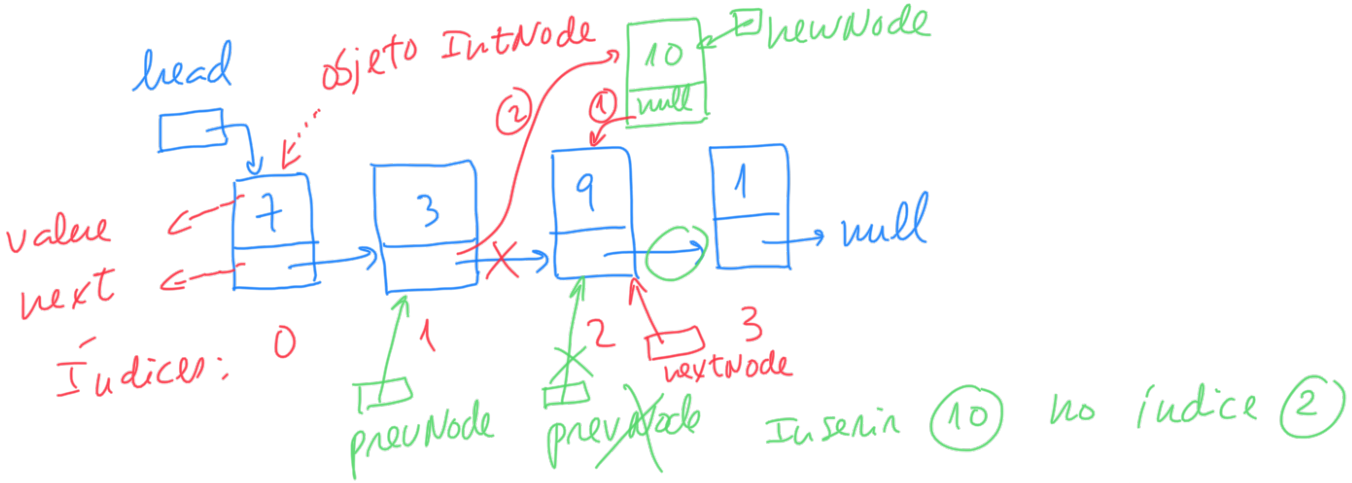
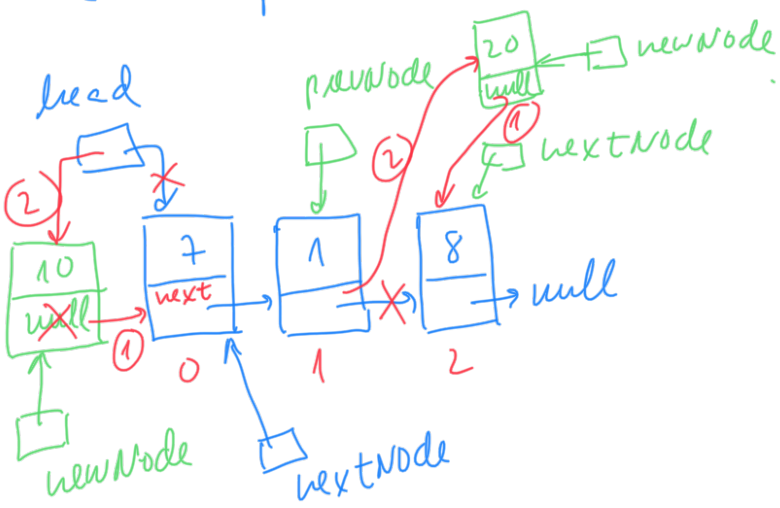
Lista simplesmente ligada

Inserir elemento num dado índice



prevNode deve apontar para o nó anterior onde se pretende inserir o novo nó → este é inserido em prevNode.next

Caso especial: inserir no índice 0 (sem sentinela) 2.

