

Aturan Praktikum Struktur Data

1. **Akun GitHub:** Setiap praktikan wajib memiliki akun GitHub yang aktif dan digunakan selama praktikum berlangsung.
2. **Invite Collaborator:** Setiap praktikan diwajibkan untuk menambahkan collaborator di setiap repository
 - a. Asisten Praktikum: AndiniNH
 - b. Asisten Praktikum: 4ldiputra
3. **Repository Praktikum:** Setiap praktikan diwajibkan untuk membuat satu repository di GitHub yang akan digunakan untuk seluruh tugas dan laporan praktikum. Repository ini harus diatur dengan rapi dan sesuai dengan instruksi yang akan diberikan di lampiran.
4. **Penamaan Folder:** Penamaan folder dalam repository akan dibahas secara rinci di lampiran. Praktikan wajib mengikuti aturan penamaan yang telah ditentukan.

Nomor	Pertemuan	Penamaan
1	Pengantalan Bahasa C++ Bagian Pertama	01_Pengenalan_CPP_Bagian_1
2	Pengenalan Bahasa C++ Bagian Kedua	02_Pengenalan_CPP_Bagian_2
3	Abstract Data Type	03_Abstract_Data_Type
4	Single Linked List Bagian Pertama	04_Single_Linked_List_Bagian_1
5	Single Linked List Bagian Kedua	05_Single_Linked_List_Bagian_2
6	Double Linked List Bagian Pertama	06_Double_Linked_List_Bagian_1
7	Stack	07_Stack
8	Queue	08_Queue
9	Assessment Bagian Pertama	09_Assessment_Bagian_1
10	Tree Bagian Pertama	10_Tree_Bagian_1
11	Tree Bagian Kedua	11_Tree_Bagian_2
12	Asistensi Tugas Besar	12_Asistensi_Tugas_Besar
13	Multi Linked List	13_Multi_Linked_List
14	Graph	14_Graph
15	Assessment Bagian Kedua	15_Assessment_Bagian_2
16	Tugas Besar	16_Tugas_Besar

5. Jam Praktikum:

- Jam masuk praktikum adalah **1 jam lebih lambat** dari jadwal yang tercantum. Sebagai contoh, jika jadwal praktikum adalah pukul 06.30 - 09.30, maka aturan praktikum akan diatur sebagai berikut:
 - **06.30 - 07.30:** Waktu ini digunakan untuk **Tugas Praktikum dan Laporan Praktikum** yang dilakukan di luar laboratorium.
 - **07.30 - 08.30:** Sesi ini mencakup **tutorial, diskusi, dan kasus problem-solving**. Kegiatan ini berlangsung di dalam laboratorium dengan alokasi waktu sebagai berikut:
 - **60 menit pertama:** Tugas terbimbing.
 - **60 menit kedua:** Tugas mandiri.

6. **Pengumpulan Tugas Pendahuluan:** Tugas Pendahuluan (TP) wajib dikumpulkan melalui GitHub sesuai dengan format berikut:

nama_repo/nama_pertemuan/TP_Pertemuan_Ke.md

Sebagai contoh:

STD_Yudha_Islalmi_Sulistya_XXXXXXXX/01_Running_Modul/TP_01.md

7. **Pengecekan Tugas Pendahuluan:** Pengumpulan laporan praktikum akan diperiksa **1 hari sebelum praktikum selanjutnya** dimulai. Pastikan tugas telah diunggah tepat waktu untuk menghindari sanksi.

**LAPORAN PRAKTIKUM
MODUL 13
MULTI LINKED LIST**



Disusun Oleh :

Zaenarif Putra 'Ainurdin – 2311104049

Kelas :

SE-07-02

Dosen :

Wahyu Andi Saputra, S.pd,M.Eng

**PROGRAM STUDI SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024**

I. TUJUAN

1. Memahami penggunaan Multi Linked list.
2. Mengimplementasikan Multi Linked list dalam beberapa studi kasus.

II. LANDASAN TEORI

Untuk representasi hierarkis, seperti organisasi data karyawan dan anak-anak datanya, modul "Multi Linked List" berguna karena konsep struktur data yang memungkinkan keterhubungan antara berbagai daftar (list) yang berbeda, di mana elemen dalam satu list dapat berfungsi sebagai induk dari elemen dalam list lain.. Operasi dasar dalam Multi Linked List termasuk penambahan (insert) dan penghapusan (delete) elemen pada list induk dan anak-anak, selain dari fungsi-fungsi lain. Untuk mengelola struktur dinamis ini secara efektif, implementasinya sering kali menggunakan pointer.

III. GUIDE

1. Guide 1

a. Syntax

```
#include <iostream>
#include <string>
using namespace std;
struct Node {
    int data;
    Node* next;
    Node* child;

    Node(int val) : data(val), next(nullptr), child(nullptr) {}
};
class MultiLinkedList {
private:
    Node* head;

public:
    MultiLinkedList() : head(nullptr) {}
    void addParent(int data) {
        Node* newNode = new Node(data);
        newNode->next = head;
        head = newNode;
    }
    void addChild(int parentData, int childData) {
        Node* parent = head;
        while (parent != nullptr && parent->data != parentData) {
            parent = parent->next;
        }
        if (parent != nullptr) {
```

```
Node* newChild = new Node(childData);
newChild->next = parent->child;
parent->child = newChild;
} else {
    cout << "Parent not found!" << endl;
}
}

void display() {
    Node* current = head;
    while (current != nullptr) {
        cout << "Parent: " << current->data << " -> ";
        Node* child = current->child;
        while (child != nullptr) {
            cout << child->data << " ";
            child = child->next;
        }
        cout << endl;
        current = current->next;
    }
}

~MultiLinkedList() {

    while (head != nullptr) {
        Node* temp = head;
        head = head->next;

        while (temp->child != nullptr) {
            Node* childTemp = temp->child;
            temp->child = temp->child->next;
            delete childTemp;
        }
        delete temp;
    }
}

};

int main() {
    MultiLinkedList mList;

    mList.addParent(1);
    mList.addParent(2);
    mList.addParent(3);

    mList.addChild(1, 10);
    mList.addChild(1, 11);
    mList.addChild(2, 20);
    mList.addChild(2, 20);
    mList.addChild(3, 30);
    mList.addChild(3, 30);
    mList.display();
}
```

```
return 0;  
}
```

