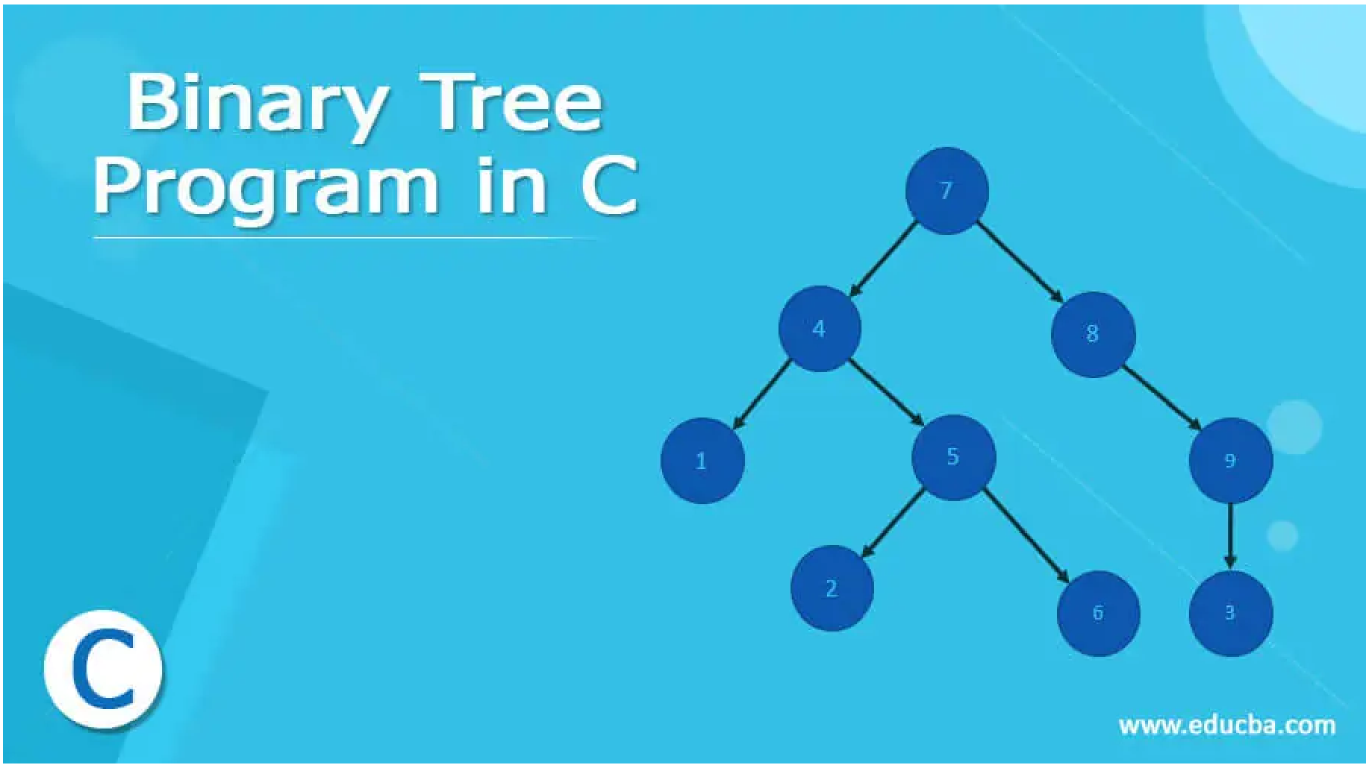
EDUCBA
(https://www.educba.com/software-development/)

← (https://www.educba.com/atm-program-in-c/)

→ (https://www.educba.com/string-concatenation-in-c/)



Introduction to Binary Tree Program in C

Binary tree program in C is a nonlinear data structure used for data search and organization. Binary tree is comprised of nodes, and these nodes each being a data component, have left and right child nodes. Unlike other data structures, such as, Arrays, Stack and queue, Linked

which are Linear type data structures whereas Trees are Hierarchical type of data struc

Binary search tree or BST in short, whose nodes each store keys greater than their left child nodes and less than all the right child nodes. As the data in a binary tree is organized, it allows

Binary tree does not have any particular Syntax but has an Algorithm to follow in implementing Binary Tree.

```
struct BT {
int data;
struct BT *rightNode, *leftNode;
};
```

- Left sub tree of node contains nodes with keys less than node’s key
- Right sub tree of node contains nodes with keys greater than node’s key
- Both left and right sub tree must also be a Binary tree, and no duplicates are allowed.

Illustration of Binary Tree:

Above properties of Binary tree provide ordering among keys such that operations like search, min, and max can be done faster. If in case there is no order, then user has to compare every key for searching a given key.

Algorithm for Binary Tree:

1. A new binary tree is created and values are assigned
2. Write a function insert() in such a way that node and key will be two parameters and check for below conditions,
a. If rootNode == NULL, then return new node to calling function.
b. If rootNode => data < keyValue, then call insert() with rootNode => rightNode and assign return value in rootNode => rightNode.
c. If rootNode => data > keyValue, then call insert() with rootNode => leftNode and assign return value in rootNode => leftNode
3. Then finally, we can return original rootNode pointer to calling function.

