**CSC221: DATA STRUCTURES & ALGORITHMS**

**BSCS 3*B***

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | LAB | | **10** | | Implementation of the BINARY SEARCH TREE with the help of algorithms for following functions   * SearchBST() * DeleteBST() |
|  |  |
| Objective(s): | Upon completion of this lab session, Student will be able to understand the following concepts |
| |  |  | | --- | --- | | 1 | Binary Search Tree definition | | 2 | Creation of Binary Search Tree | | 3 | Operation of Binary Search Tree | |  |  | |  |  | | |



**Submitted By:**

* Ahsan Ghaffar
* Reg: 48411
* BS(CS)\_3B

**Submitted to:**

* Miss Ambreen Akram(AA)

**Submission Date:**

[06/05/2018]

**DEPARTMENT OF COMPUTER SCIENCE**

**BAHRIA UNIVERSITY, KARACHI CAMPUS**

|  |
| --- |
| Lab Task(s): |
|
|  |
| |  |  | | --- | --- | | 1 | Write a program to create a BinarySearchTree . | | 2 | Add a function to the BinarySearchTree that find a Node in a Binary Search Tree | | 3 | Add a function to the BinarySearchTree that find Minimum and Maximum Value in a Binary Search Tree | | 4 | Add a function to the BinarySearchTree that delete Node from Binary Search Tree | | |

**SOURCECODE:**

#include<iostream>

#include<string>

using namespace std;

struct treenode {

int id;

treenode \*left;

treenode \*right;

}; treenode \*root = NULL;

bool add\_record(int id);

bool delete\_record(int id, treenode \*&);

void update\_record(int id);

bool search\_record(int id);

bool display\_record(treenode \*nodeptr);

void max(treenode \*nodeptr);

void min(treenode \*nodeptr);

bool add\_record(int id)

{

treenode \*new1 = new treenode;

treenode \*nodeptr;

//bool flag = true;

new1->id = id;

new1->left = NULL;

new1->right = NULL;

if (root == NULL) {

root = new1;

return true;

}

else

{

nodeptr = root;

while (nodeptr != NULL)

{

if (id < nodeptr->id)

{

if (nodeptr->left != NULL)

nodeptr = nodeptr->left;

else {

nodeptr->left = new1;

return true;

break;

}

}//End of IF:

else if (id > nodeptr->id)

{

if (nodeptr->right != NULL)

nodeptr = nodeptr->right;

else {

nodeptr->right = new1;

return true;

break;

}

}//End of Else if

else if (id == nodeptr->id) {

return false;

break;

}//End of Else if

}//End of While:

}//End Of Outer Most Else:

}//End of add\_record Function:

//Delete record

bool delete\_record(int id, treenode \*&nodeptr)

{

//Search id from record for deletion of record:

if (search\_record(id) == true)

{

//Find the location of record for deletion through recursion:

if (id < nodeptr->id)

delete\_record(id, nodeptr->left);

else if (id > nodeptr->id)

delete\_record(id, nodeptr->right);

//After find location use if else conditions for deletin of record from tree:

else

{

treenode \*temp;

treenode \*temp1;

//Delte record which have only left child:

if (nodeptr->right == NULL)

{

temp = nodeptr;

nodeptr = nodeptr->left;

delete(temp);

}

//delete record which have only right child:

else if (nodeptr->left == NULL)

{

temp = nodeptr;

nodeptr = nodeptr->right;

delete(temp);

}

//delete record which have both childs:

else

{

temp = nodeptr->right;

while (temp->left != NULL)

temp = temp->left;

temp->left = nodeptr->left;

temp1 = nodeptr;

nodeptr = nodeptr->right;

delete(temp1);

}

}//End of outer most else:

return true; //if record deleted return true

}//End of outer most if:

else

return false;

}//End of delete\_record function

//Search Record from Function:

bool search\_record(int id)

{

treenode \*temp = root;

bool flag = false;

while (temp != NULL)

{

if (temp->id == id) {

flag = true;

break;

}

else if (id < temp->id)

temp = temp->left;

else if (id > temp->id)

temp = temp->right;

}//End of While:

if (flag == true)

return flag;

else

return flag;

}// End of search\_record Function

//Update Record

void update\_record(int id)

