

# **LAPORAN PRAKTIKUM STRUKTUR DATA**

## **PERTEMUAN 10**

### **MULTI LINKED LIST**



**Nama :**

Reyner Atira Prasetyo (2311104057)

S1SE-07-02

**Dosen :**

Wahyu Andi Saputra, S.Pd., M.Eng.

**PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## **I. TUJUAN**

- a. Memahami konsep penggunaan fungsi rekursif.
- b. Mengimplementasikan bentuk-bentuk fungsi rekursif.
- c. Mengaplikasikan struktur data tree dalam sebuah kasus pemrograman.
- d. Mengimplementasikan struktur data tree, khususnya Binary Tree.
- e. Mengimplementasikan struktur data tree, khususnya Binary Tree.

## **TOOL**

1. Visual Studio Code
2. GCC

## **II. DASAR TEORI**

Multi List merupakan sekumpulan list yang berbeda yang memiliki suatu keterhubungan satu sama lain. Tiap elemen dalam multi link list dapat membentuk list sendiri. Biasanya ada yang bersifat sebagai list induk dan list anak.

## **III. GUIDED**

1. GuidedGraph.cpp

```

1 #include <iostream>
2 #include <queue>
3
4 using namespace std;
5
6 struct LinkNode;
7
8 struct FirstEdge {
9     FirstNode *Node;
10    FirstEdge *Next;
11 };
12
13 struct LinkNode {
14     char info;
15     bool visited;
16     FirstEdge *firstEdge;
17     LinkNode *Next;
18 };
19
20 struct Graph {
21     LinkNode *first;
22 };
23
24 void CreateGraph(Graph &G) {
25     G.first = NULL;
26 }
27
28 void InsertNode(Graph &G, char X) {
29     LinkNode *newNode = new LinkNode;
30     newNode->info = X;
31     newNode->visited = false;
32     newNode->firstEdge = NULL;
33     newNode->Next = NULL;
34
35     if (G.first == NULL) {
36         G.first = newNode;
37     } else {
38         LinkNode *temp = G.first;
39         while (temp->Next != NULL) {
40             temp = temp->Next;
41         }
42         temp->Next = newNode;
43     }
44 }
45
46 void ConnectNode(LinkNode *N1, LinkNode *N2) {
47     LinkNode *newEdge = new LinkNode;
48     newEdge->Node = N2;
49     newEdge->Next = N1->firstEdge;
50     N1->firstEdge = newEdge;
51 }
52
53 void PrintInfoGraph(Graph G) {
54     LinkNode *temp = G.first;
55     while (temp != NULL) {
56         cout << temp->info << " ";
57         temp = temp->Next;
58     }
59     cout << endl;
60 }
61
62 void ResetVisited(Graph &G) {
63     LinkNode *temp = G.first;
64     while (temp != NULL) {
65         temp->visited = false;
66         temp = temp->Next;
67     }
68 }
69
70 void PrintDFS(Graph G, LinkNode *N) {
71     if (N == NULL) {
72         return;
73     }
74     N->visited = true;
75     cout << N->info << " ";
76     LinkNode *edge = N->firstEdge;
77     while (edge != NULL) {
78         if (!edge->Node->visited) {
79             PrintDFS(G, edge->Node);
80         }
81         edge = edge->Next;
82     }
83 }
84
85 void PrintBFS(Graph G, LinkNode *N) {
86     queue<LinkNode> q;
87     q.push(N);
88     N->visited = true;
89
90     while (!q.empty()) {
91         LinkNode *current = q.front();
92         q.pop();
93         cout << current->info << " ";
94
95         LinkNode *edge = current->firstEdge;
96         while (edge != NULL) {
97             if (!edge->Node->visited) {
98                 edge->Node->visited = true;
99                 q.push(edge->Node);
100             }
101             edge = edge->Next;
102         }
103     }
104 }
105
106 int main() {
107     Graph G;
108     CreateGraph(G);
109
110     InsertNode(G, 'A');
111     InsertNode(G, 'B');
112     InsertNode(G, 'C');
113     InsertNode(G, 'D');
114     InsertNode(G, 'E');
115     InsertNode(G, 'F');
116     InsertNode(G, 'G');
117     InsertNode(G, 'H');
118
119     LinkNode *A = G.first;
120     LinkNode *B = A->Next;
121     LinkNode *C = B->Next;
122     LinkNode *D = C->Next;
123     LinkNode *E = D->Next;
124     LinkNode *F = E->Next;
125     LinkNode *G = F->Next;
126     LinkNode *H = G->Next;
127
128     ConnectNode(A, B);
129     ConnectNode(A, C);
130     ConnectNode(A, D);
131     ConnectNode(B, C);
132     ConnectNode(C, F);
133     ConnectNode(C, G);
134     ConnectNode(D, H);
135
136     cout << "DFS Traversal: ";
137     // ResetVisited(G);
138     PrintDFS(G, A);
139     cout << endl;
140
141     cout << "BFS Traversal: ";
142     ResetVisited(G);
143     PrintBFS(G, A);
144     cout << endl;
145
146     return 0;
147 }