b. Penjelasan syntax

- Inklusi Library: Untuk memulai program, dua library diimpor: `<iostream>` untuk input dan output dan `<string>` untuk penggunaan string, walaupun string tidak digunakan dalam kode ini.
- Definisi Struktur Node: Dalam program, struktur Node digunakan untuk menampilkan elemen dalam daftar. Setiap node memiliki tiga atribut: data, yang menyimpan nilai integer, next, yang menunjuk ke node berikutnya dalam daftar, dan child.
- Konstruktor Node: Nilai data dapat diinisialisasi dengan val menggunakan konstruktor `node(int val)`. Konstruktor ini juga mengatur next dan child menjadi nullptr, yang berarti pada awalnya tidak ada node atau anak berikutnya.
- Kelas MultiLinkedList: Kelas MultiLinkedList dirancang untuk mengelola daftar dengan struktur bertingkat banyak. Kelas ini memiliki satu atribut khusus, head, yang menunjuk ke node pertama daftar.
- Konstruktor MultiLinkedList: Konstruktor `MultiLinkedList()` menginisialisasi head menjadi nullptr, yang berarti daftar awalnya kosong.
- Menambahkan Parent: Untuk menambahkan node baru sebagai parent, metode `addParent(int data)` digunakan. Nilai data digunakan untuk membuat node baru, dan kemudian ditambahkan ke daftar dengan mengatur next dari node baru ke head yang lama, dan memperbarui head ke node baru.
- Menambahkan Child: Metode `addChild(int parentData, int childData)` mencari parent berdasarkan parentData. Jika parent ditemukan, node anak baru dibuat dengan childData dan ditambahkan ke daftar anak dari parent. Jika parent tidak ditemukan, program akan mencetak pesan "Parent not found!".
- Menampilkan Daftar: Metode `display()` digunakan untuk menampilkan semua parent dan anak-anak mereka. Program ini akan mencetak nilai dari setiap parent diikuti oleh nilai anak-anaknya.
- Destruktor MultiLinkedList: Destruktor `~MultiLinkedList()` bertanggung jawab untuk membersihkan memori yang digunakan oleh daftar. Ia menghapus semua node parent dan anak dengan cara mengiterasi melalui daftar dan menghapus setiap node serta anak-anaknya.
- Fungsi Main: Di dalam fungsi `main()`, objek MultiLinkedList dibuat, dan beberapa parent serta child ditambahkan. Setelah itu, daftar ditampilkan dengan memanggil metode `display()`.

c. Output

```
Parent: 3 -> 30 30
Parent: 2 -> 20 20
Parent: 1 -> 11 10
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan10\guide\output>
```

2. Guide 2

a. Syntax

```
#include <iostream>
#include <string>

using namespace std;
struct EmployeeNode {
    string name;
    EmployeeNode* next;
    EmployeeNode* subordinate;

    EmployeeNode(string empName) : name(empName), next(nullptr),
subordinate(nullptr) {}
};

class EmployeeList {
private:
    EmployeeNode* head;

public:
    EmployeeList() : head(nullptr) {}

    void addEmployee(string name) {
        EmployeeNode* newEmployee = new EmployeeNode(name);
        newEmployee->next = head;
        head = newEmployee;
    }

    void addSubordinate(string managerName, string subordinateName) {
        EmployeeNode* manager = head;
        while (manager != nullptr && manager->name != managerName) {
            manager = manager->next;
        }
        if (manager != nullptr) {
            EmployeeNode* newSubordinate = new
EmployeeNode(subordinateName);
            newSubordinate->next = manager->subordinate;
            manager->subordinate = newSubordinate;
        } else {
```

```
        cout << "Manager not found!" << endl;
    }
}

void display() {
    EmployeeNode* current = head;
    while (current != nullptr) {
        cout << "Manager: " << current->name << " -> ";
        EmployeeNode* sub = current->subordinate;
        while (sub != nullptr) {
            cout << sub->name << " ";
            sub = sub->next;
        }
        cout << endl;
        current = current->next;
    }
}

~EmployeeList() {

    while (head != nullptr) {
        EmployeeNode* temp = head;
        head = head->next;

        while (temp->subordinate != nullptr) {
            EmployeeNode* subTemp = temp->subordinate;
            temp->subordinate = temp->subordinate->next;
            delete subTemp;
        }
        delete temp;
    }
}

};

int main() {
    EmployeeList empList;

    empList.addEmployee("Alice");
    empList.addEmployee("Bob");
    empList.addEmployee("Charlie");

    empList.addSubordinate("Alice", "David");
    empList.addSubordinate("Alice", "Eve");
    empList.addSubordinate("Bob", "Frank");

    empList.addSubordinate("Charlie", "Frans");
    empList.addSubordinate("Charlie", "Brian");

    empList.display();
    return 0;
}
```


