

LAPORAN PRAKTIKUM
Modul 13
“MULTI LINKED LIST”



Disusun Oleh:
Nama : Ganes Gemi Putra
NIM : 2311104075
Kelas : SE-07-02

Dosen : WAHYU ANDI SAPUTRA

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. Tujuan

Memahami penggunaan Multi Linked list.

Mengimplementasikan Multi Linked list dalam beberapa studi kasus.

2. Landasan Teori

Konsep Linked List

- **Linked List adalah struktur data linear yang terdiri dari elemen-elemen yang dihubungkan menggunakan pointer.**
- **Elemen dalam linked list terdiri dari dua bagian:**

- Info/data, yang menyimpan nilai.
- Pointer, yang menunjuk ke elemen berikutnya (atau sebelumnya dalam kasus doubly linked list).

Multi Linked List

- Multi Linked List adalah pengembangan dari linked list biasa, di mana setiap elemen dalam list dapat memiliki sub-list atau child list.
- Elemen induk (parent node) memiliki pointer tambahan yang menunjuk ke elemen-elemen dalam sub-list (child node).

Implementasi Dasar

- **Insert:** Menambahkan elemen baru ke dalam list, baik di list induk maupun sub-list.
- **Delete:** Menghapus elemen dari list, dengan memperhatikan keterhubungan antar elemen agar struktur list tetap konsisten.
- **Traversal:** Proses menelusuri elemen-elemen dalam list induk dan sub-list.

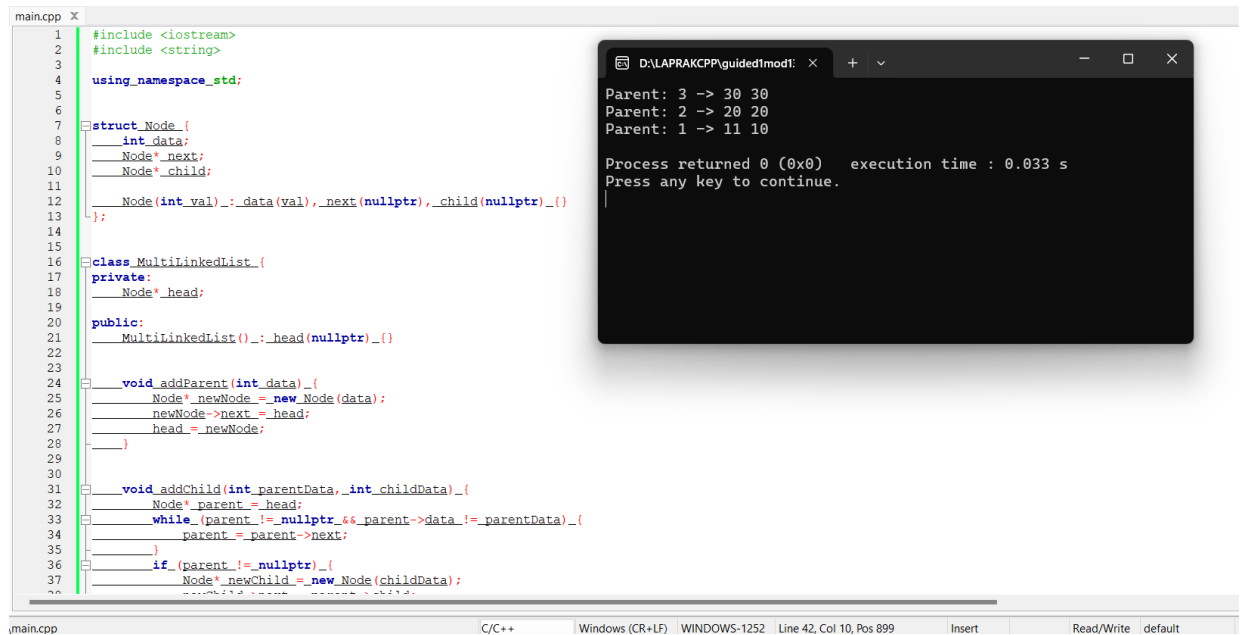
Operasi Umum

- **Insert pada List Anak (Child List):** Memerlukan informasi tentang elemen induk untuk menentukan lokasi penambahan.
- **Delete pada List Anak:** Menghapus elemen berdasarkan induknya dan memperbarui pointer.
- **Delete pada List Induk (Parent List):** Menghapus elemen induk secara otomatis menghapus semua elemen dalam list anak yang terkait.

5. Penerapan

- Digunakan untuk struktur data yang memiliki hubungan hirarkis, seperti:
 - Organisasi (Induk: Divisi, Anak: Karyawan).
 - Data relasional (Induk: Kategori, Anak: Produk).

