Laporan Praktikum

Algoritma dan Struktur Data

Ganjil 2023/2024
Program Studi Teknik Informatika
Institut Teknologi Sumatera

Modul : Binary Tree

Nama : Aulia Putri Sayidina

NIM : 122140060

Kelas (Kelas Asal) : RC

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** untuk merapikan kode yang sudah Praktikan buat ke dalam Laporan Praktikum.

Dirancang oleh Kevin Simorangkir
Terinspirasi dari Format Tata Tulis Laporan Embedded

Materi Praktikum

PENDAHULUAN:

Dalam struktur data, Binary Tree adalah suatu struktur hierarki yang terdiri dari simpul-simpul (node) yang dihubungkan oleh sisi atau cabang. Setiap node dalam pohon memiliki dua anak, yaitu anak kiri dan anak kanan. Konsep Binary Tree digunakan untuk mengorganisir data dengan cara yang efisien dan efektif, terutama untuk operasi pencarian, penambahan, dan penghapusan.

FUNGSI BINARY TREE:

Binary Tree memberikan solusi untuk menyimpan dan mengakses data dengan cara yang terstruktur dan efisien. Dengan pembagian hierarki yang jelas, Binary Tree memfasilitasi pencarian dan manipulasi data dengan cepat.

OPERASI DASAR BINARY TREE:

- Enqueue (Add): Menambahkan elemen ke dalam pohon biner dengan cara yang sesuai dengan aturan struktur Binary Tree, yaitu setiap node dapat memiliki maksimal dua anak.
- Dequeue (Remove): Menghapus elemen dari pohon biner. Proses penghapusan dapat melibatkan reorganisasi struktur pohon agar tetap memenuhi aturan Binary Tree.

IMPLEMENTASI BINARY TREE DI C++:

Binary Tree dapat diimplementasikan menggunakan struktur data yang terdiri dari node-node yang memiliki pointer ke anak kiri dan anak kanan. Setiap node menyimpan data dan menunjuk ke dua anaknya.