b. Penjelasan Syntax

- Inklusi Library: Program ini menggunakan dua library, yaitu `<iostream>` untuk melakukan input dan output, serta `<string>` untuk memanfaatkan tipe data string yang menyimpan teks.
- Definisi Struktur `EmployeeNode`: Struktur `EmployeeNode` didefinisikan untuk merepresentasikan setiap karyawan. Setiap node memiliki tiga atribut: `name`: menyimpan nama karyawan. `next`: pointer yang menunjuk ke karyawan berikutnya dalam daftar. `subordinate`: pointer yang menunjuk ke bawahan dari karyawan tersebut. Konstruktor `EmployeeNode`: Konstruktor `EmployeeNode(string empName)` digunakan untuk menginisialisasi nama karyawan dengan `empName`, dan mengatur `next` serta `subordinate` menjadi `nullptr`, yang berarti tidak ada karyawan berikutnya atau bawahan pada awalnya.
- Kelas `EmployeeList`: Kelas `EmployeeList` didefinisikan untuk mengelola daftar karyawan yang memiliki struktur multi-level. Kelas ini memiliki satu atribut privat, yaitu `head`, yang menunjuk ke karyawan pertama dalam daftar.
- Konstruktor `EmployeeList`: Konstruktor `EmployeeList()` menginisialisasi `head` menjadi `nullptr`, menandakan bahwa daftar karyawan awalnya kosong.
- Menambahkan Karyawan: Metode `addEmployee(string name)` digunakan untuk menambahkan karyawan baru ke dalam daftar. Node baru dibuat dengan nama `name`, dan kemudian ditambahkan di depan daftar dengan mengatur `next` dari node baru ke `head` yang lama, dan memperbarui `head` ke node baru.
- Menambahkan Bawahan: Metode `addSubordinate(string managerName, string subordinateName)` mencari karyawan yang berperan sebagai manajer berdasarkan `managerName`. Jika manajer ditemukan, node bawahan baru dibuat dengan `subordinateName`, dan ditambahkan ke daftar bawahan dari manajer tersebut. Jika manajer tidak ditemukan, program akan mencetak pesan "Manager not found!".
- Menampilkan Daftar Karyawan: Metode `display()` digunakan untuk menampilkan semua manajer dan bawahan mereka. Program ini akan mencetak nama setiap manajer diikuti oleh nama-nama bawahan mereka.
- Destruktor `EmployeeList`: Destruktor `~EmployeeList()` bertanggung jawab untuk membersihkan memori yang digunakan oleh daftar karyawan. Ia menghapus semua node karyawan dan bawahan dengan cara mengiterasi melalui daftar dan menghapus setiap node serta bawahan mereka.
- Fungsi Main: Di dalam fungsi `main()`, objek `EmployeeList` dibuat, dan beberapa karyawan ditambahkan. Kemudian, beberapa bawahan ditambahkan untuk masing-masing manajer. Setelah itu, daftar karyawan ditampilkan dengan memanggil metode `display()`.

c. Output

```
Manager: Charlie -> Brian Frans
Manager: Bob -> Frank
Manager: Alice -> Eve David
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan10\guide\output>
```

3. Guide 3

a. Syntax

```
#include <iostream>
#include <string>

using namespace std;
struct EmployeeNode {
    string name;
    EmployeeNode* next;
    EmployeeNode* subordinate;

    EmployeeNode(string empName) : name(empName), next(nullptr),
subordinate(nullptr) {}
};

class EmployeeList {
private:
    EmployeeNode* head;

public:
    EmployeeList() : head(nullptr) {}

    void addEmployee(string name) {
        EmployeeNode* newEmployee = new EmployeeNode(name);
        newEmployee->next = head;
        head = newEmployee;
    }

    void addSubordinate(string managerName, string subordinateName) {
        EmployeeNode* manager = head;
        while (manager != nullptr && manager->name != managerName) {
            manager = manager->next;
        }
        if (manager != nullptr) {
            EmployeeNode* newSubordinate = new
EmployeeNode(subordinateName);
            newSubordinate->next = manager->subordinate;
            manager->subordinate = newSubordinate;
        } else {
            cout << "Manager not found!" << endl;
        }
    }

    void deleteEmployee(string name) {
```

```
EmployeeNode** current = &head;
while (*current != nullptr && (*current)->name != name) {
    current = &((*current)->next);
}

if (*current != nullptr) {
    EmployeeNode* toDelete = *current;
    *current = (*current)->next;

    while (toDelete->subordinate != nullptr) {
        EmployeeNode* subTemp = toDelete->subordinate;
        toDelete->subordinate = toDelete->subordinate->next;
        delete subTemp;
    }
    delete toDelete;
    cout << "Employee " << name << " deleted." << endl;
} else {
    cout << "Employee not found!" << endl;
}
}

void deleteSubordinate(string managerName, string subordinateName)
{
    EmployeeNode* manager = head;
    while (manager != nullptr && manager->name != managerName) {
        manager = manager->next;
    }

    if (manager != nullptr) {
        EmployeeNode** currentSub = &(manager->subordinate);
        while (*currentSub != nullptr && (*currentSub)->name !=
subordinateName) {
            currentSub = &((*currentSub)->next);
        }

        if (*currentSub != nullptr) {
            EmployeeNode* toDelete = *currentSub;
            *currentSub = (*currentSub)->next;

            delete toDelete;
            cout << "Subordinate " << subordinateName << " deleted from
" << managerName << "." << endl;
        } else {
            cout << "Subordinate not found!" << endl;
        }
    } else {
        cout << "Manager not found!" << endl;
    }
}
```

```
void display() {
    EmployeeNode* current = head;
    while (current != nullptr) {
        cout << "Manager: " << current->name << " -> ";
        EmployeeNode* sub = current->subordinate;
        while (sub != nullptr) {
            cout << sub->name << " ";
            sub = sub->next;
        }
        cout << endl;
        current = current->next;
    }
}

~EmployeeList() {
    while (head != nullptr) {
        EmployeeNode* temp = head;
        head = head->next;

        while (temp->subordinate != nullptr) {
            EmployeeNode* subTemp = temp->subordinate;
            temp->subordinate = temp->subordinate->next;
            delete subTemp;
        }
        delete temp;
    }
}

};

int main() {
    EmployeeList empList;

    empList.addEmployee("Alice");
    empList.addEmployee("Bob");
    empList.addEmployee("Charlie");

    empList.addSubordinate("Alice", "David");
    empList.addSubordinate("Alice", "Eve");
    empList.addSubordinate("Bob", "Frank");

    cout << "Initial employee list:" << endl;
    empList.display();

    empList.deleteSubordinate("Alice", "David");
    empList.deleteEmployee("Charlie");

    cout << "\nUpdated employee list:" << endl;
    empList.display();

    return 0;
}
```

```
}
```

