Experiment No. 4

Aim: Implementation of Binary Tree and its traversal for real-world application.

Objectives:

1. To learn fundamentals and implementation of binary tree.
2. To develop an ability to design and analyze algorithms using tree data structures.

Theory:

- A binary tree is a data structure that is defined as a collection of elements called nodes.
- In a binary tree, the topmost element is called the root node, and each node has 0, 1 or at ~~least~~ the most 2 children.
- A node that has zero children is called a leaf node or a terminal node
- Every node contains a data element, a left pointer which points to the left child, and a right pointer which points to the right child.
- The root element is pointed by a 'root' pointer.

Terminologies:-

- **Parent -** If N is any node in T that has left successor s_1 and right successor s_2, then N is called the parent of s_1 and s_2.

Level number - Every node in the binary tree is assigned to a level number.

Degree of a node - It is equal to the number of children that a node has.

Sibling - All nodes that are at the same level and share the same parent are called siblings.

Leaf node - A node that has no children.

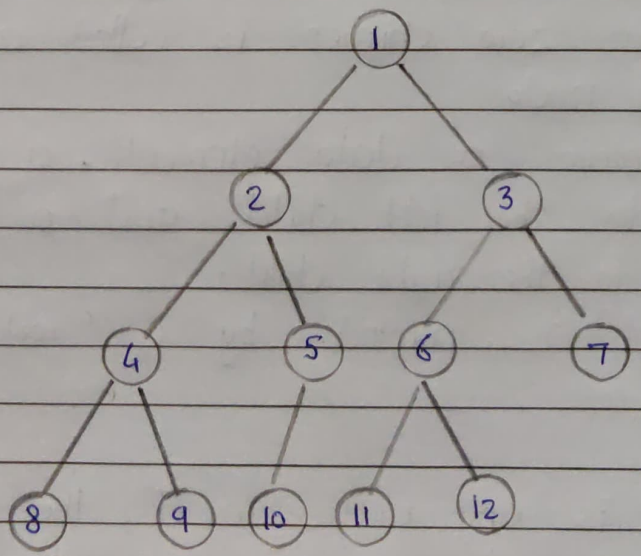
Similar binary trees - Two binary trees are said to be similar if both these trees have the same structure.

Edge - It is the line connecting a node N to any of its successors

Path - A sequence of consecutive edges.

Depth - The depth of a node is given as the length of the path from the root to the node.

Height of a tree - It is the total number of nodes on the path from the root node to the deepest node in the tree.



Operations -

1. Searching - Find the location of some specific element in a binary tree.

2. Insertion - Adding a new element to the tree at the appropriate location.

3. Deletion - Deleting some specific node from a binary tree.

4. Traversing - Process of visiting each node exactly one.

Tree traversal and its types -

- Traversing a binary tree is the process of visiting each node in the tree exactly once in a systematic way.
- Unlike linear data structure in which the elements are traversed sequentially, tree is a non-linear data structure in which the elements can traversed in many different ways.

(1) Pre-order traversal: To traverse a non-empty binary tree in pre-order, the following operations are performed recursively at each node.
The algorithm works by -
① Visiting the root node
② Traversing the left sub-tree. and
③ Traversing the right subtree.

(2) In-order traversal: To traverse a non-empty binary tree in in-order, the following operations are performed recursively at each node.
The algorithm works by -
① Traversing the left sub-tree
② Visiting the root node and finally
③ Traversing the right sub-tree

(3) Post-order traversal: To traverse a non-empty binary tree in post-order, the following operations are performed recursively at each node.
The algorithm works by -
① Traversing the left sub-tree
② Traversing the right sub-tree
③ Visiting the root node.

* Algorithms:

→ Searching for a given value.

```
Step 1 - If TREE → DATA = VAL OR TREE = NULL
            RETURN TREE
         ELSE
         IF VAL < TREE → DATA
         RETURN search element (TREE → LEFT VAL)
         ELSE
           RETURN searchElement (TREE → RIGHT, VALUE)
         [END OF IF]
         [END OF IF]
Step 2 - END
```

Insertion: INSERT (TREE, VAL)

Step 1 - IF TREE = NULL
allocate memory for tree
set TREE → DATA = VAL
set TREE → LEFT = TREE → RIGHT = NULL
ELSE
IF VAL < TREE → DATA
Insert (TREE → LEFT, VAL)
ELSE
Insert (TREE → RIGHT, VAL)
[END OF IF]
[END OF IF]

Step 2 - END

Deletion: Delete (TREE, VAL)

