

LAPORAN PRAKTIKUM
Modul 13
“Multi Linked
List ”



Disusun Oleh:
Ryan Gabriel Togar Simamora
(2311104045)
Kelas S1SE-07-02

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

A. TUJUAN PRAKTIKUM

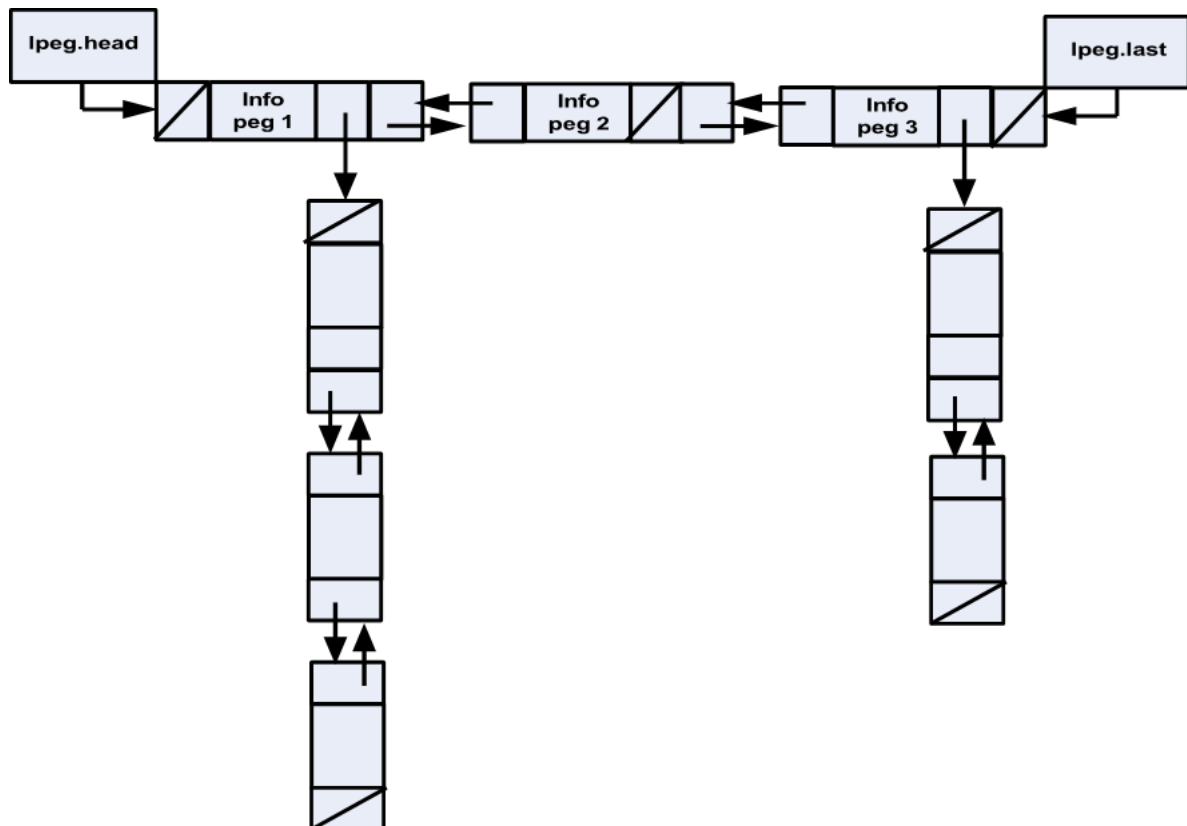
- Memahami penggunaan *Multi Linked list*.
- Mengimplementasikan *Multi Linked list* dalam beberapa studi kasus.

B. DASAR TEORI

Multi Linked List

Multi Linked List adalah struktur data yang terdiri dari kumpulan linked list yang saling terhubung. Setiap elemen dalam Multi Linked List dapat membentuk linked list tersendiri, di mana ada hubungan antara list utama (induk) dan list sekunder (anak). Konsep ini memungkinkan pengorganisasian data yang lebih kompleks, seperti hierarki atau hubungan antar elemen.

Contoh :



Komponen Utama Multi Linked List

- ❖ List Induk: Berfungsi sebagai list utama yang menunjuk ke list anak.
- ❖ List Anak: Merupakan sub-list yang diakses melalui elemen pada list induk.

