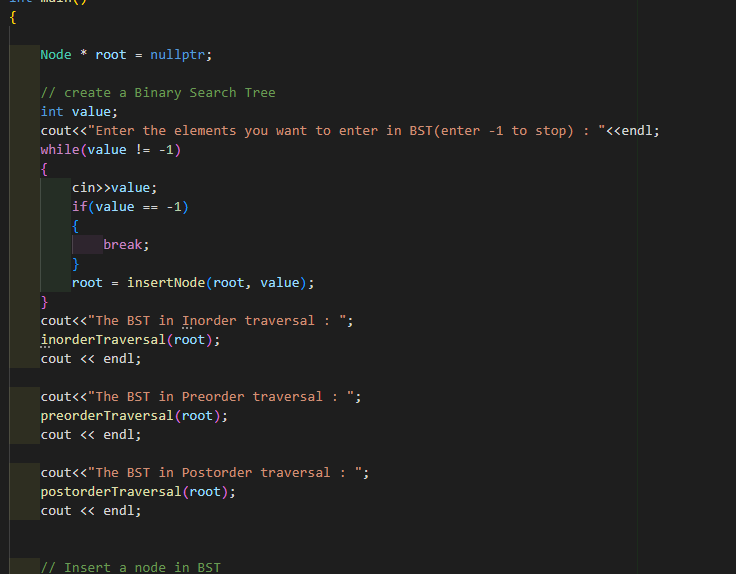
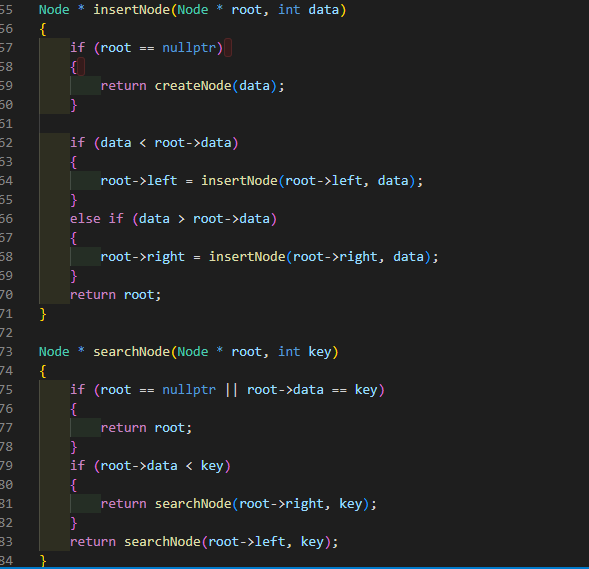
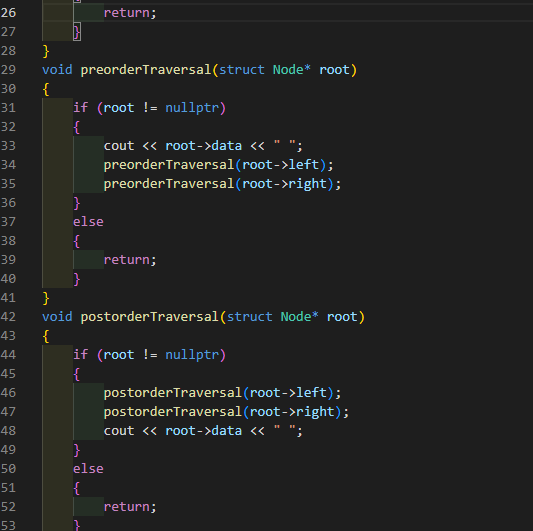
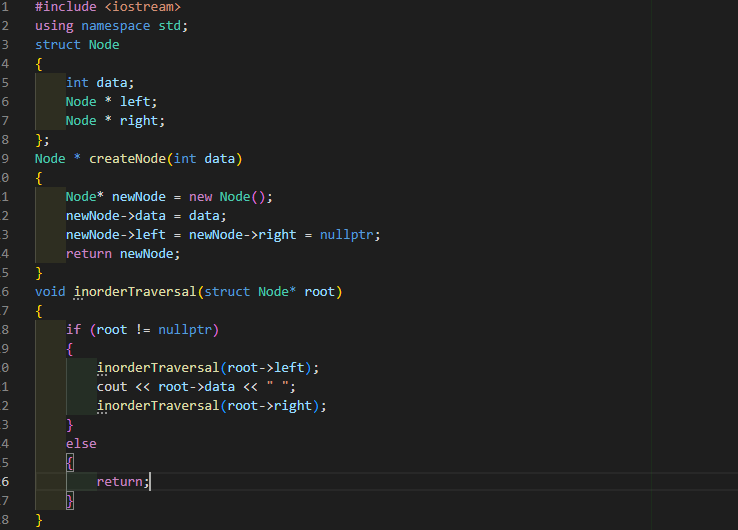
BINARY SEARCH TREE

A Binary Search Tree (BST) is a data structure that maintains sorted order of elements, enabling efficient insertion, deletion, and lookup operations. This project aims to implement a BST and demonstrate its functionalities and features through a comprehensive software application.

**Features and Functionalities**

1. **Node Structure**:
   * Each node contains a key (or value) and references to its left and right children.
   * The left child contains values less than the parent node's key.
   * The right child contains values greater than the parent node's key.
2. **Basic Operations**:
   * **Insertion**: Adding a new node to the BST while maintaining its properties.
   * **Deletion**: Removing a node and reconfiguring the tree to preserve the BST properties.
   * **Search**: Finding a node with a specific key.
   * **Traversal**: Visiting all nodes in a specific order (in-order, pre-order, post-order).
3. **Advanced Operations**:
   * **Balancing**: Techniques like AVL or Red-Black Trees to keep the tree balanced, ensuring optimal performance.
   * **Finding Minimum and Maximum**: Locating the smallest and largest elements in the BST.
   * **Successor and Predecessor**: Identifying the next and previous elements relative to a given node.
4. **Performance Characteristics**:
   * **Time Complexity**:
     + Insertion, Deletion, Search: Average-case O(log n), Worst-case O(n).
     + Traversal: O(n).
   * **Space Complexity**: O(n) for storing n nodes.

**Implementation**

The implementation can be done in various programming languages. Here are some screenshots of binary search tree code in cpp language.

**Applications**

1. **Databases**: Efficient indexing and query operations.
2. **File Systems**: Hierarchical data storage.
3. **Autocompletion**: Real-time suggestions in search engines.
4. **Network Routing**: Efficiently finding shortest paths.

**Conclusion**

Binary Search Trees are fundamental data structures that provide efficient means of managing sorted data. Understanding and implementing BSTs is crucial for developing various applications in computer science and software engineering. This project showcases the core functionalities and operations of BSTs, providing a solid foundation for further exploration and enhancement.