Step 1 - IF TREE = NULL
Write "VAL NOT FOUND IN THE TREE"
ELSE IF VAL < TREE → DATA
Delete (Tree → LEFT, VAL)
ELSE IF VAL > TREE → DATA
Delete (TREE → RIGHT, VAL)
Else if tree → LEFT AND TREE → RIGHT
Set temp = findLargestNode (TREE → LEFT)
Set TREE → DATA = TEMP → DATA
Delete (TREE → LEFT, TEMP → DATA)
ELSE
Set temp = TREE
IF TREE → LEFT = NULL and TREE → RIGHT = NULL
Set TREE = NULL

```
    ELSE IF TREE -> LEFT != NULL
        Set TREE = TREE -> LEFT
    ELSE
        Set TREE = TREE -> RIGHT
    [END OF IF]
    FREE TEMP
    [END OF IF]
Step 2: END
```

Pre-order Traversal -

```
Step 1 - Repeat steps 2 to 4 while TREE != NULL
Step 2 - Write TREE -> DATA
Step 3 - PREORDER (TREE -> LEFT)
Step 4 - PREORDER (TREE -> RIGHT)
        [END OF LOOP]
Step 5 - END.
```

In-order Traversal

```
Step 1 - Repeat steps 2 to 4 while TREE != NULL
Step 2 - INORDER (TREE -> LEFT)
Step 3 - Write TREE -> DATA
Step 4 - INORDER (TREE -> RIGHT)
    [END OF LOOP]
Step 5 - END
```

Post-order Traversal-

Step 1 - Repeat steps 2 to 4 while TREE != NULL
Step 2 - Post order (TREE -> LEFT)
Step 3 - Post order (TREE -> RIGHT)
Step 4 - Write TREE -> DATA
[END OF LOOP]
Step 5 - END

Example -

Trees are used in decision-based algorithm, ~~Bs~~ ~~to~~ that is in machine learning which works upon the algorithm of tree.

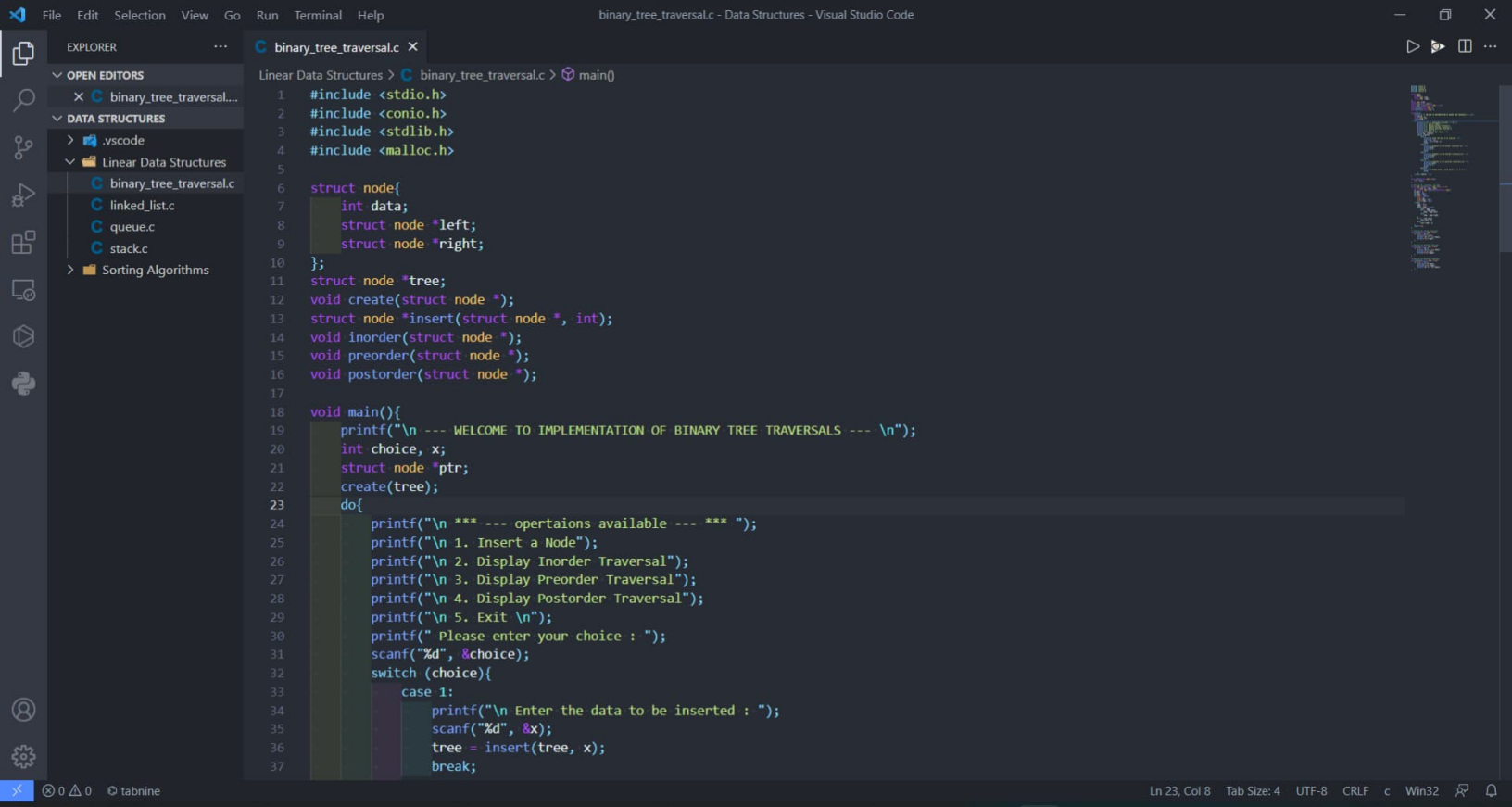
Trees are also used in databases for indexing.