```

Hasil Run :

```
SDebugLauncher.exe --stdin=Microsof
r-ygqqptbh.1fp' '--pid=Microsoft-M
DFS traversal: A C G F B E D H
BFS traversal: A C B G F E D H
PS D:\PRAKTIKUM\Struktur Data\pert
```

#### **IV. UNGUIDED**

1. Unguided1.cpp

```

1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <iomanip>
5  using namespace std;
6
7  int main() {
8      int jumlahSimpul;
9      cout << "Silakan masukan jumlah simpul: ";
10     cin >> jumlahSimpul;
11
12     vector<string> namaSimpul(jumlahSimpul);
13     for (int i = 0; i < jumlahSimpul; i++) {
14         cout << "Simpul " << i + 1 << " : ";
15         cin >> namaSimpul[i];
16     }
17
18     // Membuat adjacency matrix
19     vector<vector<int>> graph(jumlahSimpul, vector<int>(jumlahSimpul, 0));
20     cout << "\nSilakan masukkan bobot antar simpul\n";
21     for (int i = 0; i < jumlahSimpul; i++) {
22         for (int j = 0; j < jumlahSimpul; j++) {
23             if (i == j) {
24                 graph[i][j] = 0; // Bobot dari simpul ke dirinya sendiri 0
25             } else {
26                 cout << namaSimpul[i] << "--> " << namaSimpul[j] << " = ";
27                 cin >> graph[i][j];
28             }
29         }
30     }
31
32     // Menampilkan adjacency matrix
33     cout << "\nAdjacency Matrix:\n";
34     cout << setw(10) << " ";
35     for (const string& nama : namaSimpul) {
36         cout << setw(10) << nama;
37     }
38     cout << endl;
39
40     for (int i = 0; i < jumlahSimpul; i++) {
41         cout << setw(10) << namaSimpul[i];
42         for (int j = 0; j < jumlahSimpul; j++) {
43             cout << setw(10) << graph[i][j];
44         }
45         cout << endl;
46     }
47
48     return 0;
49 }
50

```

Penjelasan :

- Variabel jumlahSimpul menerima input jumlah simpul dari pengguna.
- vector<string>: Digunakan untuk menyimpan nama dari setiap simpul (misalnya: nama kota).

- Loop akan meminta pengguna untuk memasukkan nama setiap simpul sebanyak jumlahSimpul kali.
- `vector<vector<int>>`: Matriks dua dimensi untuk menyimpan bobot (jarak) antar simpul.
- Inisialisasi matriks dengan ukuran jumlahSimpul x jumlahSimpul dan diisi dengan 0.
- `graph[i][j]`: Menyimpan bobot (jarak) dari simpul i ke simpul j.
- Jika `i == j` (sama), maka bobot otomatis 0 (tidak ada jarak ke diri sendiri).
- Jika tidak, pengguna diminta memasukkan bobot untuk koneksi antar simpul.

```

r-ajs4vaif.so2' '--pid=Microsoft-MIEngine-P
Silakan masukan jumlah simpul: 2
Simpul 1 : BALI
Simpul 2 : PALU

Silakan masukan bobot antar simpul
BALI--> PALU = 3
PALU--> BALI = 4

Adjacency Matrix:
          BALI    PALU
BALI      0      3
PALU      4      0
PS D:\PRAKTIKUM\Struktur Data\pertemuan11>