b. Penjelasan Syntax

- Inklusi Library: Program ini menggunakan dua library, yaitu `<iostream>` untuk melakukan input dan output, serta `<string>` untuk memanfaatkan tipe data string yang menyimpan teks.
- Definisi Struktur `EmployeeNode`: Struktur `EmployeeNode` didefinisikan untuk merepresentasikan setiap karyawan. Setiap node memiliki tiga atribut: `name`: menyimpan nama karyawan. `next`: pointer yang menunjuk ke karyawan berikutnya dalam daftar. `subordinate`: pointer yang menunjuk ke bawahan dari karyawan tersebut. Konstruktor `EmployeeNode`: Konstruktor `EmployeeNode(string empName)` digunakan untuk menginisialisasi nama karyawan dengan `empName`, dan mengatur `next` serta `subordinate` menjadi `nullptr`, yang berarti tidak ada karyawan berikutnya atau bawahan pada awalnya.
- Kelas `EmployeeList`: Kelas `EmployeeList` didefinisikan untuk mengelola daftar karyawan yang memiliki struktur multi-level. Kelas ini memiliki satu atribut privat, yaitu `head`, yang menunjuk ke karyawan pertama dalam daftar.
- Konstruktor `EmployeeList`: Konstruktor `EmployeeList()` menginisialisasi `head` menjadi `nullptr`, menandakan bahwa daftar karyawan awalnya kosong.
- Menambahkan Karyawan: Metode `addEmployee(string name)` digunakan untuk menambahkan karyawan baru ke dalam daftar. Node baru dibuat dengan nama `name`, dan kemudian ditambahkan di depan daftar dengan mengatur `next` dari node baru ke `head` yang lama, dan memperbarui `head` ke node baru.
- Menambahkan Bawahan: Metode `addSubordinate(string managerName, string subordinateName)` mencari karyawan yang berperan sebagai manajer berdasarkan `managerName`. Jika manajer ditemukan, node bawahan baru dibuat dengan `subordinateName`, dan ditambahkan ke daftar bawahan dari manajer tersebut. Jika manajer tidak ditemukan, program akan mencetak pesan "Manager not found!".
- Menghapus Karyawan: Metode `deleteEmployee(string name)` digunakan untuk menghapus karyawan dari daftar. Program mencari karyawan berdasarkan `name`, dan jika ditemukan, menghapus karyawan tersebut beserta semua bawahan yang dimilikinya. Jika karyawan tidak ditemukan, program akan mencetak pesan "Employee not found!".
- Menghapus Bawahan: Metode `deleteSubordinate(string managerName, string subordinateName)` mencari manajer berdasarkan `managerName`. Jika manajer ditemukan, program mencari bawahan berdasarkan `subordinateName`. Jika bawahan ditemukan, bawahan tersebut dihapus dari daftar. Jika tidak ditemukan, program akan mencetak pesan "Subordinate not found!". Jika manajer tidak ditemukan, program akan mencetak pesan "Manager not found!".
- Menampilkan Daftar Karyawan: Metode `display()` digunakan untuk menampilkan semua manajer dan bawahan mereka. Program ini akan mencetak nama setiap manajer diikuti oleh nama-nama bawahan mereka.

- Destruktor EmployeeList: Destruktor ~EmployeeList() bertanggung jawab untuk membersihkan memori yang digunakan oleh daftar karyawan. Ia menghapus semua node karyawan dan bawahan dengan cara mengiterasi melalui daftar dan menghapus setiap node serta bawahan mereka.
- Fungsi Main: Di dalam fungsi main(), objek EmployeeList dibuat, dan beberapa karyawan ditambahkan. Kemudian, beberapa bawahan ditambahkan untuk masing-masing manajer. Setelah itu, daftar karyawan ditampilkan dengan memanggil metode display(). Program kemudian menghapus bawahan "David" dari manajer "Alice" dan menghapus karyawan "Charlie". Setelah penghapusan, daftar karyawan yang diperbarui ditampilkan.

c. Output

```
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
PS C:\Users\LENOVO\OneDrive - Telkom University\Documents\ALL Matkul\StrukturData\pertemuan10\guide\output>
```

