

**LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA**

Judul: Tree



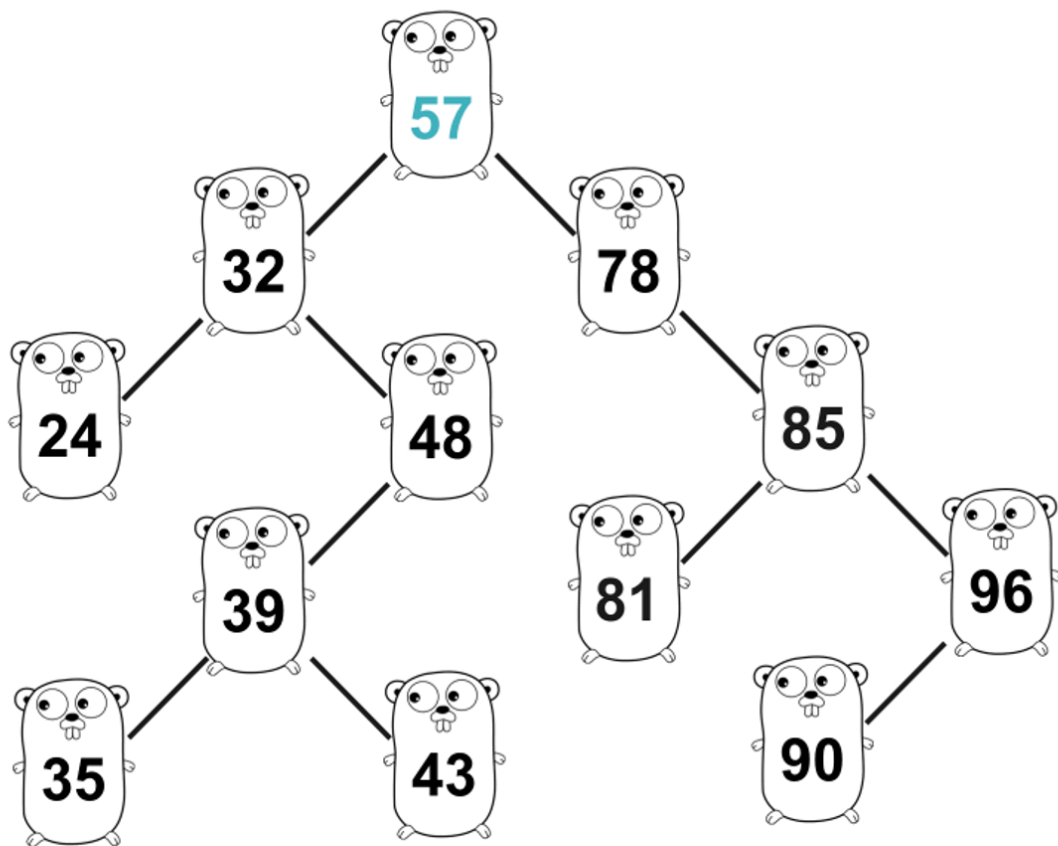
DISUSUN OLEH
Ilham Nur Romdoni M0520038

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS SEBELAS MARET
2021**

Program Binary Tree dengan Tree Traversal Postorder

Tree traversal adalah bentuk grafik *traversal*. Ini melakukan pengecekan atau pencetakan setiap *node* di *tree*. *Traversal postorder* dari *binary search tree* melakukan kunjungan terhadap masing-masing *node* di *tree* dalam urutan (*Left, Right, Root*).

Traversal postorder dari *binary tree* adalah sebagai berikut.