```

## 2. Unguided2.cpp

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main() {
6      int jumlahSimpul, jumlahSisi;
7
8      // Input jumlah simpul dan sisi
9      cout << "Masukkan jumlah simpul: ";
10     cin >> jumlahSimpul;
11     cout << "Masukkan jumlah sisi: ";
12     cin >> jumlahSisi;
13
14     // Inisialisasi adjacency matrix dengan nol
15     vector<vector<int>> adjacencyMatrix(jumlahSimpul, vector<int>(jumlahSimpul, 0));
16
17     // Input pasangan simpul yang terhubung oleh sisi
18     cout << "Masukkan pasangan simpul:\n";
19     for (int i = 0; i < jumlahSisi; i++) {
20         int simpul1, simpul2;
21         cin >> simpul1 >> simpul2;
22
23         // Karena graf tidak berarah, update kedua arah
24         adjacencyMatrix[simpul1 - 1][simpul2 - 1] = 1;
25         adjacencyMatrix[simpul2 - 1][simpul1 - 1] = 1;
26     }
27
28     // Menampilkan adjacency matrix
29     cout << "\nAdjacency Matrix:\n";
30     for (int i = 0; i < jumlahSimpul; i++) {
31         for (int j = 0; j < jumlahSimpul; j++) {
32             cout << adjacencyMatrix[i][j] << " ";
33         }
34         cout << endl;
35     }
36
37     return 0;
38 }

```

Penjelasan :

- Input jumlah simpul dan sisi menggunakan cin : Meminta jumlah simpul (jumlahSimpul) dan sisi (jumlahSisi) dari pengguna.
- Matriks 2D (adjacencyMatrix) berukuran jumlahSimpul x jumlahSimpul diisi dengan nol untuk merepresentasikan tidak adanya koneksi awal antar simpul.
- Untuk setiap sisi, pengguna memasukkan pasangan simpul yang terhubung. Matriks diperbarui di kedua arah karena graf tidak berarah:

adjacencyMatrix[simpul1 - 1][simpul2 - 1] = 1;

```
adjacencyMatrix[simpul2 - 1][simpul1 - 1] = 1;
```

- Matriks ditampilkan, menunjukkan koneksi antar simpul. Nilai 1 menandakan adanya koneksi, sedangkan 0 menandakan tidak ada koneksi.

```
Microsoft Windows [Version 6.0.6002.18005] Copyright (c) 2009 Microsoft Corporation  
C:\Users\user> cd C:\Users\user\Documents\Praktikum Struktur Data\pertemuan11  
C:\Users\user\Documents\Praktikum Struktur Data\pertemuan11> g++ graph.cpp -std=c++11 -o graph.exe  
C:\Users\user\Documents\Praktikum Struktur Data\pertemuan11> .\graph.exe  
Masukkan jumlah simpul: 4  
Masukkan jumlah sisi: 4  
Masukkan pasangan simpul:  
1 2  
1 3  
2 4  
3 4  
  
Adjacency Matrix:  
0 1 1 0  
1 0 0 1  
1 0 0 1  
0 1 1 0  
PS D:\PRAKTIKUM\Struktur Data\pertemuan11>
```

## V. KESIMPULAN

Praktikum struktur data graph memperkenalkan konsep dan implementasi graph sebagai struktur data yang merepresentasikan hubungan antar objek melalui simpul (nodes) dan sisi (edges). Graph dapat direpresentasikan dengan matriks adjacency atau list adjacency, dengan kelebihan masing-masing dalam efisiensi memori dan waktu.

Algoritma traversal seperti DFS dan BFS terbukti efektif dalam menyelesaikan berbagai masalah, termasuk pencarian jalur dan deteksi siklus. Selain itu, graph memiliki banyak aplikasi praktis, seperti pencarian rute, analisis jaringan, dan penjadwalan. Praktikum ini membantu memahami pentingnya struktur data graph dan algoritmanya dalam menyelesaikan masalah yang kompleks.