File system in any operating system also makes use of trees.

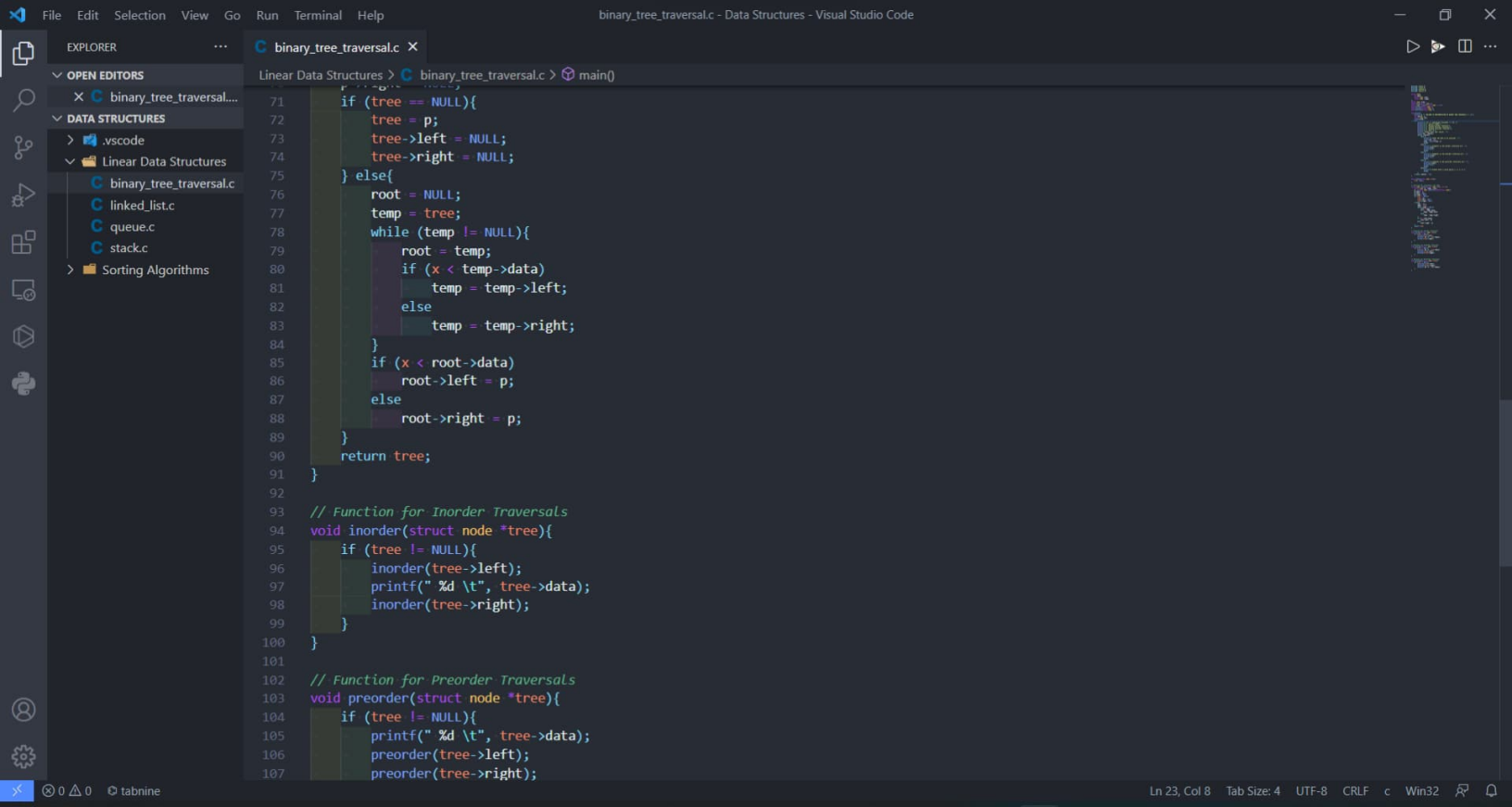
~~Websites which allows~~

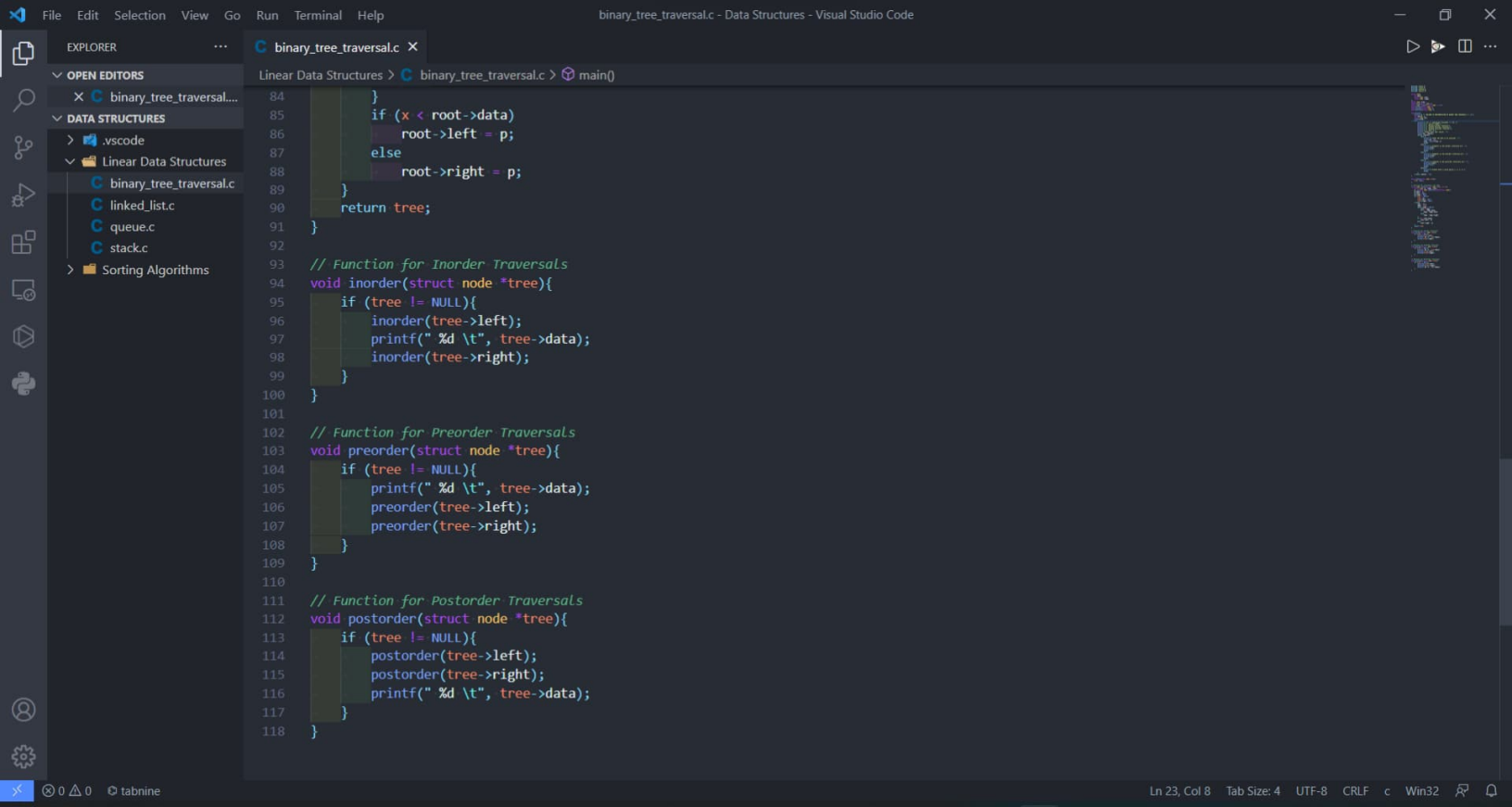
Conclusion: In this experiment, we learned about ~~queues~~ trees. We implemented trees and performed its basic operations like insertion, deletion and traversing through the nodes in the tree. We also came across the examples where trees are implemented in real-world.

