

```

BSTC - CodeBlocks20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools- Plugins DocyBlocks Settings Help
Start here X doublylinkedlist X Sort_Revers_Concate X BSTC X
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node *left, *right;
7 }
8
9 struct Node * createNode(int value)
10 {
11     struct Node *newNode = (struct Node *) malloc(sizeof(struct Node));
12     newNode->data = value;
13     newNode->left = newNode->right = NULL;
14     return newNode;
15 }
16
17
18 struct Node* insert(struct Node* root, int value)
19 {
20     if (root == NULL)
21         return createNode(value);
22     if (value < root->data)
23         root->left = insert(root->left, value);
24     else if (value > root->data)
25         root->right = insert(root->right, value);
26     return root;
27 }
28
29
30 void inorder(struct Node *root) {
31     if (root == NULL) return;
32     inorder(root->left);
33     printf("%d ", root->data);
34     inorder(root->right);
35 }
36
37
38 void preorder(struct Node *root) {
39     if (root == NULL) return;
40     printf("%d ", root->data);
41     preorder(root->left);
42     preorder(root->right);
43 }
44
45
46 void postorder(struct Node *root) {
47     if (root == NULL) return;
48     postorder(root->left);
49     postorder(root->right);
50     printf("%d ", root->data);
51 }
52
53
54 void display(struct Node *root) {
55     printf("BST Elements (Inorder): ");
56     inorder(root);
57     printf("\n");
58 }
59
60 int main() {
61     struct Node *root = NULL;
62     int choice, value;
63
64     while (1) {
65         printf("\n--- Binary Search Tree Menu ---\n");
66         printf("1. Insert into BST\n");
67         printf("2. Inorder Traversal\n");
68         printf("3. Preorder Traversal\n");
69         printf("4. Postorder Traversal\n");
70         printf("5. Display BST\n");
71         printf("6. Exit\n");
72         printf("Enter choice: ");
73         scanf("%d", &choice);
74
75         switch (choice) {
76             case 1:
77                 printf("Enter value to insert: ");
78                 scanf("%d", &value);
79                 root = insert(root, value);
80                 break;
81             case 2:
82                 printf("Inorder Traversal: ");
83                 inorder(root);
84                 printf("\n");
85                 break;
86             case 3:
87                 printf("Preorder Traversal: ");
88                 preorder(root);
89                 printf("\n");
90                 break;
91             case 4:
92                 printf("Postorder Traversal: ");
93                 postorder(root);
94                 printf("\n");
95                 break;
96             case 5:
97                 display(root);
98                 break;
99             case 6:
100                 exit(0);
101         }
102     }
103 }
104
105

```

```

BSTC - CodeBlocks20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools- Plugins DocyBlocks Settings Help
Start here X doublylinkedlist X Sort_Revers_Concate X BSTC X
52
53 void display(struct Node *root) {
54     printf("BST Elements (Inorder): ");
55     inorder(root);
56     printf("\n");
57 }
58
59 int main() {
60     struct Node *root = NULL;
61     int choice, value;
62
63     while (1) {
64         printf("\n--- Binary Search Tree Menu ---\n");
65         printf("1. Insert into BST\n");
66         printf("2. Inorder Traversal\n");
67         printf("3. Preorder Traversal\n");
68         printf("4. Postorder Traversal\n");
69         printf("5. Display BST\n");
70         printf("6. Exit\n");
71         printf("Enter choice: ");
72         scanf("%d", &choice);
73
74         switch (choice) {
75             case 1:
76                 printf("Enter value to insert: ");
77                 scanf("%d", &value);
78                 root = insert(root, value);
79                 break;
80             case 2:
81                 printf("Inorder Traversal: ");
82                 inorder(root);
83                 printf("\n");
84                 break;
85             case 3:
86                 printf("Preorder Traversal: ");
87                 preorder(root);
88                 printf("\n");
89                 break;
90             case 4:
91                 printf("Postorder Traversal: ");
92                 postorder(root);
93                 printf("\n");
94                 break;
95             case 5:
96                 display(root);
97                 break;
98             case 6:
99                 exit(0);
100         }
101     }
102 }
103
104
105

```

```
DickensOP@STime
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 1
Enter value to Insert: 23
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 1
Enter value to Insert: 11
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 1
Enter value to Insert: 455
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 5
BST Elements (Inorder): 11 23 455
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 3
Preorder Traversal: 23 11 455
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 4
Postorder Traversal: 11 455 23
--- Binary Search Tree Menu ---
1. Insert into BST
```

- 8) Binary search tree implement and traverse the tree
all the methods inorder, preorder, postorder.

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

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

```
struct node {
```

```
    int data;
```

```
    struct node * left, * 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 = value insert (root->right, value);
```

```
    return root;
```

```
}
```

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

```
    if (root == NULL) return;
```

```
    inorder (root->left);
```

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

```
    inorder (root->right);
```

```
}
```

```
void preorder(struct node* root) {
    if (root == NULL) return;
    printf("%d", root->data);
    preorder(root->left);
    preorder(root->right);
}

void postorder(struct node* root) {
    if (root == NULL) return;
    postorder(root->left);
    postorder(root->right);
    printf("%d", root->data);
}

void display(struct node* root) {
    printf("BST Elements (Inorder): ");
    inorder(root);
    printf("\n");
}

int main() {
    struct Node * root = NULL;
    int choice, value;
    while (1) {
        printf("\n --- Binary search Tree menu --- \n");
        printf("1. Insert into BST \n");
        printf("2. Inorder Traversal \n");
        printf("3. preorder Traversal \n");
        printf("4. postorder Traversal \n");
        printf("5. display \n");
        printf("6. Exit \n");
        printf("Enter choice: ");
        scanf("%d", &choice);
```



```
switch (choice) {
```

```
case 1:
```

```
    printf("Enter value of insert");  
    scanf("%d", &value);  
    root = insert(root, value);  
    break;
```

```
case 2:
```

```
    printf("Inorder traversal:");  
    inorder(root);  
    printf("\n");  
    break;
```

```
case 3:
```

```
    printf("preorder traversal");  
    preorder(root);  
    printf("\n");  
    break;
```

```
case 4:
```

```
    printf("postorder Traversal:");  
    postorder(root);  
    printf("\n");  
    break;
```

```
case 5:
```

```
    display(root);  
    break;
```

```
case 6:
```

```
    Exit(0);
```

```
    }
```

```
    }
```

```
}
```

DATE:

Output

... Binary search tree menu...

1. Insert into BST
2. Inorder Traversal
3. preorder Traversal
4. postorder Traversal
5. Display BST
6. Exit

Enter choice: 1

Enter value to insert: 23 45 567

Enter choice: 2

Inorder Traversal: 23 45 567

Enter choice: 3

preorder Traversal: 23 45 567

Enter choice: 4

postorder Traversal: 567 45 23

Enter choice: 5

BST (element): 23 45 567

Enter choice: 6

~~process returned 0 (0x0) execution time: 363.927s~~

~~Enter~~