IV. UNGUIDED

1. Unguided 1

a. Syntax :

```
// Program Management Data Pegawai dan Proyek

#include <iostream>
#include <string>
using namespace std;

struct Proyek {
    string namaProyek;
    int durasi;
    Proyek* nextProyek;
};

struct Pegawai {
    string namaPegawai;
    string idPegawai;
    Proyek* headProyek;
    Pegawai* nextPegawai;
};

Pegawai* headPegawai = nullptr;
```

```
void tambahPegawai(string nama, string id) {
    Pegawai* pegawaiBaru = new Pegawai;
    pegawaiBaru->namaPegawai = nama;
    pegawaiBaru->idPegawai = id;
    pegawaiBaru->headProyek = nullptr;
    pegawaiBaru->nextPegawai = headPegawai;
    headPegawai = pegawaiBaru;
}

void tambahProyek(string idPegawai, string namaProyek, int durasi) {
    Pegawai* currPegawai = headPegawai;
    while (currPegawai != nullptr) {
        if (currPegawai->idPegawai == idPegawai) {
            Proyek* proyekBaru = new Proyek;
            proyekBaru->namaProyek = namaProyek;
            proyekBaru->durasi = durasi;
            proyekBaru->nextProyek = currPegawai->headProyek;
            currPegawai->headProyek = proyekBaru;
            return;
        }
        currPegawai = currPegawai->nextPegawai;
    }
    cout << "Pegawai dengan ID : " << idPegawai << " tidak dapat ditemukan!\n";
}

void hapusProyek(string idPegawai, string namaProyek) {
    Pegawai* currPegawai = headPegawai;
    while (currPegawai != nullptr) {
        if (currPegawai->idPegawai == idPegawai) {
            Proyek* currProyek = currPegawai->headProyek;
            Proyek* prevProyek = nullptr;
            while (currProyek != nullptr) {
                if (currProyek->namaProyek == namaProyek) {
                    if (prevProyek == nullptr) {
                        currPegawai->headProyek = currProyek->nextProyek;
                    } else {
                        prevProyek->nextProyek = currProyek->nextProyek;
                    }
                    delete currProyek;
                    return;
                }
                prevProyek = currProyek;
                currProyek = currProyek->nextProyek;
            }
        }
        currPegawai = currPegawai->nextPegawai;
    }
    cout << "Proyek : " << namaProyek << " tidak ditemukan untuk
```

```
pegawai dengan ID : " << idPegawai << "!\\n";
}

void tampilkanData() {
    Pegawai* currPegawai = headPegawai;
    while (currPegawai != nullptr) {
        cout << "Pegawai : " << currPegawai->namaPegawai << " (ID: " <<
currPegawai->idPegawai << ")\\n";
        Proyek* currProyek = currPegawai->headProyek;
        while (currProyek != nullptr) {
            cout << " Proyek : " << currProyek->namaProyek << " (Durasi :
" << currProyek->durasi << " bulan)\\n";
            currProyek = currProyek->nextProyek;
        }
        currPegawai = currPegawai->nextPegawai;
    }
}

int main() {
    int menu;
    do {
        cout << "\\nMenu:\\n";
        cout << "1. Tambah Nama Pegawai\\n";
        cout << "2. Tambah Nama Proyek\\n";
        cout << "3. Hapus Nama Proyek\\n";
        cout << "4. Tampilkan Data Pegawai & Proyek\\n";
        cout << "5. Keluar\\n";
        cout << "Menu: ";
        cin >> menu;
        cin.ignore();

        if (menu == 1) {
            string nama, id;
            cout << "Masukkan Nama Pegawai : ";
            getline(cin, nama);
            cout << "Masukkan ID Pegawai : ";
            getline(cin, id);
            tambahPegawai(nama, id);
        } else if (menu == 2) {
            string id, namaProyek;
            int durasi;
            cout << "Masukkan ID Pegawai : ";
            getline(cin, id);
            cout << "Masukkan Nama Proyek : ";
            getline(cin, namaProyek);
            cout << "Masukkan Durasi Proyek (bulan) : ";
            cin >> durasi;
            cin.ignore();
            tambahProyek(id, namaProyek, durasi);
        } else if (menu == 3) {
```



```
string id, namaProyek;
cout << "Masukkan ID Pegawai : ";
getline(cin, id);
cout << "Masukkan Nama Proyek yang akan dihapus : ";
getline(cin, namaProyek);
hapusProyek(id, namaProyek);
} else if (menu == 4) {
    tampilkanData();
} else if (menu != 5) {
    cout << "Pilihan tidak tersedia.\n";
}
} while (menu != 5);

return 0;
}
```

