Republic of the Philippines

Bicol University

Bicol University Polangui

Computer Studies Department

Polangui Albay

FINAL PROJECT IN DATA STRUCTURE AND ALGORITHM

Theme: Mystical Number Tree

- ## Project Overview

The program simulates a magical tree where each node represent a mystical number. Users can, (1) Insert numbers into the tree (creating magical branches), (2) search for a specific number (checking its magical power), (3) Delete a number (removing its magical from the tree), (4) Traverse the tree using Preorder, Inorder, and Postorder (exploring the magical pathways).The interactive system guides the user with clear prompts, creating a fun easy-to-follow experience.

- ## How to Run the Code

This code implements a **binary search tree (BST)** with user interaction. The program starts by defining a Node structure, where each node represents an element in the tree. It stores an integer value (data) and pointers to its left and right children (left and right). The MagicTree class manages the tree, with the root of the tree represented by a private pointer (root).

When the program runs, the user is greeted with a menu to interact with the tree:

1. **Adding a Number**: The insert method is called, which uses recursion to place the number in the correct position within the tree according to the BST rules (smaller numbers go to the left, larger numbers go to the right).
2. **Searching for a Number**: The search method traverses the tree to check if a specific number exists. If the number matches a node's value, it is found; otherwise, the search continues in the left or right subtree, depending on the value.
3. **Removing a Number**: The deleteNode method removes a number by locating the node to delete and handling three cases: no children (simple deletion), one child (replacing the node with its child), or two children (replacing the node with its smallest successor in the right subtree).
4. **Displaying the Tree**: The displayTraversals method shows the tree structure in three formats:
    1. **Preorder**: Root → Left → Right
    2. **Inorder**: Left → Root → Right (gives sorted order for BST)
    3. **Postorder**: Left → Right → Root
5. **Exiting the Program**: Terminates the program gracefully.

The program is structured to repeatedly display the menu, allowing users to perform multiple operations until they choose to exit. Each functionality is executed interactively with prompts and informative messages to guide the user. For example, when a number is added or removed, the program prints a confirmation message. Traversals provide a visual representation of the tree's structure at any point during execution.

Republic of the Philippines

Bicol University

Bicol University Polangui

Computer Studies Department

Polangui Albay

- # How the Code Works

1. **Node Structure**

   Purpose: Represents each node of the binary search tree (BST). Each node contains:

   data: Stores the value of the node.

   left: Points to the left child of the node.

   right: Points to the right child of the node.

   Example:

   If you add the value 10, a node is created with data = 10, left = nullptr, and right = nullptr.

2. **MagicTree Class**

   Purpose: Implements the operations of the BST: insertion, searching, deletion, and traversals.

   Private Members:

   Node* root: Pointer to the root of the tree.

   Recursive helper functions for insertion, searching, deletion, and traversals.

   Public Members:

   Provides user-friendly functions (insert, search, deleteNode, displayTraversals) that interact with the private recursive functions.

Key Functionalities

1. Insertion

- **How It Works**:
  - If the tree is empty (root == nullptr), the inserted value becomes the root.
  - If the value is smaller than the current node's value, it's inserted into the left subtree.
  - If the value is greater, it's inserted into the right subtree.
  - **Example**:

Add 10 -> [10]
Add 5 -> [10] with 5 as left child.
Add 15 -> [10] with 15 as right child.

2. Search

- **How It Works**:
  - Starts at the root and compares the value with the current node.
  - If the value matches the current node, it's found.
  - If smaller, the search continues in the left subtree; if larger, in the right subtree.
  - If the node is nullptr, the value doesn't exist in the tree.
  - **Example**:

Tree: [10, 5, 15]
Search for 5 -> Found!
Search for 12 -> Not found.

3. Deletion

- **How It Works**:
  - Finds the node to delete.
  - Three cases:

    1. **No children**: Simply deletes the node.
    2. **One child**: Replaces the node with its child.
    3. **Two children**: Replaces the node with the smallest value in its right subtree (inorder successor) and deletes the successor.

  - **Example**:

Tree: [10, 5, 15]
Remove 10 -> Replace with 15 (if 15 is the next smallest).

4. Traversals

- **Preorder**:
  - Visit the root, traverse the left subtree, then the right subtree.
  - Output: Root → Left → Right.
- **Inorder**:
  - Traverse the left subtree, visit the root, then the right subtree.
  - Output: Left → Root → Right (gives values in sorted order for BST).
- **Postorder**:
  - Traverse the left subtree, then the right subtree, and finally visit the root.
  - Output: Left → Right → Root.

- **Example**:

  Tree: [10, 5, 15]
  Preorder: 10 5 15
  Inorder: 5 10 15
  Postorder: 5 15 10

How to Use the Program

1. **Start the Program:**
   - Compile and run the code.
   - You'll see the main menu with options to add, search, delete, and explore the tree.
2. **Adding Numbers**:
   - Choose option 1 and enter a number to add it to the tree.
3. **Searching Numbers**:

   Choose option 2 and enter a number to check if it exists in the tree.
4. **Removing Numbers**:
   - Choose option 3 and enter a number to remove it from the tree.
5. **Exploring the Tree**:
   - Choose option 4 to display the tree in Preorder, Inorder, and Postorder traversal orders.
6. **Exit**:
   - Choose option 5 to quit the program.

Republic of the Philippines

Bicol University

Bicol University Polangui

Computer Studies Department

Polangui Albay

Run Example

Input:

1 (Add 10)
1 (Add 5)
1 (Add 15)
2 (Search 5)
3 (Remove 10)
4 (Display Traversals)
5 (Exit)

Output:

Adding 10 to the Mystical Tree...
Adding 5 to the Mystical Tree...
Adding 15 to the Mystical Tree...
Searching for 5 in the Mystical Tree...
The magical number 5 is found in the tree!
Removing 10 from the Mystical Tree...
Exploring the Mystical Tree paths...
Preorder (Magical Seed Path): 15 5
Inorder (Magical Growth Order): 5 15
Postorder (Magical Destruction Order): 5 15
Exiting the Mystical Realm. Farewell!

REFERENCES:

W3Schools.com. (n.d.). https://www.w3schools.com/cpp/default.asp

MEMBERS:

CANON, ROSE ANGELA

RESENTES, SHANLEY R.

SABLAYAN, PENELOPE R.