3. Guided 1 :

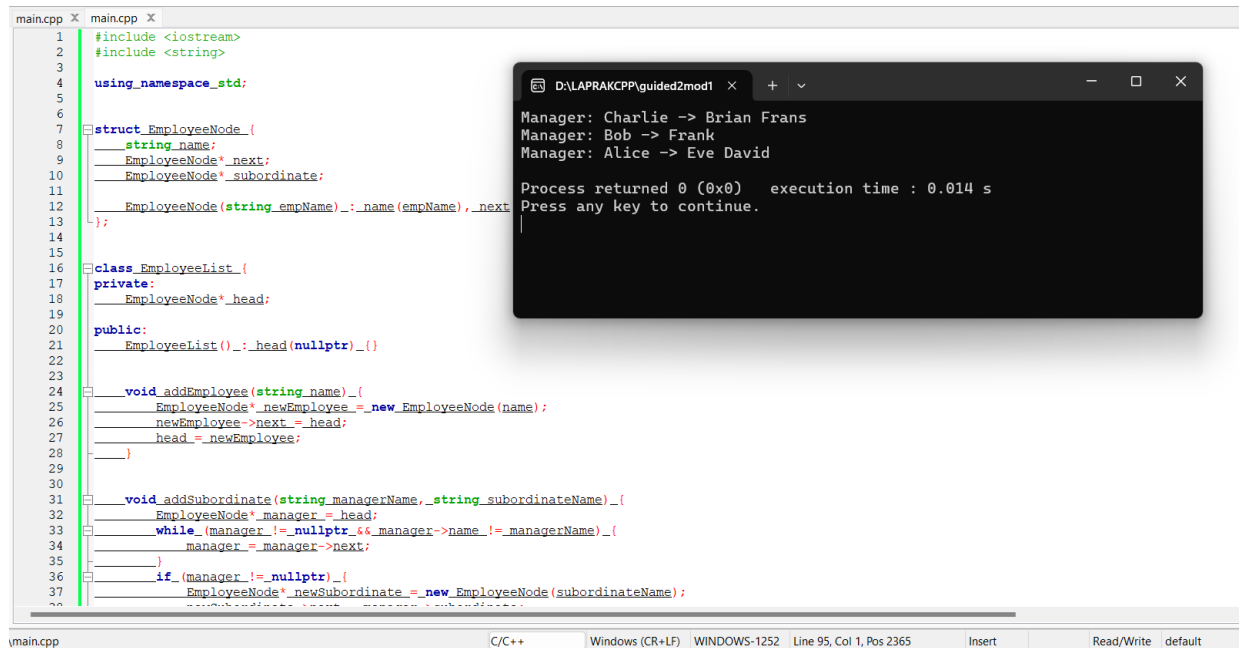


The screenshot shows a C++ IDE with a file named `main.cpp`. The code implements a `MultiLinkedList` class. The `Node` struct contains `int_data`, `Node* next`, and `Node* child`. The `MultiLinkedList` class has a `private` member `Node* head` and `public` methods `addParent` and `addChild`. The `addParent` method adds a new node to the head of the list. The `addChild` method adds a new child node to an existing parent node. The execution output shows the following:

```
Parent: 3 -> 30 30
Parent: 2 -> 20 20
Parent: 1 -> 11 10

Process returned 0 (0x0)   execution time : 0.033 s
Press any key to continue.
```

Guided 2 :



The screenshot shows a C++ IDE with a file named `main.cpp`. The code implements an `EmployeeList` class. The `EmployeeNode` struct contains `string name`, `EmployeeNode* next`, and `EmployeeNode* subordinate`. The `EmployeeList` class has a `private` member `EmployeeNode* head` and `public` methods `addEmployee` and `addSubordinate`. The `addEmployee` method adds a new employee node to the head of the list. The `addSubordinate` method adds a new subordinate node to an existing manager node. The execution output shows the following:

```
Manager: Charlie -> Brian Frans
Manager: Bob -> Frank
Manager: Alice -> Eve David

Process returned 0 (0x0)   execution time : 0.014 s
Press any key to continue.
```