b. Penjelasan Syntax

- Inklusi Library: Program ini menggunakan library <iostream> untuk melakukan input dan output, serta <string> untuk memanfaatkan tipe data string yang menyimpan teks.
- Definisi Struktur Proyek: Struktur Proyek didefinisikan untuk merepresentasikan proyek yang dikelola oleh pegawai. Setiap proyek memiliki tiga atribut:
 - namaProyek: menyimpan nama proyek.
 - durasi: menyimpan durasi proyek dalam bulan.
 - nextProyek: pointer yang menunjuk ke proyek berikutnya dalam daftar proyek pegawai.
- Definisi Struktur Pegawai: Struktur Pegawai didefinisikan untuk merepresentasikan pegawai. Setiap pegawai memiliki empat atribut: namaPegawai: menyimpan nama pegawai. idPegawai: menyimpan ID pegawai. headProyek: pointer yang menunjuk ke proyek pertama yang dikelola oleh pegawai. nextPegawai: pointer yang menunjuk ke pegawai berikutnya dalam daftar pegawai. Variabel Global: Pegawai* headPegawai didefinisikan sebagai pointer global yang menunjuk ke pegawai pertama dalam daftar pegawai. Awalnya, ini diatur ke nullptr, menandakan bahwa tidak ada pegawai yang terdaftar.
- Fungsi tambahPegawai: Fungsi ini digunakan untuk menambahkan pegawai baru ke dalam daftar. Fungsi ini menerima dua parameter: nama dan id. Sebuah node pegawai baru dibuat, diisi dengan data yang diberikan, dan ditambahkan di depan daftar pegawai.
- Fungsi tambahProyek: Fungsi ini digunakan untuk menambahkan proyek baru ke pegawai tertentu berdasarkan ID pegawai. Fungsi ini mencari pegawai dengan ID yang diberikan, dan jika ditemukan, proyek baru ditambahkan ke daftar proyek pegawai tersebut. Jika pegawai tidak ditemukan, pesan kesalahan ditampilkan.
- Fungsi hapusProyek: Fungsi ini digunakan untuk menghapus proyek tertentu dari pegawai berdasarkan ID pegawai dan nama proyek. Fungsi ini mencari pegawai dan proyek yang sesuai, dan jika ditemukan, proyek

dihapus dari daftar. Jika proyek atau pegawai tidak ditemukan, pesan kesalahan ditampilkan.

- Fungsi tampilkanData: Fungsi ini digunakan untuk menampilkan semua pegawai dan proyek yang mereka kelola. Fungsi ini mengiterasi melalui daftar pegawai dan untuk setiap pegawai, mencetak nama dan ID mereka, serta semua proyek yang mereka kelola.
- Fungsi main: Fungsi ini berisi menu interaktif yang memungkinkan pengguna untuk memilih tindakan yang ingin dilakukan:
- Menambahkan pegawai baru. Menambahkan proyek untuk pegawai tertentu. Menghapus proyek dari pegawai tertentu. Menampilkan semua data pegawai dan proyek. Keluar dari program.
- Menu ditampilkan dalam loop, dan pengguna dapat memilih opsi dengan memasukkan angka yang sesuai. Input pengguna diproses dan fungsi yang sesuai dipanggil berdasarkan pilihan yang dibuat.

c. Output

```
Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 1
Masukkan Nama Pegawai : Andi
Masukkan ID Pegawai : P001

Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 1
Masukkan Nama Pegawai : Budi
Masukkan ID Pegawai : P002

Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 1
Masukkan Nama Pegawai : Citra
Masukkan ID Pegawai : P003
```

```
Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 2
Masukkan ID Pegawai : P001
Masukkan Nama Proyek : Aplikasi Mobile
Masukkan Durasi Proyek (bulan) : 12

Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 2
Masukkan ID Pegawai : P002
Masukkan Nama Proyek : Sistem Akuntansi
Masukkan Durasi Proyek (bulan) : 8

Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 2
Masukkan ID Pegawai : P003
Masukkan Nama Proyek : E-Commerce
Masukkan Durasi Proyek (bulan) : 10
```

```

Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 4
Pegawai : Citra (ID: P003)
Proyek : E-Commerce (Durasi : 10 bulan)
Pegawai : Budi (ID: P002)
Proyek : Sistem Akuntansi (Durasi : 8 bulan)
Pegawai : Andi (ID: P001)
Proyek : Aplikasi Mobile (Durasi : 12 bulan)

Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 1
Masukkan Nama Pegawai : AAAAA
Masukkan ID Pegawai : P004

Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 2
Masukkan ID Pegawai : P001
Masukkan Nama Proyek : Analisis Data
Masukkan Durasi Proyek (bulan) : 6

4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 4
Pegawai : AAAAA (ID: P004)
Pegawai : Citra (ID: P003)
Proyek : E-Commerce (Durasi : 10 bulan)
Pegawai : Budi (ID: P002)
Proyek : Sistem Akuntansi (Durasi : 8 bulan)
Pegawai : Andi (ID: P001)
Proyek : Analisis Data (Durasi : 6 bulan)
Proyek : Aplikasi Mobile (Durasi : 12 bulan)

Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 3
Masukkan ID Pegawai : P001
Masukkan Nama Proyek yang akan dihapus : Aplikasi Mobile

Menu:
1. Tambah Nama Pegawai
2. Tambah Nama Proyek
3. Hapus Nama Proyek
4. Tampilkan Data Pegawai & Proyek
5. Keluar
Menu: 4
Pegawai : AAAAA (ID: P004)
Pegawai : Citra (ID: P003)
Proyek : E-Commerce (Durasi : 10 bulan)
Pegawai : Budi (ID: P002)
Proyek : Sistem Akuntansi (Durasi : 8 bulan)
Pegawai : Andi (ID: P001)
Proyek : Analisis Data (Durasi : 6 bulan)

```

2. Unguided 2

a. Syntax

```

// Program Sistem Manajemen Buku Perpustakaan

#include <iostream>
#include <string>
using namespace std;

struct Buku {
    string judulBuku;
    string tanggalPengembalian;
    Buku* nextBuku;
};

struct Anggota {
    string namaAnggota;
    string idAnggota;
    Buku* headBuku;
    Anggota* nextAnggota;
};

Anggota* headAnggota = nullptr;

void tambahAnggota(string nama, string id) {
    Anggota* anggotaBaru = new Anggota;
    anggotaBaru->namaAnggota = nama;
    anggotaBaru->idAnggota = id;
    anggotaBaru->headBuku = nullptr;
    anggotaBaru->nextAnggota = headAnggota;
}