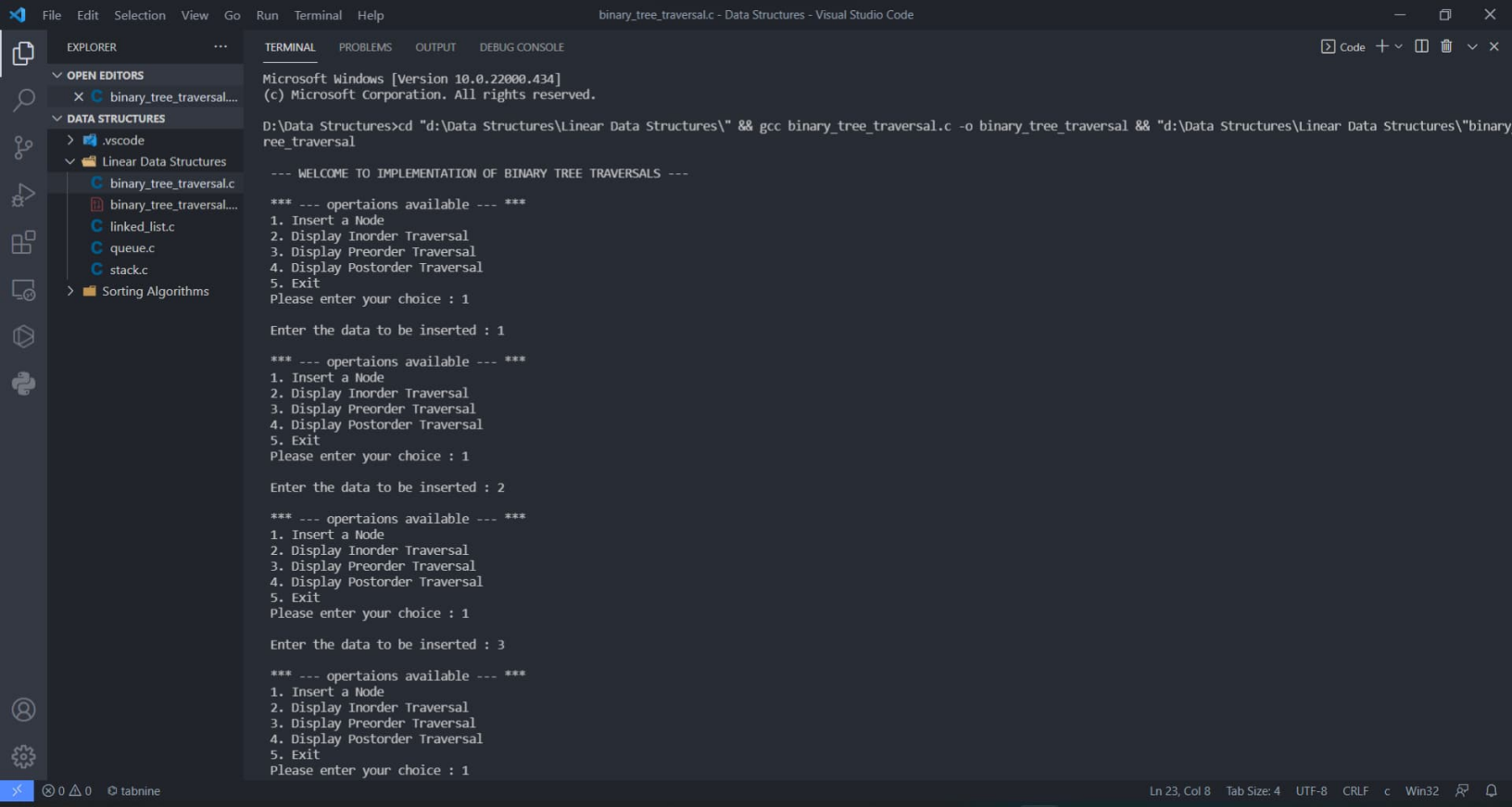
Outcome - Applied the concepts of trees for real-world application.

