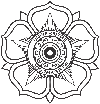
**UNIVERSITAS GADJAH MADA**Fakultas Teknik  
Departemen Teknik Elektro dan Teknologi Informasi

LAPORAN TUGAS

ALGORITMA DAN STRUKTUR DATA – KELAS B

**Expression Tree, Postorder Traversal, Postfix Evaluation**

DISUSUN OLEH

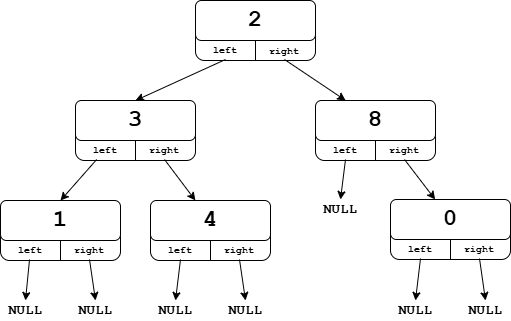
|  |  |  |
| --- | --- | --- |
| Muhammad Nafal Zakin Rustanto | 24/535255/TK/59364 | Teknologi Informasi |
| Muhammad Fachri Akbar | 24/538155/TK/59679 | Teknik Elektro |

# BINARY TREE

*Binary tree* adalah sebuah struktur data hierarkis non-linear *(tree)* yang setiap *node*-nya memiliki paling banyak 2 anak *(child)* yang selanjutnya disebut *left child* dan *right child*. Dalam program dengan bahasa pemrograman C++, setiap node dalam *binary tree* umumnya dikarakterisasi dalam sebuah *struct* dengan 3 komponen yaitu data dalam *node* tersebut dan 2 buah pointer *(left* dan *right*) yang menghubungkan *node* dengan *left child* dan *right child-*nya.

Sedangkan untuk membuat *binary tree* pada program C++, pertama-tama dibuat sebuah *node root*, kemudian pointer *left* dan *right-*nya diset *NULL*. Setelah itu, untuk setiap pointer pada tiap *node*, dapat dibuat *node baru* dengan cara yang sama, yaitu mengalokasikan memori untuk *node* baru dan mengatur nilai serta pointer *left* dan *right*-nya ke *NULL*. Proses ini dapat dilakukan secara rekursif atau iteratif. Algoritma untuk membangun sebuah *binary tree* pada program tersebut beserta contohnya, dapat dilihat sebagai berikut.

|  |
| --- |
| **Algoritma 1.** Membangun *Binary Tree* pada Program C++ |
| DICTIONARY  STRUCTURE Node:  data: string  left: pointer to Node  right: pointer to Node  FUNCTION newNode(data)  DEFINE node AS new Node  SET node->data = data  SET node->left = node->right = NULL  RETURN node  EXAMPLE  DEFINE root AS Node  SET root = CALL newNode(2)  SET root->left = CALL newNode(3)  SET root->left->left = CALL newNode(1)  SET root->left->right = CALL newNode(4)  SET root->right = CALL newNode(8)  SET root->right->right = CALL newNode(0) |

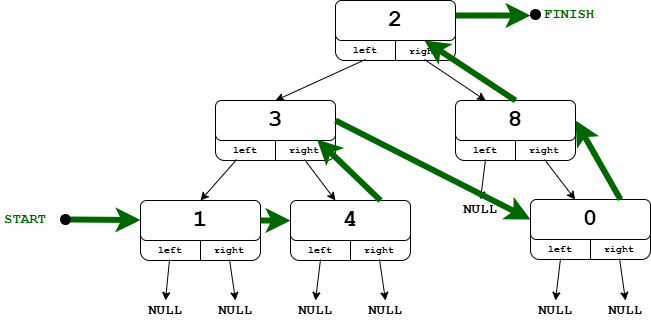
*Binary tree* yang akan terbentuk dengan contoh *binary tree* yang dibangun pada algoritma diatas, dapat dilihat pada Gambar 1 sebagai berikut

**Gambar 1.** Representasi *Binary Tree* yang dibangun pada Algoritma 1

# POSTORDER TRAVERSAL

*Postorder traversal* adalah salah satu jenis metode *depth first traversal* untuk mengunjungi tiap *node* sebuah *binary tree*. Pada metode ini, jika sebuah *binary tree* diasumsikan terdiri atas *subtree*, maka *node* pada *subtree* kiri akan dikunjungi lebih dulu, dilanjutkan *node* pada *subtree* kanan, dan diakhiri dengan mengunjungi *root*; digambarkan pada Algoritma 2 berikut.

|  |
| --- |
| **Algoritma 2.** *Postorder Traversal* pada Sebuah *Binary Tree* |
| DICTIONARY  STRUCTURE Node:  data: string  left: pointer to Node  right: pointer to Node  DEFINE root AS Node  FUNCTION traversePostOrder(root)  IF root IS NOT NULL THEN  CALL traversePostOrder(root->left)  CALL traversePostOrder(root->right)  PRINT node->data  END IF |

Hasil dari sebuah *postorder traversal* umumnya disebut *postfix*. Jika *postorder traversal* dilakukan pada tree di Gambar 1, maka akan dihasilkan *postfix* 1 4 3 0 8 2, dan urutan *node* yang dikunjungi dapat dilihat pada Gambar 2 berikut.

**Gambar 2.** *Postorder Traversal* pada Tree di Gambar 1

# EXPRESSION TREE DAN POSTFIX EVALUATION

# IMPLEMENTASI DALAM KODE

# ILUSTRASI PENGGUNAAN PROGRAM

# LAMPIRAN