Example #1: C program for insertion in a Binary tree

```
#include<stdio.h>
#include<stdlib.h>
struct BTnode
{
int keyVal;
struct BTnode *leftNode;
struct BTnode *newNode = malloc(sizeof(struct BTnode));
newNode->keyVal = value;
newNode->leftNode = NULL;
newNode->rightNode = NULL;
return newNode;
}
struct BTnode *insert(struct BTnode *rootNode, int value)
{
if(rootNode == NULL)
return getNode(value);
if(rootNode->keyVal < value)
rootNode->rightNode = insert(rootNode->rightNode,value);
else if(rootNode->keyVal > value)
rootNode->leftNode = insert(rootNode->leftNode,value);
return rootNode;
}
void insertorder(struct BTnode *rootNode)
{
if(rootNode == NULL)
return;
insertorder(rootNode->leftNode);
printf("%d ",rootNode->keyVal);
insertorder(rootNode->rightNode);

}
int main()
{
```

```
rootNode = insert(rootNode,1);
rootNode = insert(rootNode,5);
rootNode = insert(rootNode,2);
rootNode = insert(rootNode,9);
rootNode = insert(rootNode,3);
insertorder(rootNode);
return 0;
}
```

EDUCBA (https://www.educba.com/software-development/)

So here we are creating a Binary tree and then inserting the node values.

Example #2: Binary tree, inorder, preorder, and postorder traversal in C programming.

```
#include <stdio.h>
#include <stdlib.h>
struct BTnode {
int data;
struct BTnode* leftNode;
struct BTnode* rightNode;
};
void inorder(struct BTnode* rootNode) {
if (rootNode == NULL) return;
inorder(rootNode->leftNode);
printf("%d ->", rootNode->data);
inorder(rootNode->rightNode);
}
void preorder(struct BTnode* rootNode) {
if (rootNode == NULL) return;
printf("%d ->", rootNode->data);
preorder(rootNode->leftNode);

preorder(rootNode->rightNode);
}
void postorder(struct BTnode* rootNode) {
}
struct BTnode* createNode(value) {
struct BTnode* newNode = malloc(sizeof(struct BTnode));
newNode->data = value;
newNode->leftNode = NULL;
newNode->rightNode = NULL;
return newNode;
}
struct BTnode* insertLeftNode(struct BTnode* rootNode, int value)
{
rootNode->leftNode = createNode(value);
return rootNode->leftNode;
}
struct BTnode* insertRightNode(struct BTnode* rootNode, int value)
{
rootNode->rightNode = createNode(value);
return rootNode->rightNode;
}
int main() {
struct BTnode* rootNode = createNode(7);
insertLeftNode(rootNode, 4);
insertRightNode(rootNode, 8);
insertLeftNode(rootNode->leftNode, 1);
insertRightNode(rootNode->rightNode, 5);
insertLeftNode(rootNode->leftNode, 6);
insertRightNode(rootNode->rightNode, 3);
```

```
printf("\nPostorder \n");
postorder(rootNode);
}
```

Output:

So here we have performed inorder, preorder, and postorder traversal for a Binary tree by inserting nodes.
While searching for a value in Binary Tree, node is traversed from left to right.

Types of Binary Tree

Full Binary Tree:

Special type of Binary Tree where every parent node or an internal node has either 2 or no child nodes.

Perfect Binary Tree:

A Binary tree in which each internal node has exactly two children and all leaf nodes at same level.

Complete Binary Tree:

It is same as Full Binary Tree, but all leaf nodes must be at left and every level must have both left and right child nodes. And the last leaf node should not have the right child

Skewed Binary Tree:

It is similar to a pathological tree in which the binary tree is either dominated by left or right nodes. And it has two types: Left Skewed Binary tree and Right Skewed Binary Tree.

Balanced Binary Tree:

Type of Binary Tree in which difference between the height of left and right subtree for each child node is 0 or 1

Conclusion

With this, we shall conclude our topic "Binary Tree program in C". We have seen what Binary tree is and its Algorithm. Seen few examples on How Binary tree is created, how insertion is done, and how we can search the Binary Tree nodes and display. There are also types of Binary Tree which we have explained above. There are other operations too, which can help us to understand the concept of Binary Tree completely.

Recommended Articles

This is a guide to Binary Tree Program in C. Here we discuss the Introduction, syntax, Types of Binary Tree with Examples with code implementation. You may also have a look at the following articles to learn more –

1. Splay Tree in Data Structure (https://www.educba.com/splay-tree-in-data-structure/)
2. Spanning Tree Algorithm (https://www.educba.com/spanning-tree-algorithm/)
3. Tree Traversal Techniques (https://www.educba.com/tree-traversal-techniques/)
4. Binary Tree Interview Questions (https://www.educba.com/binary-tree-interview-

About Us

Blog (https://www.educba.com/blog/?source=footer)

Who is EDUCBA? (https://www.educba.com/about-us/?source=footer)

Sign Up (https://www.educba.com/software-development/signup/?source=footer)

Corporate Training (https://www.educba.com/corporate/?source=footer)

Certificate from Top Institutions (https://www.educba.com/educbalive/?source=footer)

QUIZ