Sebagai contoh, dalam aplikasi pegawai, setiap elemen pada list induk dapat merepresentasikan data pegawai, dan setiap pegawai memiliki list anak yang berisi informasi tambahan, seperti daftar proyek yang dikerjakan.

Operasi pada Multi Linked List

1.Insert (Penyisipan):

- ❖ Insert Anak: Penyisipan elemen anak memerlukan referensi elemen induk sebagai lokasi penyisipan. Operasi ini sering menggunakan pendekatan seperti insert last untuk menempatkan elemen anak pada akhir list anak.
- ❖ Insert Induk: Penyisipan elemen induk mengikuti prosedur standar linked list (single, double, atau circular linked list).

2.Delete (Penghapusan):

- ❖ Delete Anak: Menghapus elemen anak memerlukan identifikasi induk terkait. Semua elemen anak yang terhubung dengan induk tertentu juga dapat dihapus.
- ❖ Delete Induk: Menghapus elemen induk akan secara otomatis menghapus semua elemen anak yang terkait.

Manajemen Memori Alokasi dan dealokasi elemen dalam Multi Linked List menggunakan pointer untuk mengelola elemen secara dinamis. Prosedur seperti alokasi dan dealokasi memastikan penggunaan memori yang efisien.

Ilustrasi dan Implementasi

Pada implementasi, list induk dan anak masing-masing direpresentasikan menggunakan struktur data yang memiliki pointer untuk menyimpan referensi ke elemen berikutnya (dan sebelumnya untuk double linked list).

Fungsi seperti insertFirst, insertLast, findElm, dan printInfo digunakan untuk memanipulasi dan menampilkan data pada list.

Keuntungan Penggunaan Multi Linked List

Mempermudah pengelolaan data yang memiliki hubungan hierarkis.

Fleksibilitas dalam menambah dan menghapus elemen tanpa mengganggu struktur data lainnya.

Cocok untuk aplikasi yang membutuhkan pengorganisasian data yang kompleks, seperti sistem manajemen proyek atau struktur organisasi.

C. GUIDED 1

Sourcecode :

File Guided1.cpp

```
#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;

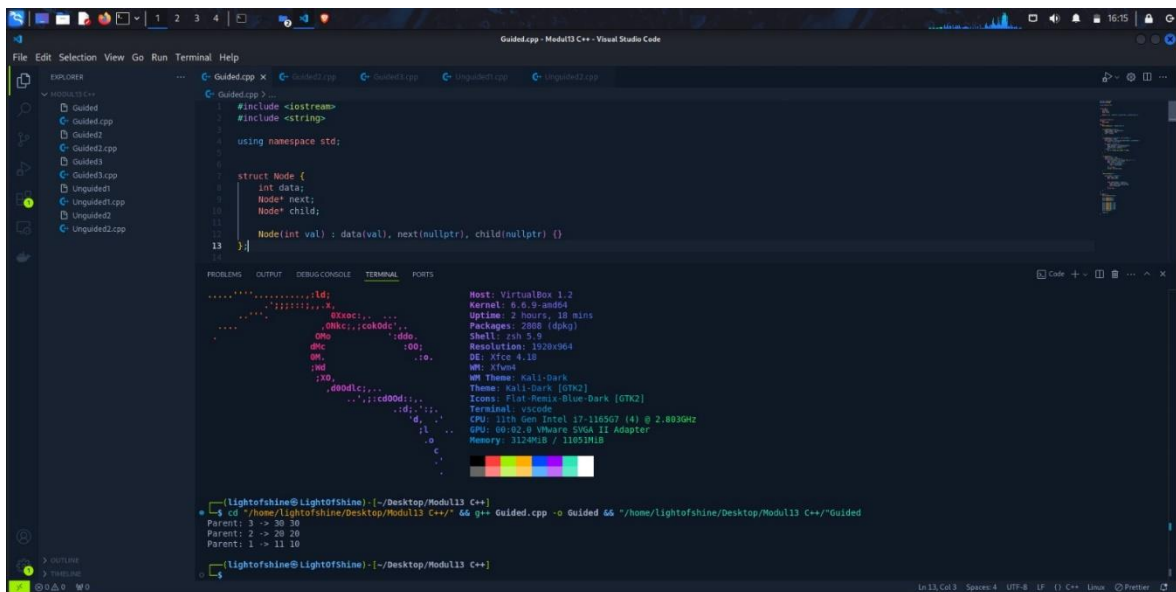
    mList.addParent(1);
    mList.addParent(2);
    mList.addParent(3);

    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;
}

