UML-диаграмма B_Tree

BNode

- + keys[2t-1]: int
- + children[2t+1]: BNode*
- + parent: BNode*
 + keys_count: int
 + child_count: int
- + leaf: bool;
- + BNode(key: int): constructor

BTree

- + root: BNode*
- + BTree(): constructor + ~BTree(): destructor
- + insert(key: int): void
 + search(key: int): bool
 + remove(key: int): void
- insert in node(key: int, node: BNode*): void
- sort(node: BNode*): void
- split_node(node: BNode*): void
- delete_node(node: BNode*): void
- search_key(key: int, node: BNode*): bool
- remove_any(key: int, node: BNode*): void
- remove_node(key: int, node: BNode*): void
- remove_leaf(key: int, node: BNode*): void
- left_connect(node: BNode*, neighbor: BNode*): void
- right_connect(node: BNode*, neighbor: BNode*): void
- repair(node: BNode*): void

UML-диаграмма Binary_Tree

BinaryTreeNode

- + key: int + data: int*
- + left_child: BinaryTreeNode*
 + right_child: BinaryTreeNode*
- + BinaryTreeNode(key: int, data: int*): constructor



- + root: BinaryTreeNode*
- + BinaryTree(): constructor
- + PRINT(root: BinaryTreeNode*): void
- + ADD(key: int, data: int*): int
- + FIND(root: BinaryTreeNode*, key: int): BinaryTreeNode*
- + REMOVE(root: BinaryTreeNode*, key: int): BinaryTreeNode*
- parent_node(root: BinaryTreeNode*, key: int): BinaryTreeNode*
- min_node(root: BinaryTreeNode*): BinaryTreeNode*
- max_node(root: BinaryTreeNode*): BinaryTreeNode*
- find_parent_for_new_node(root: BinaryTreeNode*, key: int): BinaryTreeNode*