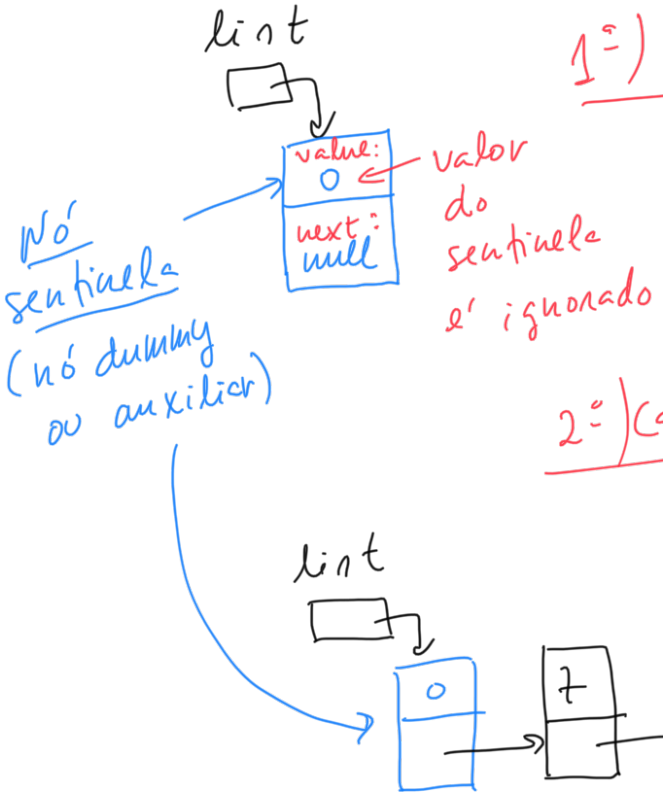


Complexidade temporal no pior caso: $\delta(N)$

```
class SIntList {
...
fun insert(v: Int, index: Int) {
        val newNode = IntNode(v)
        var prevNode : IntNode? = null
        var nextNode : IntNode? = null
        if (index == 0) {
            nextNode = head
        (1) newNode.next = nextNode
        (2) head = newNode
            ++size
            return
        }
        var i = 0
        prevNode = head
        while (i < index - 1) {
            prevNode = prevNode.next
            ++i
        }
        // Inserir newNode à frente de prevNode
        nextNode = prevNode.next
    (1) newNode.next = nextNode
    (2) prevNode.next = newNode
        ++size
}
```

Nós auxiliares que simplificam as operações

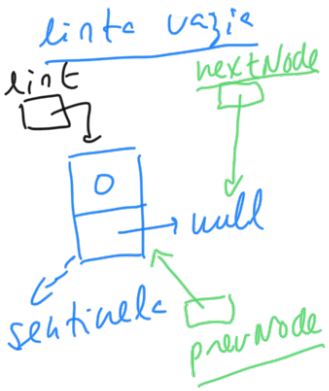
SList com nó sentinela



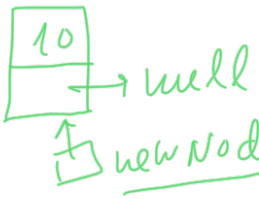
1º) Caso: lista vazia (o nó sentinela não conta)

2º) Caso: lista com elementos: 7, 1, 3

Alteração ao método insert(v, index):



inserir (10) no índice (0):



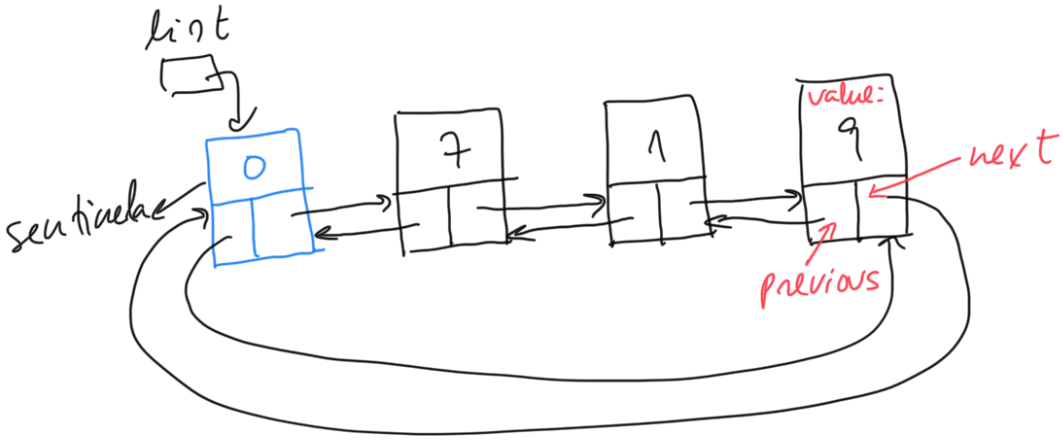
Código igual para o caso da lista estar vazia ou não.

```
insert:
...
prevNode = list
while ( i < index - 1 ){
    prevNode = prevNode.next
    ++i
}
nextNode = prevNode.next
//Inserir
prevNode.next = newNode
newNode.next = nextNode
```

