- In STL (C++), TPosition is represented by an iterator.
- For example - vector:

  iterator insert(iterator position, const value_type& val)
  - Returns an iterator which points to the newly inserted element

  iterator erase (iterator position);
  - Returns an iterator which points to the element after the removed one

- If we consider that TPosition is an Iterator (similar to C++) we can have an *IteratedList*.
- In case of an *IteratedList* the operations that take as parameter a position use an Iterator (and the position is the current element from the Iterator)
- Operations *valid*, *next*, *previous* no longer exist in the interface of the List (they are operations for the Iterator).
- init(l)
  - **descr:** creates a new, empty list
  - **pre:** true
  - **post:** $l \in \mathcal{L}$, l is an empty list
- first(l)
  - **descr:** returns an Iterator set to the first element
  - **pre:** $l \in \mathcal{L}$
  - **post:** $first \leftarrow it \in Iterator$

$$it = \begin{cases} \text{an iterator set to the first element} & \text{if } l \neq \emptyset \\ \text{an invalid iterator} & otherwise \end{cases}$$

- last(l)
  - **descr:** returns an Iterator set to the last element
  - **pre:** $l \in \mathcal{L}$
  - **post:** $last \leftarrow it \in Iterator$

$$\text{it} = \begin{cases} \text{an iterator set to the last element} & \text{if l} \neq \emptyset \\ \text{an invalid iterator} & otherwise \end{cases}$$

- getElement(l, it)
  - **descr:** returns the element from the position denoted by an Iterator
  - **pre:** $l \in \mathcal{L}, it \in Iterator$, $valid(it)$
  - **post:** $getElement \leftarrow e$, $e \in TElem$, e = the element from l from the current position
  - **throws:** exception if *it* is not valid

- position(l, e)
  - **descr:** returns an iterator set to the first position of an element
  - **pre:** $l \in \mathcal{L}, e \in TElem$
  - **post:**

$$position \leftarrow it \in Iterator$$

$$\text{it} = \begin{cases} \text{an iterator set to the first position of element e from l} & \text{if } e \in l \\ \text{an invalid iterator} & otherwise \end{cases}$$

- setElement(l, it, e)
  - **descr:** replaces the element from the position denoted by an Iterator with another element
  - **pre:** $l \in \mathcal{L}, it \in Iterator, e \in TElem$, valid(it)
  - **post:** $l' \in \mathcal{L}$, the element from the position denoted by *it* from l' is e, $setElement \leftarrow el$, $el \in TElem$, *el* is the element from the current position from *it* from *l* (returns the previous value from the position)
  - **throws:** exception if *it* is not valid

- addToBeginning(l, e)
  - **descr:** adds a new element to the beginning of a list
  - **pre:** $l \in \mathcal{L}, e \in TElem$
  - **post:** $l' \in \mathcal{L}$, l' is the result after the element e was added at the beginning of l
- addToEnd(l, e)
  - **descr:** inserts a new element at the end of a list
  - **pre:** $l \in \mathcal{L}, e \in TElem$
  - **post:** $l' \in \mathcal{L}$, l' is the result after the element e was added at the end of l
- addToPosition(l, it, e)
  - **descr:** inserts a new element at a given position specified by the iterator (it is the same as *addAfterPosition*)
  - **pre:** $l \in \mathcal{L}, it \in Iterator, e \in TElem$, $valid(it)$
  - **post:** $l' \in \mathcal{L}$, l' is the result after the element e was added in l at the position specified by it
  - **throws:** exception if it is not valid
- remove(l, it)
  - **descr:** removes an element from a given position specfied by the iterator from a list
  - **pre:** $l \in \mathcal{L}, it \in Iterator$, $valid(it)$
  - **post:** $remove \leftarrow e$, $e \in TElem$, e is the element from the position from l denoted by it, $l' \in \mathcal{L}$, l' = l - e.
  - **throws:** exception if it is not valid
- remove(l, e)
  - **descr:** removes the first occurrence of a given element from a list
  - **pre:** $l \in \mathcal{L}, e \in TElem$
  - **post:**

$$remove \leftarrow \begin{cases} true & \text{if } e \in l \text{ and it was removed} \\ false & otherwise \end{cases}$$

- search(l, e)
  - **descr:** searches for an element in the list
  - **pre:** $l \in \mathcal{L}, e \in TElem$
  - **post:**

$$search \leftarrow \begin{cases} true & \text{if } e \in l \\ false & otherwise \end{cases}$$

- isEmpty(l)
  - **descr:** checks if a list is empty
  - **pre:** $l \in \mathcal{L}$
  - **post:**

$$isEmpty \leftarrow \begin{cases} true & \text{if } l = \emptyset \\ false & otherwise \end{cases}$$

- size(l)
  - **descr:** returns the number of elements from a list
  - **pre:** $l \in \mathcal{L}$
  - **post:** $size \leftarrow$ the number of elements from l
- destroy(l)
  - **descr:** destroys a list
  - **pre:** $l \in \mathcal{L}$
  - **post:** l was destroyed