

BinarySearchTree

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *left;
    struct node *right;
};

struct node* createNode(int value) {
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

struct node* insert(struct node* root, int value) {
    if (root == NULL)
        return createNode(value);
    if (value < root->data)
        root->left = insert(root->left, value);
    else if (value > root->data)
```

```
    root->right = insert(root->right, value);

    return root;

}
```

```
void inorder(struct node* root) {

    if (root != NULL) {

        inorder(root->left);

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

        inorder(root->right);

    }

}
```

```
void preorder(struct node* root) {

    if (root != NULL) {

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

        preorder(root->left);

        preorder(root->right);

    }

}
```

```
void postorder(struct node* root) {

    if (root != NULL) {

        postorder(root->left);

        postorder(root->right);
```

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

}

}

int main() {

    struct node* root = NULL;

    int choice, value;

    printf("\n\n--- Binary Search Tree Menu ---");

    printf("\n1. Insert element");

    printf("\n2. In-order Traversal");

    printf("\n3. Preorder Traversal");

    printf("\n4. Postorder Traversal");

    printf("\n5. Exit");



do {

    printf("\nEnter your choice: ");

    scanf("%d", &choice);

    switch (choice) {

        case 1:

            printf("Enter value to insert: ");

            scanf("%d", &value);

            root = insert(root, value);

            break;

    }

}
```

case 2:

```
printf("In-order Traversal: ");
inorder(root);
break;
```

case 3:

```
printf("Preorder Traversal: ");
preorder(root);
break;
```

case 4:

```
printf("Postorder Traversal: ");
postorder(root);
break;
```

case 5:

```
printf("Exiting program...");
break;
```

default:

```
printf("Invalid choice!");
}
} while (choice != 5);
```

```
return 0;  
}  
  
Output:  
--- Binary Search Tree Menu ---  
1. Insert element  
2. In-order Traversal  
3. Preorder Traversal  
4. Postorder Traversal  
5. Exit  
Enter your choice: 1  
Enter value to insert: 10  
  
Enter your choice: 1  
Enter value to insert: 5  
  
Enter your choice: 1  
Enter value to insert: 15  
  
Enter your choice: 2  
In-order Traversal: 5 10 15  
Enter your choice: 3  
Preorder Traversal: 10 5 15  
Enter your choice: 4  
Postorder Traversal: 5 15 10  
Enter your choice: 5  
Exiting program...  
Process returned 0 (0x0) execution time : 19.466 s  
Press any key to continue.
```