# KEGIATAN PRAKTIKUM PERTEMUAN 12

**STRUKTUR DATA HIRARKI BAGIAN II (BST DAN AVL TREE)**

## Perhatikan baris kode berikut:

# include <iostream> # include <cstdlib>

using namespace std; **1**

struct node {

int info;

struct node \*left; struct node \*right;

}\*root;

class BST {

public:

void fun\_1(int, node \*\*, node \*\*); void fun\_2(int);

void fun\_3(node \*, node \*); void fun\_4(node \*, int); void case\_a(node \*,node \*); void case\_b(node \*,node \*); void case\_c(node \*,node \*); BST()

{

root = NULL;

}};

int main()

{

int choice, num; **2**

BST bst;

node \*temp; while (1)

{

cout<<" "<<endl; cout<<"Operasi pada Binary Search Tree"<<endl; cout<<" "<<endl;

cout<<" "<<endl;

cout<<" "<<endl;

cout<<" "<<endl;

cout<<" "<<endl;

cout<<"Ketik angka yang akan anda pilih: "; cin>>choice;

switch(choice)

{

case 1:

cout<<" "<<endl;

bst.fun\_4(root,1); cout<<endl;

break; case 2:

if (root == NULL){ cout<<"Tree kosong"<<endl; continue;}

cout<<"Ketik angka ... ... ... dari BST: ";

cin>>num; bst.fun\_2(num); break;

case 3:

temp = new node;

cout<<"Ketik ... ... ... ... ... ke BST: ";

cin>>temp->info; bst.fun\_3(root, temp); break;

case 4:

exit(1); default:

cout<<"Tidak ada angka tersebut!"<<endl;

}

}

}

void BST::fun\_1(int item, node \*\*par, node \*\*loc)

{ **3**

node \*ptr, \*ptrsave;

if (root == NULL){

\*loc = NULL;

\*par = NULL; return;}

if (item == ){

\*loc = root;

\*par = NULL; return;}

if (item < root->info)

ptr = ;

else

ptr = ;

ptrsave = root; while (ptr != NULL)

{

if (item == ptr->info)

{

\*loc = ptr;

\*par = ptrsave; return;

}

ptrsave = ptr;

if (item < ptr->info)

ptr = ;

else

ptr = ;

}

\*loc = NULL;

\*par = ptrsave;

}

void BST::fun\_3(node \*tree, node \*newnode)

{ **4**

if (root == )

{

root = ;

root->info = newnode->info; root->left = NULL;

root->right = ;

cout<<"Node Root ditambahkan ke dalam Tree"<<endl; return;

}

if (... ... ... ... ... == )

{

cout<<"Node yang akan ditambahkan sudah berada di dalam tree"<<endl; return;

}

if (tree->info > newnode->info)

{

if (tree->left != NULL)

{

}

else

{

}

}

... ... ... ... ... (tree->left, newnode);

tree->left = newnode;

(tree->left)->left = ;

(tree->left)->right = ;

return;

else

{

if (... ... ... ... ... != NULL)

{

fun\_3(... ... ... ... ..., newnode);

}

else

{

tree->right = ;

(tree->right)->left = NULL; (tree->right)->right = NULL; return;

}

}

}

void BST::fun\_2(int item)

{

node \*parent, \*location; **5**

if ( )

{

cout<<"Tree dalam keadaan empty"<<endl; return;

}

fun\_1(item, &parent, &location); if (location == NULL)

{

cout<<"Node yang dicari tidak dapat ditemukan di dalam tree"<<endl; return;

}

if (location->left == NULL && location->right == NULL)

... ... ... ... ... (parent, location);

if (location->left != NULL && location->right == NULL)

... ... ... ... ... (parent, location);

if (location->left == NULL && location->right != NULL)

... ... ... ... ... (parent, location);

if (location->left != NULL && location->right != NULL) case\_c(parent, location);

free(location);

}

void BST::case\_a(node \*par, node \*loc )

{

if (par == NULL) **6**

{

}

else

{

}

}

root = NULL;

if (loc == par->left) par->left = NULL;

else

par->right = NULL;

void BST::case\_b(node \*par, node \*loc)

{

node \*child;

if (loc->left != NULL) **7**

child = loc->left;

else

child = loc->right;

if (par == NULL)

{

}

else

{

}

}

root = child;

if (loc == par->left) par->left = child;

else

par->right = child;

void BST::case\_c(node \*par, node \*loc)

{

node \*ptr, \*ptrsave, \*suc, \*parsuc; **8**

ptrsave = loc; ptr = loc->right;

while (ptr->left != NULL)

{

ptrsave = ptr; ptr = ptr->left;

}

suc = ptr;

parsuc = ptrsave;

if (suc->left == NULL && suc->right == NULL) case\_a(parsuc, suc);

else

case\_b(parsuc, suc);

if (par == NULL)

{

root = suc;

}

}

}

cout<<ptr->info; fun\_4(ptr->left, level+1);

}

cout<<" ";

**9**

void BST::fun\_4(node \*ptr, int level)

{

int i;

if (ptr != NULL)

{

fun\_4(ptr->right, level+1); cout<<endl;

if (ptr == root) cout<<"Root->: ";

else

{

for (i = 0;i < level;i++)

}

else

{

if (loc == par->left) par->left = suc;

else

par->right = suc;

}

suc->left = loc->left; suc->right = loc->right;

## Berdasarkan kode program di atas, jawab pertanyaan berikut ini..

* 1. Pada potongan kode (snippet) nomor 1, jelaskan hasil eksekusi dari baris kode tersebut?
  2. Lengkapi potongan kode (snippet) nomor 3, lakukan penelusuran dan jelaskan apa kegunaan dari fungsi tersebut?
  3. Lengkapi potongan kode (snippet) nomor 4, lakukan penelusuran dan jelaskan apa kegunaan dari fungsi tersebut?
  4. Lengkapi potongan kode (snippet) nomor 5, lakukan penelusuran dan jelaskan apa kegunaan dari fungsi tersebut?
  5. Jelaskan procedural dari potongan kode 6, 7, 8. Apa kegunaanya?
  6. Lengkapi potongan kode (snippet) nomor 2, dan implementasikan keseluruhan kode agar dapat menjalankan perintah *insertion*, dan *deletion* BST dengan kriteria sebagai berikut: Insertion: 32 – 11 – 13 – 46 – 78 – 4 – 16 – 8 – 25 – 37 – 56 – 61 – 44 – 29

Deletion: 8 – 46 – 32 - 5

Catatan: *PrtSc* hasil eksekusi kode tersebut.

1. **Essai**

Berikan suatu penjelasan dan gambaran/ide yang dapat dilakukan agar kode pada poin A di atas, dapat berfungsi sebagai AVL tree. Prosedur apa yang perlu dilengkapi, bagaimana prosesnya? (**jelaskan dengan minimal 100 kata**)