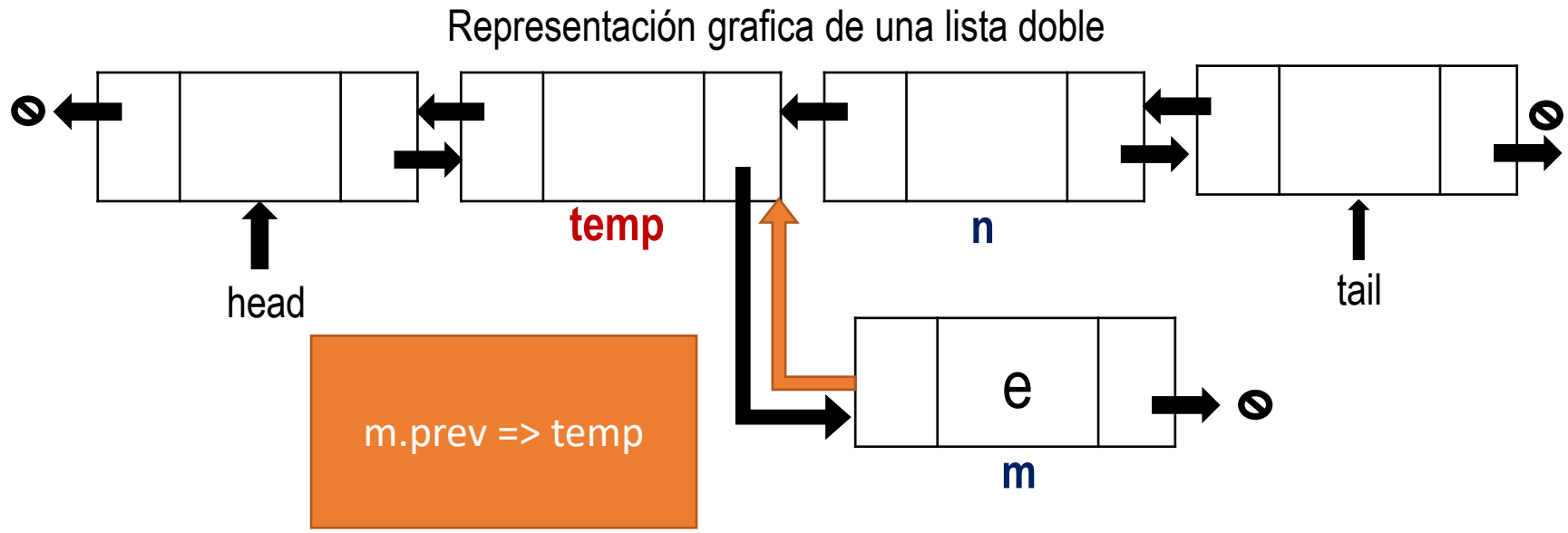
Agregar antes de un nodo -> addBefore(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

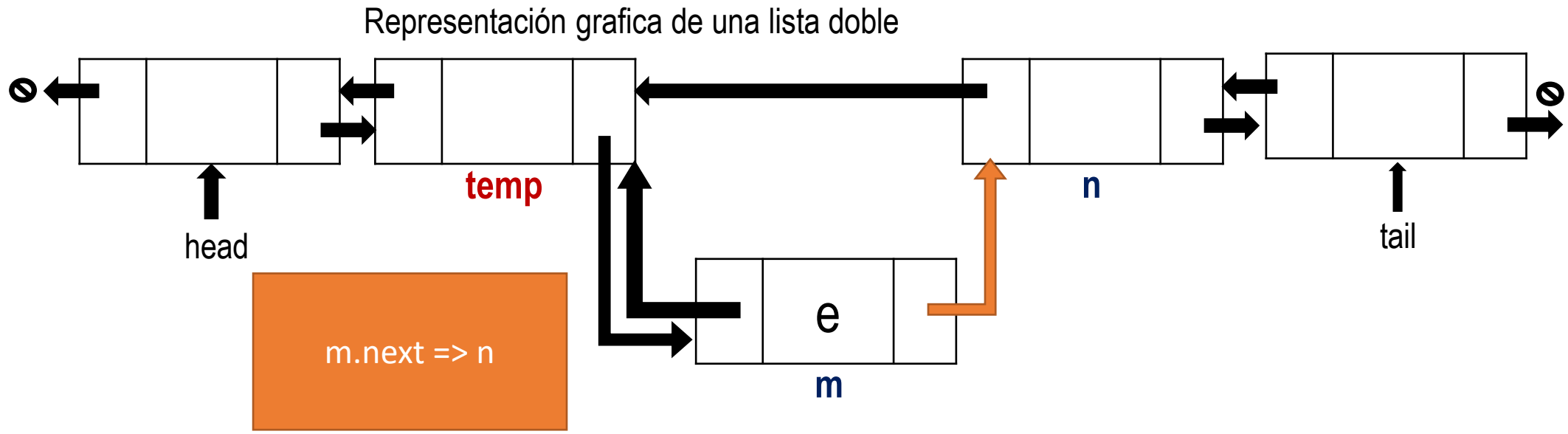
Agregar antes de un nodo -> addBefore(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