{

treenode \*temp = root;

bool flag = false;

long long int up\_id;

while (temp != NULL)

{

if (temp->id == id)

{

cout << "Enter Contact number to override this id: ";

cin >> up\_id;

if (search\_record(up\_id) == true)

cout << "\nThis contact number is already in the Record!!!" << endl;

else

{

delete\_record(id, root);

add\_record(up\_id);

flag = true;

break;

}

}//End of Outer most IF:

else if (id < temp->id)

temp = temp->left;

else if (id > temp->id)

temp = temp->right;

}//End of Outer most while:

if (flag == true) {

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " RECORD UPDATED SUCCESSFULLY" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" << endl;

}

else

cout << "\nRecord not found which you want to update!!!" << endl;

}//End of update\_record Function

//Display All records :

bool display\_record(treenode \*nodeptr)

{

bool flag = false;

if (nodeptr != NULL)

{

display\_record(nodeptr->left);

cout << "\*\*\*\*\*\*" << endl;

cout << nodeptr->id << endl;

cout << "\*\*\*\*\*\*" << endl;

display\_record(nodeptr->right);

return true;

}

if (flag == false)

return false;

}//End of display\_record Function:

void min(treenode \*nodeptr)

{

if (nodeptr->left != NULL)

min(nodeptr->left);

else if (nodeptr->left == NULL) {

cout << "\nMinimum: " << nodeptr->id << endl;

}

}

void max(treenode \*nodeptr)

{

if (nodeptr->right != NULL)

max(nodeptr->right);

else if (nodeptr->right == NULL) {

cout << "Maximum: " << nodeptr->id << endl;

}

}

int main()

{

int id;

char c;

int choice;

do {

cout << "\n============================" << endl;

cout << " BINARY SEARCH TREE " << endl;

cout << "============================" << endl;

cout << "1- ADD NEW ID " << endl;

cout << "2- DELETE ID " << endl;

cout << "3- UPDATE ID " << endl;

cout << "4- SEARCH ID " << endl;

cout << "5- DISPLAY ID " << endl;

cout << "6- MAXIMUM & MINIMUM " << endl;

cout << "7- Exit " << endl;

cout << "choose any one from the above options: ";

cin >> choice;

if (choice == 1)

{

cout << "\n===================" << endl;

cout << " ADD NEW ID" << endl;

cout << "===================\n" << endl;

cout << "Enter ID for insertion: ";

cin >> id;

if (add\_record(id) == true) {

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "ID ADDED SUUCCESSFULLY" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" << endl;

}

else

cout << "\nDuplicate ID don't allowed!!! " << endl;

}

else if (choice == 2)

{

cout << "\n===================" << endl;

cout << " DELETE ID" << endl;

cout << "===================\n" << endl;

cout << "\nEnter ID num to deletion of Record: ";

cin >> id;

if (delete\_record(id, root) == true) {

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "ID DELETED SUCCESSFULLY" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" << endl;

}

else

cout << "\nID Not Found Which you wants to Delete!!!" << endl;

}

else if (choice == 3) {

cout << "\n===================" << endl;

cout << " UPDATE ID" << endl;

cout << "===================\n" << endl;

cout << "Enter ID number which you wants to update: ";

cin >> id;

update\_record(id);

}

else if (choice == 4)

{

cout << "\n===================" << endl;

cout << " SEARCH ID" << endl;

cout << "===================\n" << endl;

cout << "Enter ID number to Search a Record: ";

cin >> id;

if (search\_record(id) == true)

cout << "\nID FOUND:" << endl;

else

cout << "\nID NOT FOUND:" << endl;

}

else if (choice == 5)

{

cout << "\n===================" << endl;

cout << " DISPLAY ALL" << endl;

cout << "===================\n" << endl;

if (display\_record(root) == false)

cout << "RECORD EMPTY" << endl;

}

else if (choice == 6) {

cout << "\n=====================" << endl;

cout << " MAXIMUM & MINIMUM" << endl;

cout << "=====================\n" << endl;

max(root);

min(root);

}

else if (choice == 7)

{

system("pause");

exit(0);

}

cout << "\nPress 'y' to continue and 'n' to exit: ";

cin >> c;

} while (c == 'Y' || c == 'y');

system("pause");

return 0;

}

**SCREENSHOTS:**