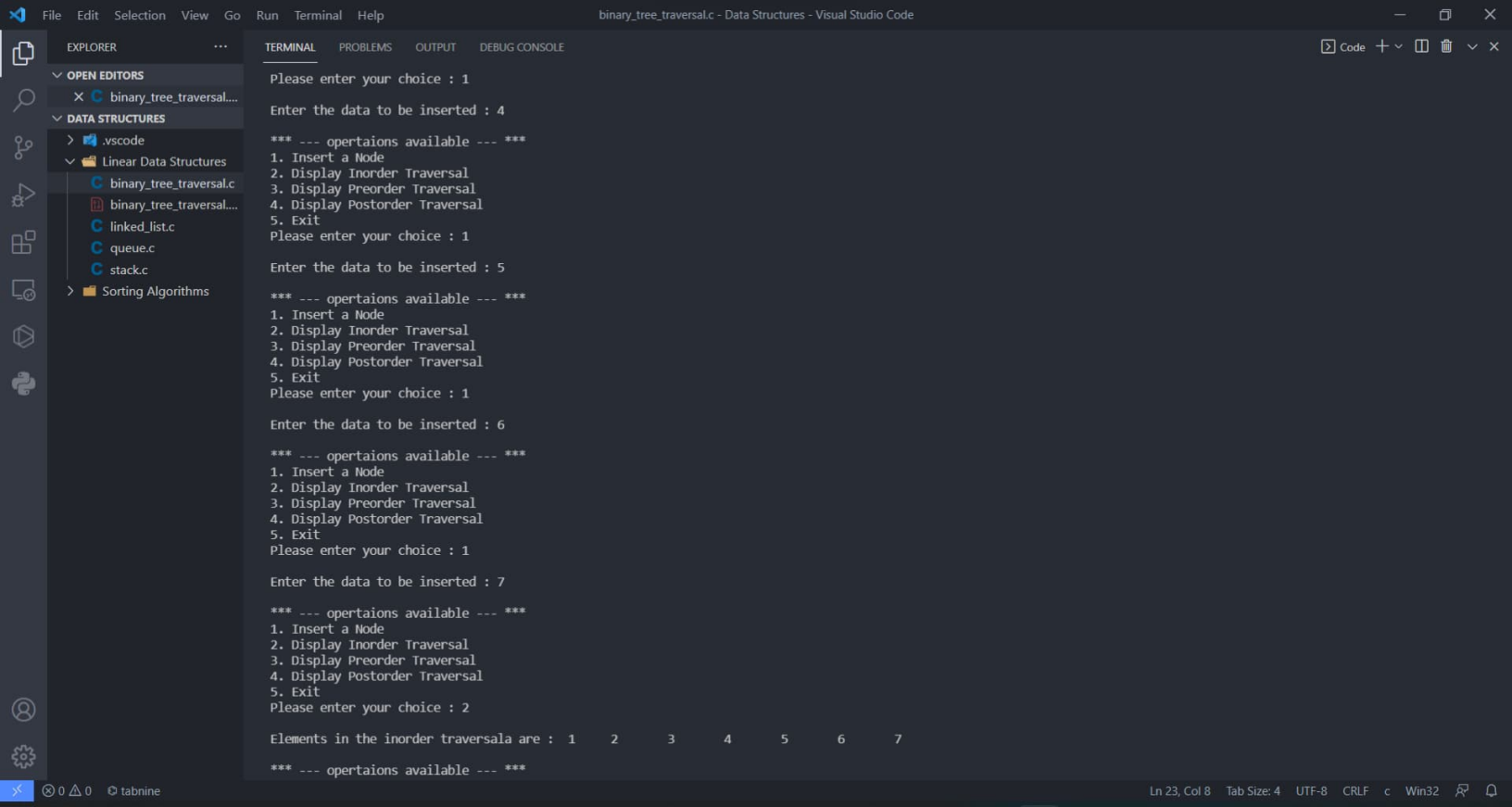

File Edit Selection View Go Run Terminal Help
binary_tree_traversal.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
binary_tree_traversal....
DATA STRUCTURES
.vscode
Linear Data Structures
binary_tree_traversal.c
linked_list.c
queue.c
stack.c
Sorting Algorithms
binary_tree_traversal.c
Linear Data Structures > binary_tree_traversal.c > main()
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>
struct node{
int data;
struct node *left;
struct node *right;
};
struct node *tree;
void create(struct node *);
struct node *insert(struct node *, int);
void inorder(struct node *);
void preorder(struct node *);
void postorder(struct node *);
void main(){
printf("\n --- WELCOME TO IMPLEMENTATION OF BINARY TREE TRAVERSALS --- \n");
int choice, x;
struct node *ptr;
create(tree);
do{
printf("\n *** --- opertaions available --- *** ");
printf("\n 1. Insert a Node");
printf("\n 2. Display Inorder Traversal");
printf("\n 3. Display Preorder Traversal");
printf("\n 4. Display Postorder Traversal");
printf("\n 5. Exit \n");
printf(" Please enter your choice : ");
scanf("%d", &choice);
switch (choice){
case 1:
printf("\n Enter the data to be inserted : ");
scanf("%d", &x);
tree = insert(tree, x);
break;
tabnine
Ln 23, Col 8
Tab Size: 4
UTF-8
CRLF
c
Win32

