

**Laporan Praktikum**

**Algoritma dan Struktur Data**

Ganjil 2023/2024

Program Studi Teknik Informatika

Institut Teknologi Sumatera

|  |  |
| --- | --- |
| **Modul :** | **Binary Tree** |
| **Nama :** | **Muhammad Yusuf** |
| **NIM :** | **122140193** |
| **Kelas (Kelas Asal) :** | **ASD RA** |

Instruksi sederhana :

* Disarankan kepada **Praktikan Algoritma Struktur Data** untuk mengeditnya menggunakan Google Docs agar tidak berantakan dan rapi,
* Silahkan mengganti **Nama Modul** baik yang ada pada **Cover** dan **Header** sesuai dengan materi praktikum,
* Gunakan text styling seperti **Heading 1**, **Normal Text** yang telah terformat / Text Style lainnya yang digunakan untuk menjaga estetika laporan,
* Gunakan [**Syntax Highlighter**](https://highlight.hohli.com/index.php)untuk merapikan kode yang sudah Praktikan buat ke dalam Laporan Praktikum.

**Materi Praktikum**

Binary Tree

**Link repl.it Source Code**

<https://replit.com/@muhammad1221401/Muhammad-Yusuf-Prak-ASD-RA122140193#minggu5/latihan.cpp>

**Source Code**

Gunakan [**Syntax Highlighter**](https://highlight.hohli.com/index.php)untuk merapikan Source Code yang dipindahkan dari text editor anda ke dokumen ini.

1. #include <iostream>
2. #include <iomanip>
3. using namespace std;
5. **struct Tree{**
6. int info;
8. Tree \*left;
9. Tree \*right;
10. **Tree \*parent;**
11. };
13. Tree \*Initialize(int value){
14. Tree \*newNode = new Tree;
16. newNode->info = value;
17. newNode->left = nullptr;
18. newNode->right = nullptr;
19. newNode->parent = nullptr;
21. return newNode;
22. }
24. bool IsTreeEmpty(Tree \*P) { return P == nullptr; }
26. void InsertLeft(Tree \*P, int value){
27. Tree \*newNode = Initialize(value);
29. if (P->left == nullptr){
30. **newNode->parent = P;**
31. P->left = newNode;
32. cout << "Success insert left node with value " << value << " with " << P->info << " as parrent**\n**";
33. }else{
34. cout << "Tree left was declared!";
35. **}**
36. }
38. void InsertRight(Tree \*P, int value){
39. Tree \*newNode = Initialize(value);
40. **if (P->right == nullptr){**
41. newNode->parent = P;
42. P->right = newNode;
43. cout << "Success insert right node with value " << value << " with " << P->info << " as parrent**\n**";
44. }
45. **else{**
46. cout << "Tree right was declared!";
47. }
48. }
50. **void PreOrder(Tree \*P){**
51. if (IsTreeEmpty(P)){
52. }else{
53. cout << P->info << " ";
54. PreOrder(P->left);
55. **PreOrder(P->right);**
56. }
57. }
59. void InOrder(Tree \*P){
60. **if (IsTreeEmpty(P)){**
61. }else{
62. InOrder(P->left);
63. cout << P->info << " ";
64. InOrder(P->right);
65. **}**
66. }
68. void display(Tree \*P){
69. cout << "<>Pre Order : ";
70. **PreOrder(P);**
71. cout << endl;
72. cout << "<>In Order : ";
73. InOrder(P);
74. cout << endl;
75. **}**
77. void displayIndent(Tree \*P, int level=0){
78. if (P != nullptr){
79. displayIndent(P->right, level + 1);
80. **cout << setw(level \* 3) << " " << P->info << endl;**
81. displayIndent(P->left, level + 1);
82. }
83. }
85. **int main(){**
86. Tree \*P = Initialize(1);
87. InsertLeft(P, 2);
88. InsertRight(P, 3);
90. **InsertLeft(P->left, 4);**
91. InsertRight(P->left, 5);
93. InsertLeft(P->right, 6);
94. InsertRight(P->right, 7);
96. InsertLeft(P->left->left, 8);
98. cout << endl << "Display tree : " << endl;
99. display(P);
101. cout << "Display tree with indent : " << endl;
102. displayIndent(P);
104. return 0;
105. **}**

**Dokumentasi Hasil Running**

|  |
| --- |
|  |
| **Gambar 1.** Output Code Binary Tree Latihan Minggu 5.  Program di atas adalah implementasi pohon biner dalam C++. Setiap node dalam pohon memiliki informasi (info), pointer ke anak kiri (left), anak kanan (right), dan pointer ke induk (parent). Fungsi `Initialize` digunakan untuk membuat dan menginisialisasi node baru dengan nilai tertentu. Fungsi `IsTreeEmpty` mengembalikan true jika pohon kosong. Fungsi `InsertLeft` dan `InsertRight` digunakan untuk menyisipkan node baru di sebelah kiri atau kanan dari node P. Fungsi `PreOrder` dan `InOrder` melakukan penelusuran pohon secara pre-order dan in-order, masing-masing. Fungsi `display` menampilkan hasil penelusuran pre-order dan in-order. Fungsi `displayIndent` menampilkan pohon dengan indentasi sesuai tingkatan node. Dalam fungsi `main`, sebuah pohon dibangun dengan beberapa node, dan hasil penelusuran serta tampilan dengan indentasi ditampilkan. |

|  |
| --- |
|  |
|  |