|  |  |  |
| --- | --- | --- |
| Image result for latest marwadi university logo | **Marwadi University**  **Faculty of Technology**  **Department of Information and Communication Technology** | |
| **Subject: DSC  (01CT0308)** | Aim: Implementations of Binary Tree menu-driven program to traversal, insert, delete, and search. | |
| **Experiment No: 6** | **Date: 21- 10 - 2023** | **Enrolment No:-** 92200133030 |

**Experiment – 6**

**Objective:** Implementations of Binary Tree menu-driven program to traversal, insert, delete, and search.

**Code :-**

#include<iostream>

using namespace std ;

class Node {

public :

int data ;

Node\* Left;

Node\* Right;

Node(int val) {

data = val ;

Left = NULL ;

Right = NULL ;

}

};

class Tree {

public :

Node\* Root ;

Tree() {

Root = NULL ;

}

void InsertNode(Node\* &Root , int val) {

Node\* newNode = new Node(val);

if(Root == NULL) {

Root = newNode ;

return ;

}

if(val > Root->data) {

InsertNode(Root->Right,val);

return;

}

else {

InsertNode(Root->Left , val);

return;

}

}

void Display(Node\* Root) {

if (Root == NULL) {

return;

}

Display(Root->Left);

cout << Root->data << " ";

Display(Root->Right);

}

bool Search(Node\* &Root , int val) {

if(Root == NULL) {

return false;

}

if(val == Root->data) {

return true;

}

if(val > Root->data) {

return Search(Root->Right , val);

}

else {

return Search(Root->Left , val);

}

}

void Delete(Node\* &Root , int val) {

if(val > Root->data) {

Delete(Root->Right,val);

return;

}

else if(val < Root->data) {

Delete(Root->Left , val);

return;

}

else {

if(Root->Left == NULL && Root->Right == NULL) {

delete Root;

return;

}

else if(Root->Left == NULL) {

Node\* todelete = Root ;

Root = Root->Right ;

delete todelete;

return;

}

else if (Root->Right == NULL) {

Node\* toDelete = Root;

Root = Root->Left ;

delete toDelete;

return;

}

else {

Node\* temp = Root;

Root = Root->Right;

Root->Right = temp ;

delete temp;

return;

}

}

}

void PreOrder(Node\* &Root) {

if(Root == NULL) {

return;

}

cout << Root->data << " " ;

PreOrder(Root->Left);

PreOrder(Root->Right);

}

void InOrder(Node\* &Root) {

if(Root == NULL) {

return ;

}

InOrder(Root->Left);

cout << Root->data << " " ;

InOrder(Root->Right);

}

void PostOrder(Node\* &Root) {

if(Root == NULL) {

return ;

}

PostOrder(Root->Left);

PostOrder(Root->Right);

cout << Root->data << " " ;

}

};

int main() {

Tree T;

int choice, value;

do {

cout << "Binary Search Tree Menu:" << endl;

cout << "1. Insert a Node" << endl;

cout << "2. Search for a Node" << endl;

cout << "3. Delete a Node" << endl;

cout << "4. Pre-Order Traversal " << endl;

cout << "5. In-Order Traversal" << endl ;

cout << "6. Post-Order Traversal" << endl ;

cout << "7. Exiting The Proramm" << endl ;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter the value to insert: ";

cin >> value;

T.InsertNode(T.Root,value);

break;

case 2:

cout << "Enter the value to search for: ";

cin >> value;

if (T.Search(T.Root,value)) {

cout << "Value found in the tree." << endl;

} else {

cout << "Value not found in the tree." << endl;

}

break;

case 3:

cout << "Enter the value to delete: ";

cin >> value;

T.Delete(T.Root,value);

break;

case 4:

cout << "Pre - Order Traversal : ";

T.PreOrder(T.Root);

break;

case 5:

cout << "In - Order Traversal : ";

T.InOrder(T.Root);

break;

case 6:

cout << "Post - Order Traversal : " ;

T.PostOrder(T.Root);

break;

case 7:

cout << "Exiting the program." << endl;

break;

default:

cout << "Invalid choice. Please try again." << endl;

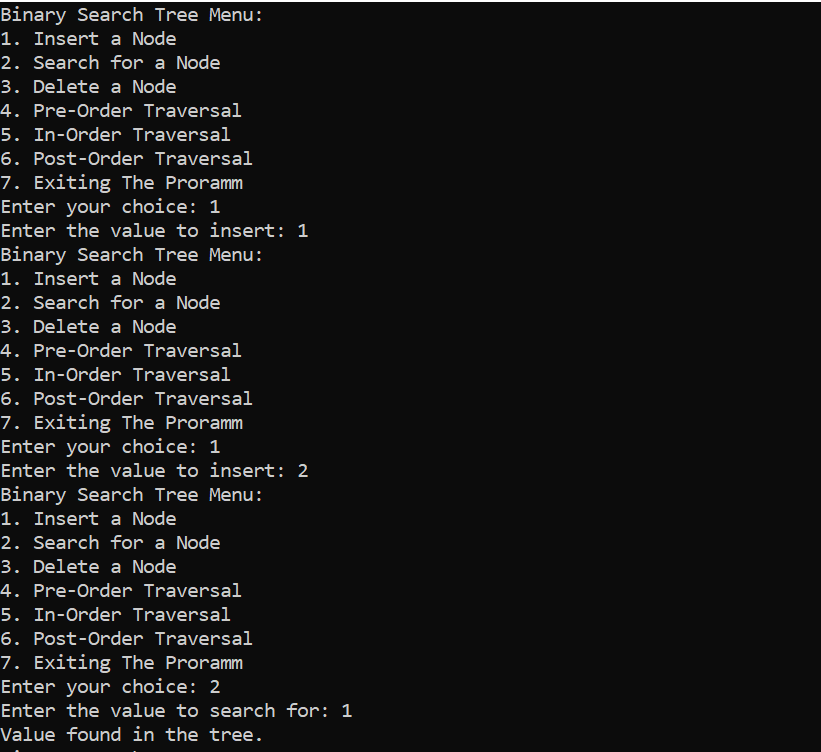
}

} while (choice != 7);

return 0;

}

**Output:**

****