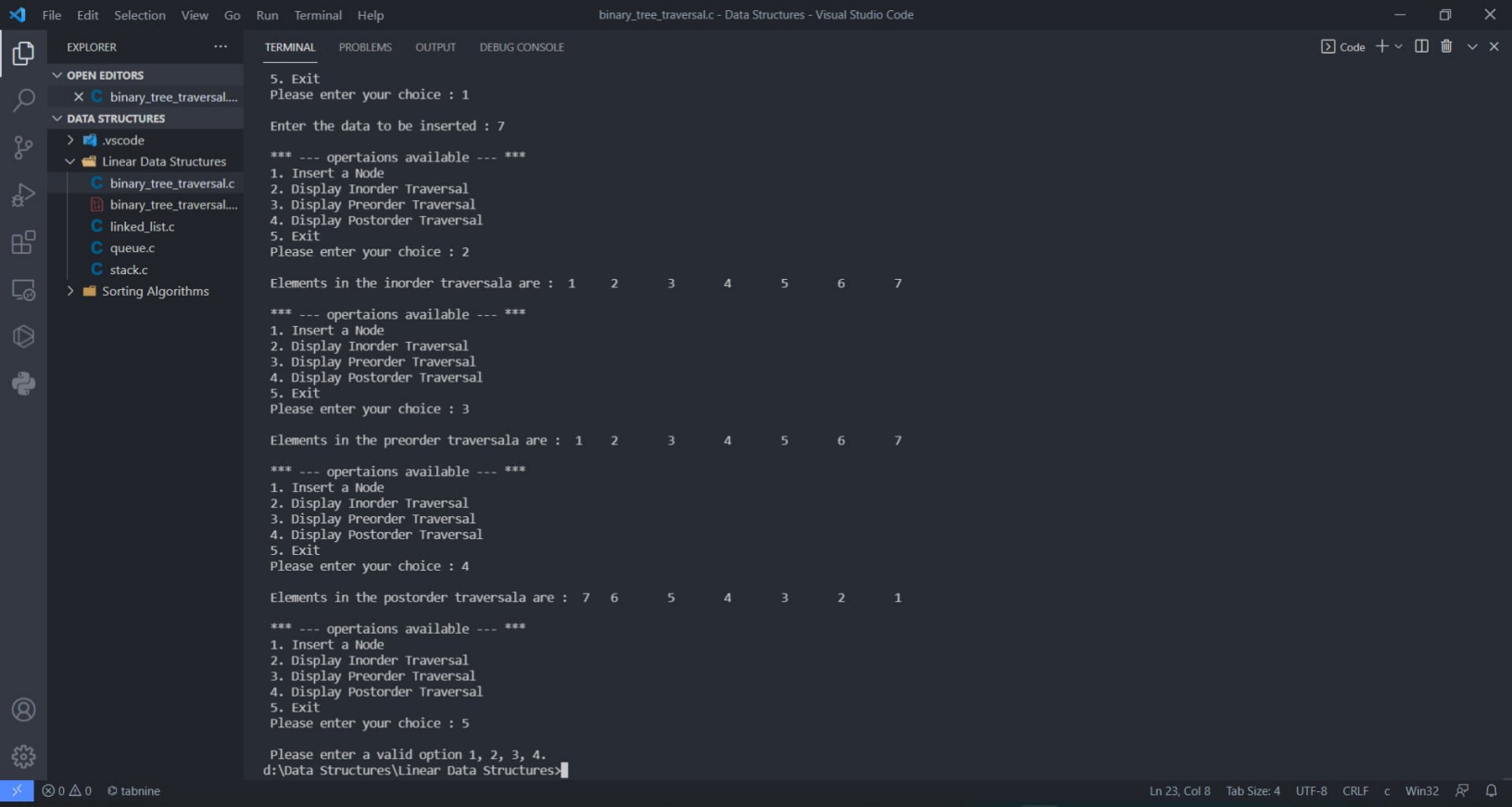
```
                printf("\n Enter the data to be inserted : ");
                scanf("%d", &x);
                tree = insert(tree, x);
                break;
            case 2:
                printf("\n Elements in the inorder traversala are : ");
                inorder(tree);
                printf("\n");
                break;
            case 3:
                printf("\n Elements in the preorder traversala are : ");
                preorder(tree);
                printf("\n");
                break;
            case 4:
                printf("\n Elements in the postorder traversala are : ");
                postorder(tree);
                printf("\n");
                break;
            default:
                printf("\n Please enter a valid option 1, 2, 3, 4.");
                break;
        }
    } while (choice != 5);
}

void create(struct node *tree){
    tree = NULL;
}

// Function for inserting a new node
struct node *insert(struct node *tree, int x){
    struct node *p, *temp, *root;
    p = (struct node *)malloc(sizeof(struct node));
    p->data = x;
    p->left = NULL;
    p->right = NULL;
```


