Algorytmy i struktury danych

Lista zadań 7 - B-drzewa

Zadanie 1

Zapisz warunki jakie muszą spełniać klucze drzewa BST.

- Klucze w lewym poddrzewie są mniejsze od klucza w danym węźle.
- Klucze w prawym poddrzewie są większe lub równe kluczowi w danym węźle.

Zadanie 2

Napisz procedurę node* find(node* tree, int x), która zwraca wskaźnik na węzeł zawierający x lub NULL, jeśli nie ma takiego węzła.

```
node *find(node *tree, int x) // wyszukiwanie klucza (nierekurencyjne)
{
    while (tree->key != x)
        if (tree == nullptr) return nullptr;
        if (tree->key > x) tree = tree->left;
        else
            tree = tree->right;
    return tree;
}

node *find(node *tree, int x) // wyszukiwanie klucza (rekurencyjne)
{
    if (tree == nullptr) return nullptr;
    if (tree->key == x) return tree;
    if (tree->key > x) return find_recursive(tree->left, x);
    return find(tree->right, x);
}
```

Zadanie 3

Napisz procedurę void insert(node *tree, int x) (dodaje do drzewa tree klucz x).

```
void insert(node *&tree, int x) // wstawianie (rekurencyjnie)
{
    if (tree == nullptr)
        tree = new node(x);

    else if (tree->key > x)
        insert(tree->left, x);
    else
        insert(tree->right, x);
}
```

Zadanie 4

- Narysuj drzewo BST reprezentowane przez listę par: 1:2, 2:4, 3:2, 4:5, 6:7, 7:9, 8:7, 9:5.
- Wypisz jego klucze w porządku: INORDER, PREORDER, POSTORDER.

```
void inOrder(node *tree)
{
    if (tree != nullptr)
    {
        inOrder(tree->left);          // wypisze klucze z lewego poddrzewa
        std::cout << tree->key;  // wypisz klucz z wezla
        inOrder(tree->right);         // wypisze klucze z lewego poddrzewa
    }
}
```

```
void preOrder(node *tree)
{
    if (tree != nullptr)
    {
        std::cout << tree->key;  // najpierw wypisz klucz z wezla
        preOrder(tree->left);         // a potem klucze z lewego
        preOrder(tree->right);        // i prawego poddrzewa
    }
}
```

```
void postOrder(node *tree)
{
    if (tree != nullptr)
    {
        postOrder(tree->left);        // wypisz klucze z poddrzewa lewego,
        postOrder(tree->right);       // potem z prawego,
        std::cout << tree->key;  // a na koncu klucz z wezla
    }
}
```