Postorder Traversal adalah: 24 35 43 39 48 32 81 90 96 85 78 57

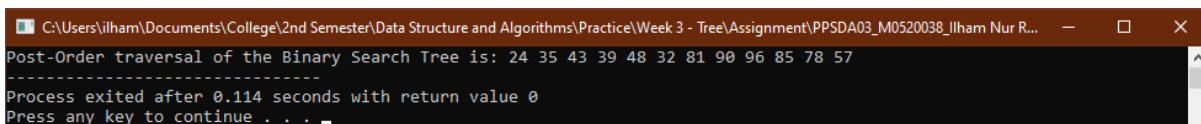
Program untuk melakukan *postorder recursive traversal* diberikan sebagai berikut.

```

1  #include<iostream>
2  using namespace std;
3  struct node {
4      int data;
5      struct node *left;
6      struct node *right;
7  };
8  struct node *createNode(int val) {
9      struct node *temp = (struct node *)malloc(sizeof(struct node));
10     temp->data = val;
11     temp->left = temp->right = NULL;
12     return temp;
13 }
14 void postorder(struct node *root) {
15     if (root != NULL) {
16         postorder(root->left);
17         postorder(root->right);
18         cout<<root->data<<" ";
19     }
20 }
21 struct node* insertNode(struct node* node, int val) {
22     if (node == NULL) return createNode(val);
23     if (val < node->data)
24         node->left = insertNode(node->left, val);
25     else if (val > node->data)
26         node->right = insertNode(node->right, val);
27     return node;
28 }
29 int main() {
30     struct node *root = NULL;
31     root = insertNode(root, 57);
32     insertNode(root, 78);
33     insertNode(root, 85);
34     insertNode(root, 96);
35     insertNode(root, 90);
36     insertNode(root, 81);
37     insertNode(root, 32);
38     insertNode(root, 48);
39     insertNode(root, 39);
40     insertNode(root, 43);
41     insertNode(root, 35);
42     insertNode(root, 24);
43     cout<<"Post-Order traversal of the Binary Search Tree is: ";
44     postorder(root);
45     return 0;
46 }

```

Output



```

C:\Users\ilham\Documents\College\2nd Semester\Data Structure and Algorithms\Practice\Week 3 - Tree\Assignment\PPSDA03_M0520038_Ilham Nur R...
Post-Order traversal of the Binary Search Tree is: 24 35 43 39 48 32 81 90 96 85 78 57
Process exited after 0.114 seconds with return value 0
Press any key to continue . . .

```

Dalam program di atas, struktur `node` membuat `node` dari sebuah *tree*. Struktur ini adalah *self referential structure* karena berisi *pointers* dari tipe `struct node`. Struktur ini ditunjukkan sebagai berikut.

```

3 struct node {
4     int data;
5     struct node *left;
6     struct node *right;
7 };

```

Fungsi `createNode()` membuat temp node dan mengalokasikan memori menggunakan `malloc`. Nilai data `val` disimpan dalam data temp. `NULL` disimpan di *pointers* temp left dan right. Ini ditunjukkan dengan cuplikan kode berikut.

```

8 struct node *createNode(int val) {
9     struct node *temp = (struct node *)malloc(sizeof(struct node));
10    temp->data = val;
11    temp->left = temp->right = NULL;
12    return temp;
13 }

```

Fungsi `postorder()` mengambil `root` dari *binary tree* sebagai argumen dan mencetak elemen *tree* dalam `postorder`. Ini adalah fungsi *recursive*. Ini ditunjukkan menggunakan kode berikut.

```

14 void postorder(struct node *root) {
15     if (root != NULL) {
16         postorder(root->left);
17         postorder(root->right);
18         cout<<root->data<<" ";
19     }
20 }

```

Fungsi `insertNode()` menyisipkan nilai yang diperlukan dalam *binary tree* pada posisi yang benar. Jika node `NULL`, maka `createNode` dipanggil. Jika tidak, posisi yang benar untuk node ditemukan di *tree*. Ini dapat diamati dalam cuplikan kode berikut.

```

21 struct node* insertNode(struct node* node, int val) {
22     if (node == NULL) return createNode(val);
23     if (val < node->data)
24         node->left = insertNode(node->left, val);
25     else if (val > node->data)
26         node->right = insertNode(node->right, val);
27     return node;
28 }

```

Dalam fungsi `main()`, node `root` pertama kali didefinisikan sebagai `NULL`. Kemudian semua node dengan nilai yang diperlukan dimasukkan ke dalam *binary search tree*. Hal ini ditunjukkan di bawah ini.

```

29 int main()
30     struct node *root = NULL;
31     root = insertNode(root, 57);
32     insertNode(root, 78);
33     insertNode(root, 85);
34     insertNode(root, 96);
35     insertNode(root, 90);
36     insertNode(root, 81);
37     insertNode(root, 32);
38     insertNode(root, 48);
39     insertNode(root, 39);
40     insertNode(root, 43);
41     insertNode(root, 35);
42     insertNode(root, 24);

```

Akhirnya, fungsi `postorder()` dipanggil menggunakan `node root` dari *tree* dan semua nilai *tree* ditampilkan di `postorder`. Ini diberikan di bawah ini.

```

43     cout<<"Post-Order traversal of the Binary Search Tree is: ";
44     postorder(root);

```

Referensi

Atqia, Nuha Lina.2021.“Week 3 Tree (google.com)”, diakses pada 13 April 2021.

Yadav, Chandu.2018.“C++ Program to Perform Postorder Recursive Traversal of a Given Binary Tree (tutorialspoint.com)”, diakses pada 19 April 2021.