```

```
        headAnggota = anggotaBaru;
    }

void tambahBuku(string idAnggota, string judulBuku, string
tanggalPengembalian) {
    Anggota* currAnggota = headAnggota;
    while (currAnggota != nullptr) {
        if (currAnggota->idAnggota == idAnggota) {
            Buku* bukuBaru = new Buku;
            bukuBaru->judulBuku = judulBuku;
            bukuBaru->tanggalPengembalian = tanggalPengembalian;
            bukuBaru->nextBuku = currAnggota->headBuku;
            currAnggota->headBuku = bukuBaru;
            return;
        }
        currAnggota = currAnggota->nextAnggota;
    }
    cout << "Anggota dengan ID : " << idAnggota << " tidak dapat
ditemukan!\n";
}

void hapusAnggota(string idAnggota) {
    Anggota* currAnggota = headAnggota;
    Anggota* prevAnggota = nullptr;
    while (currAnggota != nullptr) {
        if (currAnggota->idAnggota == idAnggota) {
            if (prevAnggota == nullptr) {
                headAnggota = currAnggota->nextAnggota;
            } else {
                prevAnggota->nextAnggota = currAnggota->nextAnggota;
            }
            Buku* currBuku = currAnggota->headBuku;
            while (currBuku != nullptr) {
                Buku* temp = currBuku;
                currBuku = currBuku->nextBuku;
                delete temp;
            }
            delete currAnggota;
            return;
        }
        prevAnggota = currAnggota;
        currAnggota = currAnggota->nextAnggota;
    }
    cout << "Anggota dengan ID : " << idAnggota << " tidak dapat
ditemukan!\n";
}

void tampilkanData() {
    Anggota* currAnggota = headAnggota;
    while (currAnggota != nullptr) {
```

```
        cout << "Anggota : " << currAnggota->namaAnggota << " (ID : "
<< currAnggota->idAnggota << ")\\n";
        Buku* currBuku = currAnggota->headBuku;
        while (currBuku != nullptr) {
            cout << "  Buku : " << currBuku->judulBuku << " (Pengembalian
: " << currBuku->tanggalPengembalian << ")\\n";
            currBuku = currBuku->nextBuku;
        }
        currAnggota = currAnggota->nextAnggota;
    }
}

int main() {
    int menu;
    do {
        cout << "\\nMenu Peminjaman Perpustakaan:\\n";
        cout << "1. Tambah Nama Anggota\\n";
        cout << "2. Tambah Nama Buku\\n";
        cout << "3. Hapus Nama Anggota\\n";
        cout << "4. Tampilkan Data\\n";
        cout << "5. Keluar\\n";
        cout << "Menu : ";
        cin >> menu;
        cin.ignore();

        if (menu == 1) {
            string nama, id;
            cout << "Masukkan Nama Anggota: ";
            getline(cin, nama);
            cout << "Masukkan ID Anggota: ";
            getline(cin, id);
            tambahAnggota(nama, id);
        } else if (menu == 2) {
            string id, judulBuku, tanggalPengembalian;
            cout << "Masukkan ID Anggota: ";
            getline(cin, id);
            cout << "Masukkan Judul Buku: ";
            getline(cin, judulBuku);
            cout << "Masukkan Tanggal Pengembalian: ";
            getline(cin, tanggalPengembalian);
            tambahBuku(id, judulBuku, tanggalPengembalian);
        } else if (menu == 3) {
            string id;
            cout << "Masukkan ID Anggota yang akan dihapus: ";
            getline(cin, id);
            hapusAnggota(id);
        } else if (menu == 4) {
            tampilkanData();
        } else if (menu != 5) {
            cout << "Pilihan tidak valid.\\n";
        }
    } while (menu != 5);
}
```

```
    }  
    } while (menu != 5);  
  
    return 0;  
}
```

b. Penjelasan Syntax

- Inklusi Library: Program ini menggunakan library <iostream> untuk melakukan input dan output, serta <string> untuk memanfaatkan tipe data string yang menyimpan teks.
- Definisi Struktur Buku: Struktur Buku didefinisikan untuk merepresentasikan buku yang dipinjam oleh anggota perpustakaan. Setiap buku memiliki tiga atribut:
 - judulBuku: menyimpan judul buku.
 - tanggalPengembalian: menyimpan tanggal pengembalian buku.
 - nextBuku: pointer yang menunjuk ke buku berikutnya dalam daftar buku yang dipinjam oleh anggota.
- Definisi Struktur Anggota: Struktur Anggota didefinisikan untuk merepresentasikan anggota perpustakaan. Setiap anggota memiliki empat atribut: namaAnggota: menyimpan nama anggota. idAnggota: menyimpan ID anggota. headBuku: pointer yang menunjuk ke buku pertama yang dipinjam oleh anggota. nextAnggota: pointer yang menunjuk ke anggota berikutnya dalam daftar anggota. Variabel Global: Anggota* headAnggota didefinisikan sebagai pointer global yang menunjuk ke anggota pertama dalam daftar anggota. Awalnya, ini diatur ke nullptr, menandakan bahwa tidak ada anggota yang terdaftar.
- Fungsi tambahAnggota: Fungsi ini digunakan untuk menambahkan anggota baru ke dalam daftar. Fungsi ini menerima dua parameter: nama dan id. Sebuah node anggota baru dibuat, diisi dengan data yang diberikan, dan ditambahkan di depan daftar anggota.
- Fungsi tambahBuku: Fungsi ini digunakan untuk menambahkan buku baru yang dipinjam oleh anggota tertentu berdasarkan ID anggota. Fungsi ini mencari anggota dengan ID yang diberikan, dan jika ditemukan, buku baru ditambahkan ke daftar buku anggota tersebut. Jika anggota tidak ditemukan, pesan kesalahan ditampilkan.
- Fungsi hapusAnggota: Fungsi ini digunakan untuk menghapus anggota dari daftar berdasarkan ID anggota. Fungsi ini mencari anggota yang sesuai, dan jika ditemukan, anggota dihapus dari daftar. Selain itu, semua buku yang dipinjam oleh anggota tersebut juga dihapus dari memori. Jika anggota tidak ditemukan, pesan kesalahan ditampilkan.
- Fungsi tampilkanData: Fungsi ini digunakan untuk menampilkan semua anggota dan buku yang mereka pinjam. Fungsi ini mengiterasi melalui daftar anggota dan untuk setiap anggota, mencetak nama dan ID mereka, serta semua buku yang mereka pinjam beserta tanggal pengembaliannya.
- Fungsi main: Fungsi ini berisi menu interaktif yang memungkinkan pengguna untuk memilih tindakan yang ingin dilakukan: Menambahkan anggota baru. Menambahkan buku untuk anggota tertentu. Menghapus anggota dari daftar. Menampilkan semua data anggota dan buku yang