‘Guided 3 :

```
main.cpp X main.cpp X main.cpp X
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 // Struktur untuk node karyawan
7 struct EmployeeNode {
8     string name; // Nama karyawan
9     EmployeeNode* next; // Pointer ke karyawan berikutnya
10    EmployeeNode* subordinate; // Pointer ke subordinate
11};
12
13 // Kelas untuk Multi-Linked List Karyawan
14 class EmployeeList {
15 private:
16     EmployeeNode* head; // Pointer ke kepala list
17
18 public:
19     EmployeeList() { head = nullptr; }
20
21     // Menambahkan karyawan (induk)
22     void addEmployee(string name) {
23         EmployeeNode* newEmployee = new EmployeeNode(name);
24         newEmployee->next = head; // Menyambungkan ke karyawan
25         head = newEmployee; // Memperbarui head
26     }
27
28     // Menambahkan subordinate ke karyawan tertentu
29     void addSubordinate(string managerName, string subordinateName) {
30         EmployeeNode* manager = head;
31         while (manager != nullptr && manager->name != managerName) {
32             manager = manager->next;
33         }
34         if (manager != nullptr) { // Jika manager ditemukan
35             EmployeeNode* newSubordinate = new EmployeeNode(subordinateName);
36             newSubordinate->next = manager->subordinate;
37             manager->subordinate = newSubordinate;
38         }
39     }
40
41     // Menampilkan daftar karyawan
42     void display() {
43         EmployeeNode* current = head;
44         while (current != nullptr) {
45             cout << "Manager: " << current->name << " -> ";
46             EmployeeNode* subordinate = current->subordinate;
47             while (subordinate != nullptr) {
48                 cout << subordinate->name << " ";
49                 subordinate = subordinate->next;
50             }
51             cout << endl;
52             current = current->next;
53         }
54     }
55
56     // Menghapus karyawan
57     void deleteEmployee(string name) {
58         EmployeeNode* current = head;
59         EmployeeNode* prev = nullptr;
60         while (current != nullptr) {
61             if (current->name == name) {
62                 if (prev == nullptr) {
63                     head = current->next;
64                 } else {
65                     prev->next = current->next;
66                 }
67                 delete current;
68                 return;
69             }
70             prev = current;
71             current = current->next;
72         }
73     }
74
75     // Menghapus subordinate
76     void deleteSubordinate(string managerName, string subordinateName) {
77         EmployeeNode* manager = head;
78         while (manager != nullptr) {
79             EmployeeNode* subordinate = manager->subordinate;
80             while (subordinate != nullptr) {
81                 if (subordinate->name == subordinateName) {
82                     manager->subordinate = subordinate->next;
83                     delete subordinate;
84                     return;
85                 }
86                 subordinate = subordinate->next;
87             }
88             manager = manager->next;
89         }
90     }
91
92     // Menampilkan daftar subordinate
93     void displaySubordinates() {
94         EmployeeNode* current = head;
95         while (current != nullptr) {
96             cout << "Manager: " << current->name << " -> ";
97             EmployeeNode* subordinate = current->subordinate;
98             while (subordinate != nullptr) {
99                 cout << subordinate->name << " ";
100                subordinate = subordinate->next;
101            }
102            cout << endl;
103            current = current->next;
104        }
105    }
106};
107
108 int main() {
109     EmployeeList list;
110     list.addEmployee("Charlie");
111     list.addEmployee("Bob");
112     list.addEmployee("Alice");
113     list.addSubordinate("Alice", "David");
114     list.deleteEmployee("Charlie");
115     list.deleteSubordinate("Alice", "David");
116     list.display();
117     list.displaySubordinates();
118     return 0;
119 }
```

```
D:\LAPRAKCPP\guided3mod1 X + -
Initial employee list:
Manager: Charlie ->
Manager: Bob -> Frank
Manager: Alice -> Eve David
Subordinate David deleted from Alice.
Employee Charlie deleted.
Updated employee list:
Manager: Bob -> Frank
Manager: Alice -> Eve
Process returned 0 (0x0) execution time : 0.074 s
Press any key to continue.
```

4. Unguided

```
main.cpp X main.cpp X main.cpp X main.cpp X
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 struct Node {
7     string nama;
8     string nim;
9     char jenisKelamin;
10    float ipk;
11    Node* next;
12};
13
14 struct List {
15     Node* first;
16};
17
18 // Inisialisasi list
19 void createList(List& L) {
20     L.first = nullptr;
21 }
22
23 // Membuat node baru
24 Node* createData(string nama, string nim, char jenisKelamin, float ipk) {
25     Node* newNode = new Node;
26     newNode->nama = nama;
27     newNode->nim = nim;
28     newNode->jenisKelamin = jenisKelamin;
29     newNode->ipk = ipk;
30     newNode->next = nullptr;
31     return newNode;
32 }
33
34 // Menambahkan node di awal
35 void insertFirst(List& L, Node* newNode) {
36     newNode->next = L.first;
37     L.first = newNode;
38 }
```

```
D:\LAPRAKCPP\UnguidedMOI X + -
Nama : Danu
NIM : 04
L/P : L
IPK : 3.4
Nama : Eli
NIM : 05
L/P : P
IPK : 3.4
Nama : Fahmi
NIM : 06
L/P : L
IPK : 3.45
Nama : Gita
NIM : 07
L/P : P
IPK : 3.75
Nama : Hilmi
NIM : 08
L/P : P
IPK : 3.3
Process returned 0 (0x0) execution time : 0.024 s
Press any key to continue.
```

Program ini menunjukkan implementasi dasar linked list untuk menyimpan dan mengelola data mahasiswa.

Struktur modular dan fungsi-fungsi terpisah mempermudah pembacaan dan pengelolaan kode.

Anda dapat menambahkan fitur lain seperti penghapusan data (delete) atau

pencarian berdasarkan atribut lain.

5. Kesimpulan

Multi Linked List adalah pengembangan struktur data Linked List, yang memungkinkan setiap elemen untuk memiliki sub-list (child node).

Operasi dasar seperti insert, delete, dan traversal telah diimplementasikan untuk list induk dan sub-list.

Multi Linked List berguna dalam menyimpan data yang memiliki hubungan hierarkis, seperti struktur organisasi atau data relasional.

Studi kasus dalam dokumen menunjukkan penggunaan dasar struktur ini, dengan modularitas dalam kode yang mempermudah pengelolaan.