SLL - Iterator

- How can we define an iterator for a SLL?
- Remember, an iterator needs a reference to a *current element* from the data structure it iterates over. How can we denote a *current element* for a SLL?
- Remember, for the dynamic array the current element was the index of the element. Can we do the same here?

SLL - Iterator

- In case of a SLL, the current element from the iterator is actually a node of the list.

SLLIterator:
list: SLL
currentElement: ↑ SLLNode

SLL - Iterator - init operation

- What should the *init* operation do?

subalgorithm init(it, sll) **is:**
//pre: sll is a SLL
//post: it is a SLLIterator over sll
    it.sll $\leftarrow$ sll
    it.currentElement $\leftarrow$ sll.head
end-subalgorithm

- Complexity: $\Theta(1)$

SLL - Iterator - getCurrent operation

- What should the *getCurrent* operation do?

```
function getCurrent(it) is:
//pre: it is a SLLIterator, it is valid
//post: getCurrent ← e, e is TElem, the current element from it
//throws: exception if it is not valid
    if it.currentElement = NIL then
        @throw an exception
    end-if
    e ← [it.currentElement].info
    getCurrent ← e
end-function
```

- Complexity: $\Theta(1)$

SLL - Iterator - next operation

- What should the *next* operation do?

subalgorithm next(it) **is:**
//pre: it is a SLLIterator, it is valid
//post: it' is a SLLIterator, the current element from it' refers to the next element
//throws: exception if it is not valid
  if it.currentElement = NIL **then**
    @throw an exception
  end-if
  it.currentElement ← [it.currentElement].next
end-subalgorithm

- Complexity: $\Theta(1)$

SLL - Iterator - valid operation

- What should the *valid* operation do?

```
function valid(it) is:
//pre: it is a SLLIterator
//post: true if it is valid, false otherwise
    if it.currentElement ≠ NIL then
        valid ← True
    else
        valid ← False
    end-if
end-subalgorithm
```

- Complexity: $\Theta(1)$