Link repl.it Source Code

https://onlinegdb.com/lbp7W5IE9

Source Code

```
1. //Aulia Putri Sayidina
2. //122140060
3.
4. #include <iostream>
5. #include <iomanip>
6. using namespace std;
7.
8. struct Tree{
9.   int info;
10.
11.  Tree *left;
12.  Tree *right;
13.  Tree *parent;
14.};
15.
16.Tree *Initialize(int value){
17.  Tree *newNode = new Tree;
18.
19.  newNode->info = value;
20.  newNode->left = nullptr;
21.  newNode->right = nullptr;
22.  newNode->parent = nullptr;
23.
24.  return newNode;
25.}
26.
27.bool IsTreeEmpty(Tree *P) { return P == nullptr; }
28.
29.void InsertLeft(Tree *P, int value){
30.  Tree *newNode = Initialize(value);
31.
32.  if (P->left == nullptr){
33.    newNode->parent = P;
34.    P->left = newNode;
```

```
35.     cout << "Success insert left node with value " << value << " with " << P->info
   << " as parrent\n";
36.   }else{
37.     cout << "Tree left was declared!";
38.   }
39.}
40.
41.void InsertRight(Tree *P, int value){
42.   Tree *newNode = Initialize(value);
43.   if (P->right == nullptr){
44.     newNode->parent = P;
45.     P->right = newNode;
46.     cout << "Success insert right node with value " << value << " with " << P->info
   << " as parrent\n";
47.   }
48.   else{
49.     cout << "Tree right was declared!";
50.   }
51.}
52.
53.void PreOrder(Tree *P){
54.   if (IsTreeEmpty(P)){
55.   }else{
56.     cout << P->info << " ";
57.     PreOrder(P->left);
58.     PreOrder(P->right);
59.   }
60.}
61.
62.void InOrder(Tree *P){
63.   if (IsTreeEmpty(P)){
64.   }else{
65.     InOrder(P->left);
66.     cout << P->info << " ";
67.     InOrder(P->right);
68.   }
69.}
70.
71.void PostOrder(Tree *P){
72.   if (IsTreeEmpty(P)){
73.   }else{
74.     PostOrder(P->left);
75.     PostOrder(P->right);
76.     cout << P->info << " ";
77.   }
```

```
78.}
79.
80.void display(Tree *P){
81.  cout << "<>Pre Order : ";
82.  PreOrder(P);
83.  cout << endl;
84.  cout << "<>In Order : ";
85.  InOrder(P);
86.  cout << endl;
87.  cout << "<>Post Order : ";
88.  PostOrder(P);
89.  cout << endl << endl;
90.}
91.
92.void displayIndent(Tree *P, int level=0){
93.  if (P != nullptr){
94.    displayIndent(P->right, level + 1);
95.    cout << setw(level * 3) << " " << P->info << endl;
96.    displayIndent(P->left, level + 1);
97.  }
98.}
99.
100.     void deleteRightLeaf(Tree *P){
101.       if (P->right == nullptr && P->left == nullptr){
102.         if (P->parent->right != nullptr){
103.           P->parent->right = nullptr;
104.           return;
105.         }
106.
107.         if (P->parent->left != nullptr){
108.           P->parent->left = nullptr;
109.         }
110.       }
111.
112.       if (P->right != nullptr && P->left == nullptr){
113.         deleteRightLeaf(P->right);
114.       }
115.       else if (P->right == nullptr && P->left != nullptr){
116.         deleteRightLeaf(P->left);
117.       }
118.       else if (P->right != nullptr && P->left != nullptr){
119.         deleteRightLeaf(P->right);
120.       }
121.     }
122.
```

```
123.    int main(){
124.      Tree *P = Initialize(1);
125.      InsertLeft(P, 2);
126.      InsertRight(P, 3);
127.
128.      InsertLeft(P->left, 4);
129.      InsertRight(P->left, 5);
130.
131.      InsertLeft(P->right, 6);
132.      InsertRight(P->right, 7);
133.
134.      InsertLeft(P->left->left, 8);
135.
136.      cout << endl << "Display tree : " << endl;
137.      display(P);
138.
139.      cout << "Display tree with indent : " << endl;
140.      displayIndent(P);
141.
142.      cout << "Processing delete tree...." << endl;
143.      while (!IsTreeEmpty(P)){
144.        deleteRightLeaf(P);
145.        displayIndent(P);
146.        display(P);
147.      }
148.
149.      return 0;
150.    }
```

