Data Structures and Algorithms using C++

Overview:

This project involves the implementation of a Binary Search Tree (BST) in C++. The primary focus is on key operations such as insertion, searching, and deletion. The BST maintains its properties by placing smaller values in the left subtree and larger values in the right subtree, ensuring efficient data management and retrieval.

Tools and Libraries:

- Standard C++ libraries are used for this implementation.

Features and Functionalities:

1. **Insert Elements:**
    - **Description:** Users can insert integers into the BST. Each new integer is placed in the correct position to maintain the BST properties. For example, smaller values go to the left subtree, while larger values go to the right subtree.
    - **Usage:** The user specifies the number of values to insert and inputs each value. The program confirms each insertion.
2. **Search Elements:**
    - **Description:** Users can search for specific integers within the BST. The program checks if the given integer exists in the tree and returns a message indicating whether the element is found or not.
    - **Usage:** The user specifies the number of values to search for and inputs each value. The program provides feedback for each search query.
3. **Delete Elements:**
    - **Description:** Users can delete elements from the BST. When an element is deleted, the BST restructures itself to maintain its properties. This involves rearranging the nodes to ensure the tree remains balanced and properly ordered.
    - **Usage:** The user specifies the number of values to delete and inputs each value. The program provides feedback on the success or failure of each deletion attempt.
4. **Display Elements:**
    - **Description:** The program supports displaying the elements of the BST using different traversal methods. These methods help users understand the structure and order of elements within the tree.
        - **In-order Traversal:** Displays the elements in sorted order. The traversal visits the left subtree, the root node, and then the right subtree.
        - **Pre-order Traversal:** Displays the elements in a sequence starting from the root node, then the left subtree, and finally the right subtree.
        - **Post-order Traversal:** Displays the elements starting with the left subtree, then the right subtree, and finally the root node.
    - **Usage:** The user selects a traversal method from the menu to view the elements in the desired order.
5. **User-Friendly Menu:**

- **Description:** A simple text-based menu allows users to interact with the BST. The menu presents options to insert, search, delete, and display elements, as well as exit the program.
- **Usage:** Users navigate the menu to perform various operations on the BST. The program continues to run until the user chooses to exit.

Example Interaction:

1. **Insert Elements:**
   - **User Action:** Inserts values: 25, 15, 35.
   - **Program Output:**
     - "Inserted 25"
     - "Inserted 15"
     - "Inserted 35"
2. **Search for Elements:**
   - **User Action:** Searches for values: 15, 40.
   - **Program Output:**
     - "Found" for 15
     - "Not found" for 40
3. **Delete Elements:**
   - **User Action:** Deletes value: 15.
   - **Program Output:** "Deleted node with key 15"
4. **Display Elements:**
   - **In-order:**
     - **User Action:** Displays elements using in-order traversal.
     - **Program Output:** "In-order: 25 35"
   - **Pre-order:**
     - **User Action:** Displays elements using pre-order traversal.
     - **Program Output:** "Pre-order: 25 15 35"
   - **Post-order:**
     - **User Action:** Displays elements using post-order traversal.
     - **Program Output:** "Post-order: 15 35 25"
5. **Exit:**
   - **User Action:** Chooses to exit the program.
   - **Program Behavior:** The program terminates.

Conclusion:

This project demonstrates the implementation of a Binary Search Tree in C++ with fundamental operations that are crucial for understanding data structures. The user-friendly interface allows for straightforward interaction with the BST. This implementation serves as a foundational exercise for those learning about data structures and algorithms, providing practical experience with a key concept in computer science.