dipinjam. Keluar dari program. Menu ditampilkan dalam loop, dan pengguna dapat memilih opsi dengan memasukkan angka yang sesuai. Input pengguna diproses dan fungsi yang sesuai dipanggil berdasarkan pilihan yang dibuat.

c. Output

<pre>Menu Peminjaman Perpustakaan: 1. Tambah Nama Anggota 2. Tambah Nama Buku 3. Hapus Nama Anggota 4. Tampilkan Data 5. Keluar Menu : 1 Masukkan Nama Anggota: Rani Masukkan ID Anggota: A001 Menu Peminjaman Perpustakaan: 1. Tambah Nama Anggota 2. Tambah Nama Buku 3. Hapus Nama Anggota 4. Tampilkan Data 5. Keluar Menu : 1 Masukkan Nama Anggota: Dito Masukkan ID Anggota: A002 Menu Peminjaman Perpustakaan: 1. Tambah Nama Anggota 2. Tambah Nama Buku 3. Hapus Nama Anggota 4. Tampilkan Data 5. Keluar Menu : 1 Masukkan Nama Anggota: Vina Masukkan ID Anggota: A003</pre>	<pre>Menu Peminjaman Perpustakaan: 1. Tambah Nama Anggota 2. Tambah Nama Buku 3. Hapus Nama Anggota 4. Tampilkan Data 5. Keluar Menu : 2 Masukkan ID Anggota: A001 Masukkan Judul Buku: Pemrograman C++ Masukkan Tanggal Pengembalian: 10/12/2024 Menu Peminjaman Perpustakaan: 1. Tambah Nama Anggota 2. Tambah Nama Buku 3. Hapus Nama Anggota 4. Tampilkan Data 5. Keluar Menu : 2 Masukkan ID Anggota: A002 Masukkan Judul Buku: Algoritma Pemrograman Masukkan Tanggal Pengembalian: 15/12/2024 Menu Peminjaman Perpustakaan: 1. Tambah Nama Anggota 2. Tambah Nama Buku 3. Hapus Nama Anggota 4. Tampilkan Data 5. Keluar Menu : 2 Masukkan ID Anggota: A001 Masukkan Judul Buku: Struktur Data Masukkan Tanggal Pengembalian: 10/24/2024</pre>	<pre>Menu Peminjaman Perpustakaan: 1. Tambah Nama Anggota 2. Tambah Nama Buku 3. Hapus Nama Anggota 4. Tampilkan Data 5. Keluar Menu : 4 Anggota : Vina (ID : A003) Anggota : Dito (ID : A002) Buku : Algoritma Pemrograman (Pengembalian : 15/12/2024) Anggota : Rani (ID : A001) Buku : Struktur Data (Pengembalian : 10/24/2024) Buku : Pemrograman C++ (Pengembalian : 10/12/2024) Menu Peminjaman Perpustakaan: 1. Tambah Nama Anggota 2. Tambah Nama Buku 3. Hapus Nama Anggota 4. Tampilkan Data 5. Keluar Menu : 3 Masukkan ID Anggota yang akan dihapus: A002 Menu Peminjaman Perpustakaan: 1. Tambah Nama Anggota 2. Tambah Nama Buku 3. Hapus Nama Anggota 4. Tampilkan Data 5. Keluar Menu : 4 Anggota : Vina (ID : A003) Anggota : Rani (ID : A001) Buku : Struktur Data (Pengembalian : 10/24/2024) Buku : Pemrograman C++ (Pengembalian : 10/12/2024)</pre>
--	---	--

V. KESIMPULAN

Kesimpulan yang saya dapatkan bahwa struktur data ini memungkinkan pengelolaan data yang kompleks dengan hubungan hierarkis antara elemen-elemen dalam list induk dan anak. Multi Linked List sangat berguna untuk representasi data yang memiliki keterkaitan antar kelompok, seperti sistem organisasi atau database relasional. Dengan memanfaatkan operasi seperti insert dan delete, serta manajemen memori yang tepat, Multi Linked List memberikan fleksibilitas tinggi dalam pengolahan data, meskipun membutuhkan pemahaman mendalam tentang manipulasi pointer dan struktur dinamis.