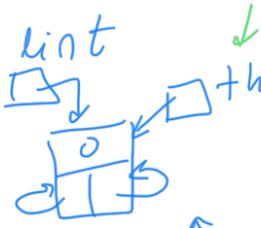
Nota: Como temos ponteiros auxiliares (prevNode e nextNode), podemos ligar por qualquer ordem.

Lista duplamente ligada

Normalmente, unem um nó sentinela e não circulares



só existe no construtor




```
class IntNode {
    var value: Int
    var next : IntNode!
    var previous: IntNode?
    constructor() {
        next = this
        previous = this
    }
    constructor(v: Int) {
        value = v
    }
}
```

construtor chamado para construir o sentinela

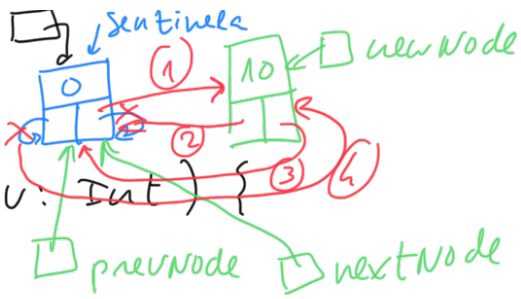
ex: var list = IntNode()

para construir outros nós (previous e next não null)

class DList {

val list = IntNode()

fun addFront (v: Int) {



// como tenho sentinela:

var prevNode = list

var nextNode = list.next // Primeiro nó

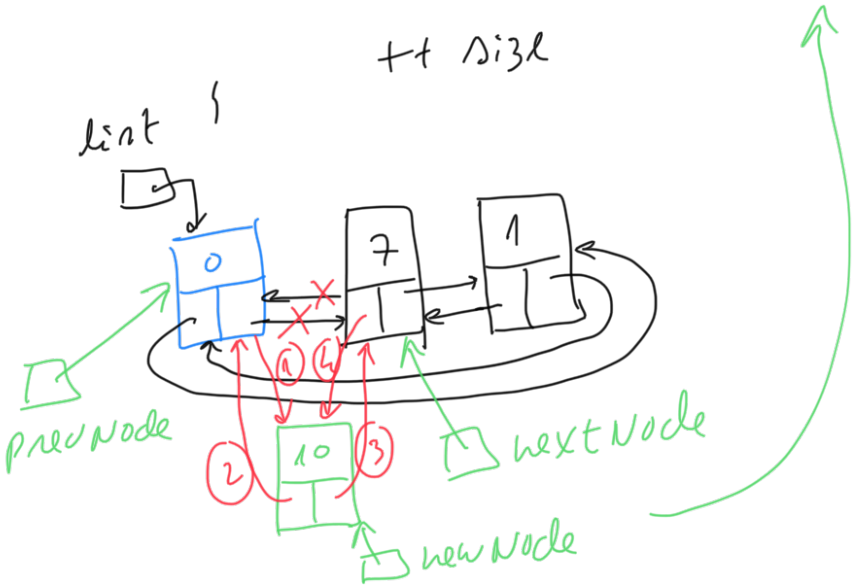
① prevNode.next = newNode

② newNode.previous = prevNode

③ newNode.next = nextNode

④ nextNode.previous = newNode

++ size



T.P.C. addLast, insert (v, index)