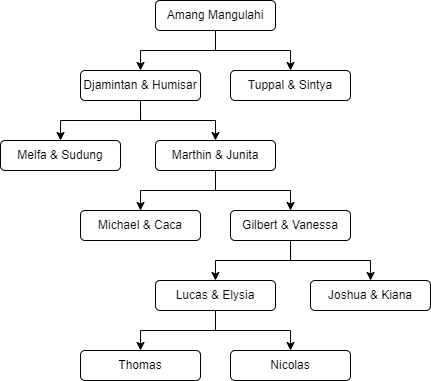
Gilbert Immanuel Hasiholan

Diagram

Description automatically generated5027211056 | Teknologi Informasi | Struktur Data-B

Text

Description automatically generated

#include <iostream>

#include <string>

using namespace std;

*// node*

struct Node{

string label;

Node \*left, \*right, \*parent;

};

*// variabel pointer global*

Node \*root, \*newNode;

void isEmpty() {

if( root != NULL)

return;

}

void createNewTree(string label)

{

if(!isEmpty)

return;

else {

root = new Node();

root->label = label;

root->left = NULL;

root->right = NULL;

root->parent = NULL;

}

}

*// insert Left*

Node \*insertLeft( string label, Node \*node){

if( root == NULL ){

cout << "\nBuat tree terlebih dahulu!!" << endl;

return NULL;

}else{

if( node->left != NULL ){

*// if available*

cout << "\nNode "<< node->label << " sudah ada anak kiri!!" << endl;

return NULL;

}else{

*// if not available*

newNode = new Node();

newNode->label = label;

newNode->left = NULL;

newNode->right = NULL;

newNode->parent = node;

node->left = newNode;

cout << "\nNode " << label << " berhasil ditambahkan ke anak kiri " << newNode->parent->label << endl;

return newNode;

}

}

}

*// insert right*

Node \*insertRight( string label, Node \*node )

{

if( root == NULL ){

cout << "\nBuat tree terlebih dahulu!!" << endl;

return NULL;

}else{

*// to check if left son exist*

if( node->right != NULL ){

*// exist*

cout << "\nNode " << node->label << " sudah ada anak kanan!!" << endl;

return NULL;

}else{

*// not exist*

newNode = new Node();

newNode->label = label;

newNode->left = NULL;

newNode->right = NULL;

newNode->parent = node;

node->right = newNode;

cout << "\nNode " << label << " berhasil ditambahkan ke anak kanan " << newNode->parent->label << endl;

return newNode;

}

}

}

*// update*

void update(string label, Node \*node)

{

if( !root ){

cout << "\nBuat tree terlebih dahulu!!" << endl;

}else{

if( !node )

cout << "\nNode yang ingin diganti tidak ada!!" << endl;

else{

string temp = node->label;

node->label = label;

cout << "\nLabel node " << temp << " berhasil diubah menjadi " << label << endl;

}

}

}

*// Tranversal*

*// preOrder*

void preOrder( Node \*node = root )

{

if( !root )

cout << "\nBuat tree terlebih dahulu!!" << endl;

else{

if( node != NULL ){

cout << node->label << ", ";

preOrder(node->left);

preOrder(node->right);

}

}

}

*// inOrder*

void inOrder( Node \*node = root )

{

if( !root )

cout << "\nBuat tree terlebih dahulu!!" << endl;

else{

if( node != NULL ){

inOrder(node->left);

cout << node->label << ", ";

inOrder(node->right);

}

}

}

*// postOrder*

void postOrder( Node \*node = root )

{

if( !root )

cout << "\nBuat tree terlebih dahulu!!" << endl;

else{

if( node != NULL ){

postOrder(node->left);

postOrder(node->right);

cout << node->label << ", ";

}

}

}

int main() {

createNewTree("Amang Mangulahi");

Node \*nodeB, \*nodeC, \*nodeD, \*nodeE, \*nodeF, \*nodeG, \*nodeH, \*nodeI, \*nodeJ, \*nodeK;

nodeB = insertLeft("Djamintan & Humisar", root);

nodeC = insertRight("Tuppal & Sintya", root);

nodeD = insertLeft("Melfa & Sudung", nodeB);

nodeE = insertRight("Marthin & Junita", nodeB);

nodeF = insertLeft("Michael & Caca", nodeE);

nodeG = insertRight("Gilbert & Vanessa", nodeE);

nodeH = insertLeft("Lukas & Elysia", nodeG);

nodeI = insertRight("Joshua & Kiana", nodeG);

nodeJ = insertLeft("Thomas", nodeH);

nodeK = insertRight("Nicolas", nodeH);

cout << "\nPreOrder :" << endl;

preOrder();

cout << "\n" << endl;

cout << "InOrder :" << endl;

inOrder();

cout << "\n" << endl;

cout << "PostOrder :" << endl;

postOrder();

cout << "\n" << endl;

update("Sibarani & Megawati", nodeC);

cout << "\nPreOrder :" << endl;

preOrder();

cout << "\n" << endl;

cout << "InOrder :" << endl;

inOrder();

cout << "\n" << endl;

cout << "PostOrder :" << endl;

postOrder();

cout << "\n" << endl;

}