ALGORITHMEN ZUM SUCHBAUM

```
TOrderedTree=class(TBinTree)
private
   function IsoliereGroesstesElement(var grElem: TOrderedTree): TOrderedTree;
   function EchtLoeschen: TOrderedTree;
public
   function insertItem(pItem:TItem): TOrderedTree;
   function searchItem(pItem:TItem):TItem;
   function removeItem(pItem:TItem): TOrderedTree;
   function getSortedList:TLinList;
end;
```

ALGORITHMUS ***searchItem***

Outputobjekte:	result:	TItem	*{ als Funktionsergebnis }*
Inputobjekte:	pItem:	TItem	
Hilfsobjekte:	Teilbaum:	TOrderedTree	

- Falls *isEmpty* true ist,
  - dann • *result* ← nil
  - sonst • Falls *getRootItem.isEqual(* pItem *)*,
    - dann • *result* ← *getRootItem*
    - sonst • Falls *getRootItem.isGreater(*pItem *)*,
      - dann • Teilbaum ← *getLeftTree*
        - • *result*← Teilbaum.***searchItem(pItem)***
      - sonst • Teilbaum ← *getRightTree*
        - • *result*← Teilbaum.***searchItem(pItem)***

ALGORITHMUS ***insertItem***

Outputobjekte:	result:	TOrderedTree	*{ als Funktionsergebnis }*
Inputobjekte:	pItem:	TItem	
Hilfsobjekte:	Teilbaum:	TOrderedTree	

- Falls *isEmpty* true ist,
  - dann • *result* ← TMyOrderedTree.Create(pItem)
  - sonst • Falls *getRootItem.isEqual(*pItem *)*,
    - dann • *result* ← *Self*
    - sonst • Falls *getRootItem.isGreater(*pItem *)*,
      - dann • Teilbaum ← *getLeftTree*
        - • Teilbaum.***insertItem(pItem)***
        - • *setLefTreet (* Teilbaum *)*
        - • *result*← *Self*
      - sonst • Teilbaum ← *getRightTree*
        - • Teilbaum.***insertItem(pItem)***
        - • *setRightTree(* Teilbaum *)*
        - • *result*← *Self*

ALGORITHMUS ***removeItem***

Outputobjekte:	result:	TOrderedTree	*{ veränderter Suchbaum }*
Inputobjekte:	pItem:	TItem	
Hilfsobjekte:	Teilbaum:	TOrderedTree	

- Falls *isEmpty* true ist,
  - dann
    - *result* ← *Self*
  - sonst
    - Falls *getRootItem.isEqual(* pItem*)*,
      - dann
        - *result* ← ***Echtloeschen***
      - sonst
        - Falls *getRootItem.isGreater(* pItem *)*,
          - dann
            - Teilbaum ← *getLeftTree*
            - Teilbaum.***deleteItem(pItem)***
            - *setLeftTree (* Teilbaum *)*
            - *result* ← *Self*
          - sonst
            - Teilbaum ← *getRightTree*
            - Teilbaum.***deleteItem(pItem)***
            - *setRightTree (* Teilbaum *)*
            - *result* ← *Self*

ALGORITHMUS ***EchtLoeschen***

Outputobjekte:	EchtLoeschen:	TOrderedTree	*{ veränderter Suchbaum }*
Hilfsobjekte:	Teilbaum:	TOrderedTree	
	GroesstesElement	TOrderedTree	*{ größtes El. im linken Teilb. }*

- Falls *getRightTree* = nil ist,
  - dann
    - *result* ← *getLeftTree*
    - Free
  - sonst
    - Falls *getLeftTree* = nil ist,
      - dann
        - *result* ← *getRightTree*
        - Free
      - sonst
        - Teilbaum ← *getLeftTree.*
        - Teilbaum ← Teilbaum.***IsoliereGroesstesElement(GroesstesElement)***
        - *setLeftTree (* Teilbaum *)*
        - GroesstesElement.*setLeftTree(getLeftTree)*
        - GroesstesElement.*setRightTree(getRightTree)*
        - *setLeftTree (* NIL *)*
        - *setRightTree (* NIL *)*
        - *Free*
        - *result*← GroesstesElement

ALGORITHMUS ***IsoliereGroesstesElement***

Outputobjekte:	IsoliereGroesstesElement:	TOrderedTree	*{ veränderter Suchbaum }*
	GroesstesElement:	TOrderedTree	*{ natürlich ein Blatt }*
Hilfsobjekte:	Teilbaum:	TOrderedTree	

- Falls *getRightTree* = nil ist,
  - dann
    - *result* ← *getLeftTree*
    - GroesstesElement ← *Self*
  - sonst
    - Teilbaum ← *getRightTree*
    - Teilbaum ← Teilbaum.***IsoliereGroesstesElement(GroesstesElement)***
    - *setRightTree (* Teilbaum *)*
    - *result* ← *Self*