**Beschreibung:**

Die Funktion previous fuhrt den current Iterator auf das vorige Element, falls dieses existiert und wirft eine Exception andernfalls.

**Pre:**

smmit ist ein SortedMultiMap Iterator  
reverseStackSize ist ein Integer  
stackSize ist ein Integer  
stackCapacity ist ein Integer  
current ist ein Pointer auf BSTNode  
stack ist ein Stack  
reverseStack ist ein Stack

**Post:** current’ = vorherige Element (falls dieses existiert)

**Throws:** eine Exception, falls es keinen vorherigen Element gibt (smmit zeigt auf dem kleinsten Element)

**Pseudocode:**

**subalgorithm** next() **is:**

**if** smmit.valid = false **then**:  
 @ wirft eine Exception  
 **end-if**

**if** smmit.reverseStack = smmit.reverseStackCapacity **then:**  
 **@ resize reverseStack  
 end-if**

smmit.[reverseStack[smmit.reverseStackSize]] <- smmit.current  
 smmit.reverseStackSize <- smmit.reverseStackSize + 1

**if** smmit.current.right != NIL **then:**  
 smmit.current <- smmit.current.right  
 while smmit.current != NIL execute:  
 **if** smmit.stackSize = smmit.stackCapacity **then:**  
 **@ resize Stack**  
 **end-if**  
 smmit.stack[smmit.stackSize] <- smmit.current  
 smmit.stackSize <- smmit.stackSize + 1  
 smmit.current <- smmit.current.left  
 **end-while**  
 **end-if**

**if** smmit.stackSize > 0 **then:**  
 smmit.stackSize <- smmit.stackSize - 1  
 smmit.current <- smmit.stack[smmit.stackSize]

**else**  
 smmit.current <- NIL  
**end-if**

**end-subalgorithm**

**subalgorithm** previous() **is**:

**if** smmit.reverseStackSize = 0 **then:  
 @ wirft eine Exception  
end-if**

**if** smmit.stackSize = smm.stackCapacity **then:  
 @ resize stack  
end-if**

smm.[stack[smm.stackSize]] <- smm.current  
smm.stackSize <- smm.stackSize + 1  
smmit.reverseStackSize <- smmit.reverseStackSize - 1  
smm.current <- smmit.[reverseStack[smmit.reverseStackSize]]

**end-subalgorithm**

**Komplexitat: Theta(1)**