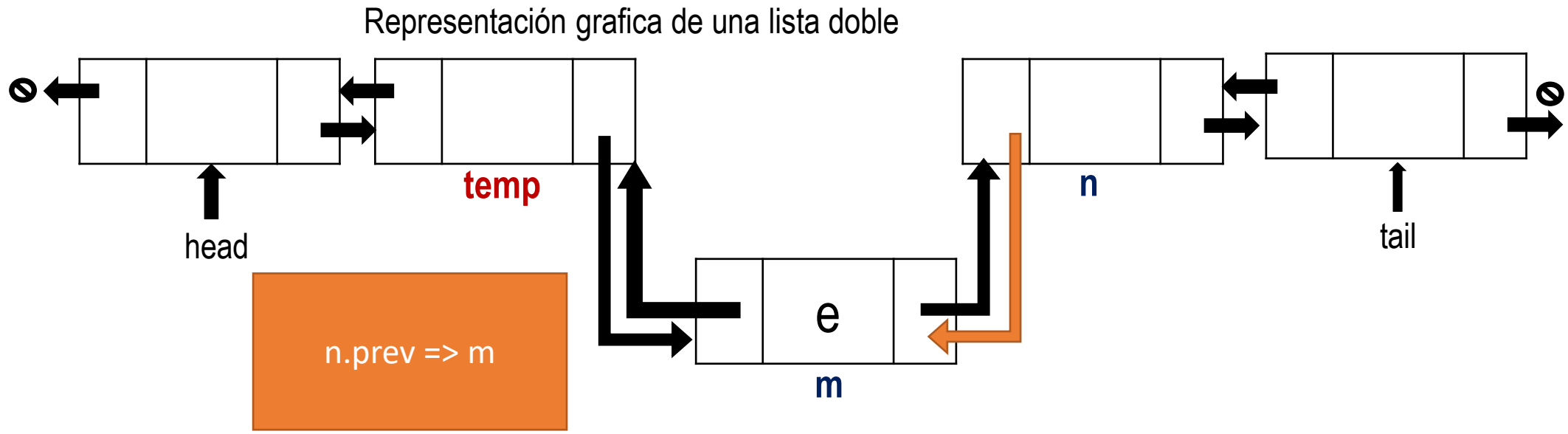
Agregar antes de un nodo -> addBefore(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

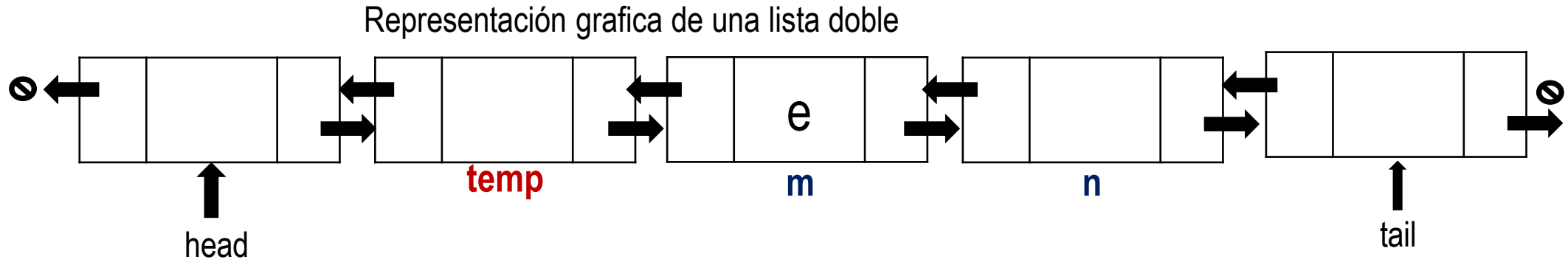
Agregar antes de un nodo -> addBefore(DoubleNode n, Object e)



Estructura de datos

Operaciones básicas con lista doble

Agregar antes de un nodo -> addBefore(DoubleNode n, Object e)



4. Actualizamos el tamaño
Size++

Estructura de datos

Operaciones básicas con lista doble

```
addBefore(DoubleNode n, Object e)
```

Estructura de datos

Operaciones básicas con lista doble

```
addBefore(DoubleNode n, Object e)
    if n == head
        add First(e)
    else
        DoubleNode m = new DoubleNode(e)
```

Creamos un nuevo nodo que contenga el dato e

Estructura de datos

Operaciones básicas con lista doble

```
addBefore(DoubleNode n, Object e)
    if n == head
        add First(e)
    else
        DoubleNode m = new DoubleNode(e)
        DoubleNode temp = n.getPrev()
```

Creamos una variable temporal para almacenar el nodo anterior a n

Estructura de datos

Operaciones básicas con lista doble

```
addBefore(DoubleNode n, Object e)
    if n == head
        add First(e)
    else
        DoubleNode m = new DoubleNode(e)
        DoubleNode temp = n.getPrev()
        temp.setNext(m)
        m.setPrev(temp)
        m.setNext(n)
        n.setPrev(m)
```

Conectamos el nuevo nodo a n y temp

Estructura de datos

Operaciones básicas con lista doble

```
addBefore(DoubleNode n, Object e)
   if n == head
      add First(e)
   else
      DoubleNode m = new DoubleNode(e)
      DoubleNode temp = n.getPrev()
      temp.setNext(m)
      m.setPrev(temp)
      m.setNext(n)
      n.setPrev(m)
    size++
```

Actualizamos el tamaño
Size++

Estructura de datos

Operaciones básicas con lista doble

```
addBefore(DoubleNode n, Object e)
   if n == head
      add First(e)
   else
      DoubleNode m = new DoubleNode(e)
      DoubleNode temp = n.getPrev()
      temp.setNext(m)
      m.setPrev(temp)
      m.setNext(n)
      n.setPrev(m)
    size++
```

Estructura de datos

Clase lista doble

DoubleList
-head: DoubleNode -tail: DoubleNode -size: int
+DoubleList() +size(): int +isEmpty(): Boolean +first(): DoubleNode +last(): DoubleNode +addFirst(Object e) +addLast(Object e) +removeFirst(): Object +removeLast(): Object +remove(DoubleNode n): Object +addAfter(DoubleNode n, Object e) +addBefore(DoubleNode n, Object e)