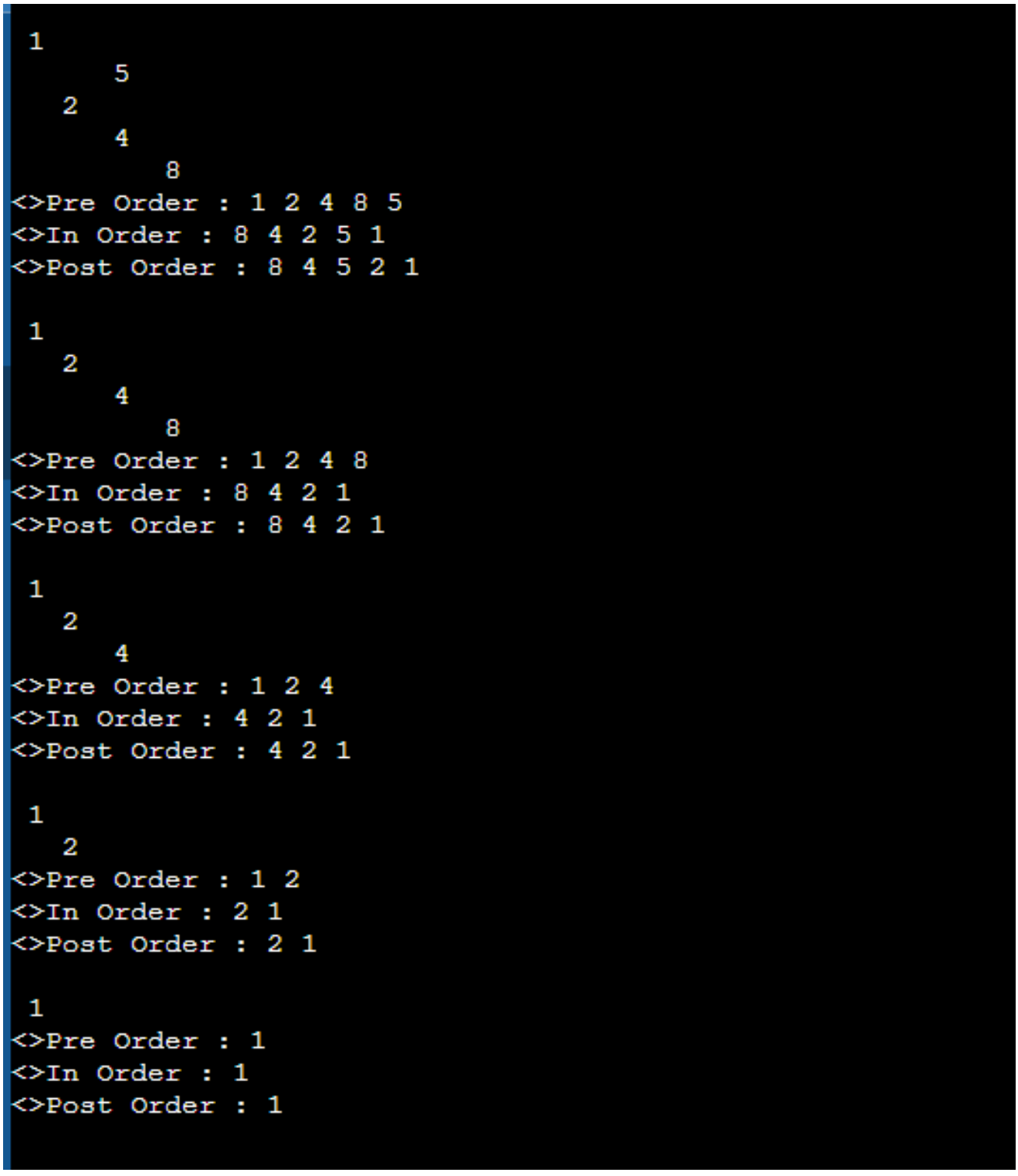
Dokumentasi Hasil Running

```
Success insert left node with value 2 with 1 as parrent
Success insert right node with value 3 with 1 as parrent
Success insert left node with value 4 with 2 as parrent
Success insert right node with value 5 with 2 as parrent
Success insert left node with value 6 with 3 as parrent
Success insert right node with value 7 with 3 as parrent
Success insert left node with value 8 with 4 as parrent

Display tree :
<>Pre Order : 1 2 4 8 5 3 6 7
<>In Order : 8 4 2 5 1 6 3 7
<>Post Order : 8 4 5 2 6 7 3 1

Display tree with indent :
      7
   3
      6
 1
      5
   2
      4
         8
Processing delete tree....
   3
      6
 1
      5
   2
      4
         8
<>Pre Order : 1 2 4 8 5 3 6
<>In Order : 8 4 2 5 1 6 3
<>Post Order : 8 4 5 2 6 3 1

   3
 1
      5
   2
      4
         8
<>Pre Order : 1 2 4 8 5 3
<>In Order : 8 4 2 5 1 3
<>Post Order : 8 4 5 2 3 1
```



Gambar Hasil Running Code Tugas Binary Tree

Referensi

Modul perkuliahan Algoritma Struktur Data – Binary Tree dari ITERA