File Edit Selection View Go Run Terminal Help
binary_tree_traversal.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
binary_tree_traversal....
DATA STRUCTURES
.vscode
Linear Data Structures
binary_tree_traversal.c
linked_list.c
queue.c
stack.c
Sorting Algorithms
binary_tree_traversal.c
Linear Data Structures > binary_tree_traversal.c > main()
if (tree == NULL){
tree = p;
tree->left = NULL;
tree->right = NULL;
} else{
root = NULL;
temp = tree;
while (temp != NULL){
root = temp;
if (x < temp->data)
temp = temp->left;
else
temp = temp->right;
}
if (x < root->data)
root->left = p;
else
root->right = p;
}
return tree;
}
// Function for Inorder Traversals
void inorder(struct node *tree){
if (tree != NULL){
inorder(tree->left);
printf(" %d \t", tree->data);
inorder(tree->right);
}
}
// Function for Preorder Traversals
void preorder(struct node *tree){
if (tree != NULL){
printf(" %d \t", tree->data);
preorder(tree->left);
preorder(tree->right);
0 0
tabnine
Ln 23, Col 8
Tab Size: 4
UTF-8
CRLF
c
Win32


File Edit Selection View Go Run Terminal Help
binary_tree_traversal.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
binary_tree_traversal....
DATA STRUCTURES
.vscode
Linear Data Structures
binary_tree_traversal.c
linked_list.c
queue.c
stack.c
Sorting Algorithms
binary_tree_traversal.c
Linear Data Structures > binary_tree_traversal.c > main()
84 }
85 if (x < root->data)
86 root->left = p;
87 else
88 root->right = p;
89 }
90 return tree;
91 }
92
93 // Function for Inorder Traversals
94 void inorder(struct node *tree){
95 if (tree != NULL){
96 inorder(tree->left);
97 printf(" %d \t", tree->data);
98 inorder(tree->right);
99 }
100 }
101
102 // Function for Preorder Traversals
103 void preorder(struct node *tree){
104 if (tree != NULL){
105 printf(" %d \t", tree->data);
106 preorder(tree->left);
107 preorder(tree->right);
108 }
109 }
110
111 // Function for Postorder Traversals
112 void postorder(struct node *tree){
113 if (tree != NULL){
114 postorder(tree->left);
115 postorder(tree->right);
116 printf(" %d \t", tree->data);
117 }
118 }
0 0
tabnine
Ln 23, Col 8
Tab Size: 4
UTF-8
CRLF
c
Win32


File Edit Selection View Go Run Terminal Help
binary_tree_traversal.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
binary_tree_traversal....
DATA STRUCTURES
.vscode
Linear Data Structures
binary_tree_traversal.c
binary_tree_traversal....
linked_list.c
queue.c
stack.c
Sorting Algorithms
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
Code
Microsoft Windows [Version 10.0.22000.434]
(c) Microsoft Corporation. All rights reserved.
D:\Data Structures>cd "d:\Data Structures\Linear Data Structures\" && gcc binary_tree_traversal.c -o binary_tree_traversal && "d:\Data Structures\Linear Data Structures\"binary
ree_traversal
--- WELCOME TO IMPLEMENTATION OF BINARY TREE TRAVERSALS ---
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 1
Enter the data to be inserted : 1
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 1
Enter the data to be inserted : 2
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 1
Enter the data to be inserted : 3
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 1
0 0 tabnine
Ln 23, Col 8 Tab Size: 4 UTF-8 CRLF c Win32


File Edit Selection View Go Run Terminal Help
binary_tree_traversal.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
binary_tree_traversal....
DATA STRUCTURES
.vscode
Linear Data Structures
binary_tree_traversal.c
binary_tree_traversal....
linked_list.c
queue.c
stack.c
Sorting Algorithms
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
Code
Please enter your choice : 1
Enter the data to be inserted : 4
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 1
Enter the data to be inserted : 5
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 1
Enter the data to be inserted : 6
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 1
Enter the data to be inserted : 7
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 2
Elements in the inorder traversala are : 1 2 3 4 5 6 7
*** --- opertaions available --- ***
tabnine
Ln 23, Col 8
Tab Size: 4
UTF-8
CRLF
c
Win32


File Edit Selection View Go Run Terminal Help
binary_tree_traversal.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
binary_tree_traversal....
DATA STRUCTURES
.vscode
Linear Data Structures
binary_tree_traversal.c
binary_tree_traversal....
linked_list.c
queue.c
stack.c
Sorting Algorithms
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
Code
5. Exit
Please enter your choice : 1
Enter the data to be inserted : 7
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 2
Elements in the inorder traversala are : 1 2 3 4 5 6 7
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 3
Elements in the preorder traversala are : 1 2 3 4 5 6 7
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 4
Elements in the postorder traversala are : 7 6 5 4 3 2 1
*** --- opertaions available --- ***
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice : 5
Please enter a valid option 1, 2, 3, 4.
d:\Data Structures\Linear Data Structures>
0 0 tabnine
Ln 23, Col 8 Tab Size: 4 UTF-8 CRLF c Win32