**1.Binary search tree**

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node \*right\_child;

struct node \*left\_child;

};

struct node\* new\_node(int x){

struct node \*temp;

temp = malloc(sizeof(struct node));

temp->data = x;

temp->left\_child = NULL;

temp->right\_child = NULL;

return temp;

}

struct node\* search(struct node \* root, int x){

if (root == NULL || root->data == x)

return root;

else if (x > root->data)

return search(root->right\_child, x);

else

return search(root->left\_child, x);

}

struct node\* insert(struct node \* root, int x){

if (root == NULL)

return new\_node(x);

else if (x > root->data)

root->right\_child = insert(root->right\_child, x);

else

root -> left\_child = insert(root->left\_child, x);

return root;

}

struct node\* find\_minimum(struct node \* root) {

if (root == NULL)

return NULL;

else if (root->left\_child != NULL)

return find\_minimum(root->left\_child);

return root;

}

void inorder(struct node \*root){

if (root != NULL)

{

inorder(root->left\_child);

printf(" %d ", root->data);

inorder(root->right\_child);

}

}

int main() {

struct node \*root;

root = new\_node(20);

insert(root, 5);

insert(root, 1);

insert(root, 15);

insert(root, 9);

insert(root, 7);

insert(root, 12);

insert(root, 30);

insert(root, 25);

insert(root, 40);

insert(root, 45);

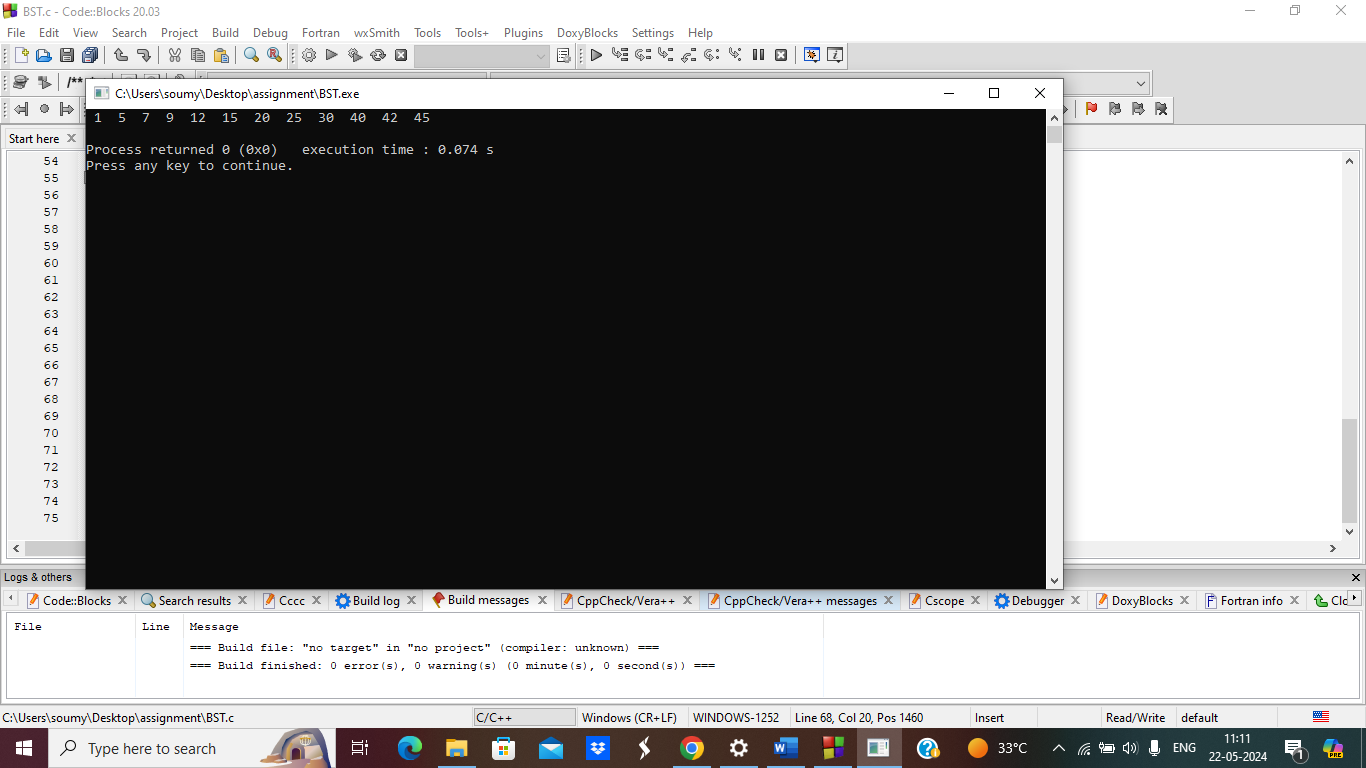
insert(root, 42);

inorder(root);

printf("\n");

return 0;

}



**2.Binary tree(in order, pre order, post order)**

#include <stdio.h>

#include <limits.h>

struct Node {

int data;

struct Node\* left;

struct Node\* right;

};

struct Node\* newNode(int key) {

struct Node\* node = (struct Node\*)malloc(sizeof(struct Node));

node->data = key;

node->left = node->right = NULL;

return node;

}

void inOrderTraversal(struct Node\* root) {

if (root != NULL) {

inOrderTraversal(root->left);

printf("%d ", root->data);

inOrderTraversal(root->right);

}

}

void preOrderTraversal(struct Node\* root) {

if (root != NULL) {

printf("%d ", root->data);

preOrderTraversal(root->left);

preOrderTraversal(root->right);

}

}

void postOrderTraversal(struct Node\* root) {

if (root != NULL) {

postOrderTraversal(root->left);

postOrderTraversal(root->right);

printf("%d ", root->data);

}

}

int main() {

// Constructing a sample binary tree

struct Node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

printf("In-order Traversal: ");

inOrderTraversal(root);

printf("\n");

printf("Pre-order Traversal: ");

preOrderTraversal(root);

printf("\n");

printf("Post-order Traversal: ");

postOrderTraversal(root);

printf("\n");

return 0;

}