```

Output :



```

#include <iostream>
#include <string>

using namespace std;

struct Node {
    int data;
    Node* next;
    Node* child;
};

Node(int val) : data(val), next(nullptr), child(nullptr) {}

// ... (rest of the code is partially visible)

Host: VirtualBox 1.2
Kernel: 6.9.9-amd64
Uptime: 2 hours, 18 mins
Packages: 2088 (dpkg)
Shell: zsh 5.9
Resolution: 1920x1064
DE: Alice 4.10
WM: Xfwm4
WM Theme: Kali-Dark
Theme: Kali-Dark [GTK2]
Icons: Flat-Remix-Blue-Dark [GTK2]
Terminal: vscode
CPU: 11th Gen Intel i7-116507 (4) @ 2.803GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 3124MiB / 11051MiB

lightofshine@lightofshine: ~/Desktop/Modul13 C++
$ cd ~/home/lightofshine/Desktop/Modul13 C++/ 46 g++ Guided.cpp -o Guided && ./Guided
Parent: 3 -> 30 30
Parent: 2 -> 20 20
Parent: 1 -> 11 10
lightofshine@lightofshine:~/Desktop/Modul13 C++$
  
```

GUIDED 2

Sourcecode

```

#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");

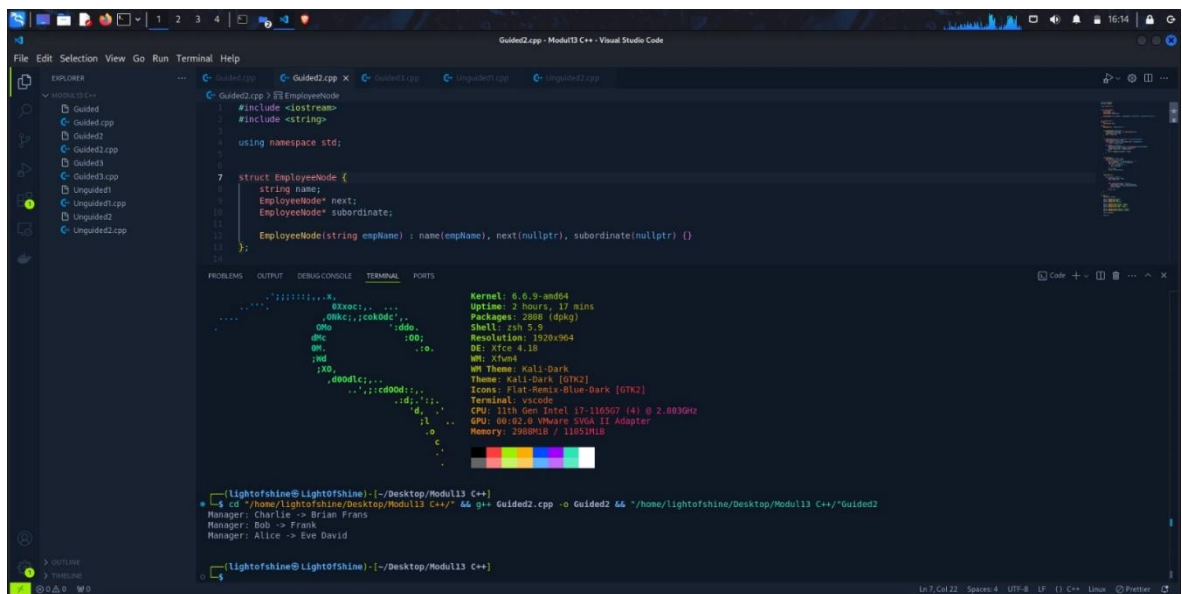
empList.addSubordinate("Alice", "David");
empList.addSubordinate("Alice", "Eve");
empList.addSubordinate("Bob", "Frank");

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

empList.display();

return 0;
}
```

Output :



```

Kernel: 6.6.9-amd64
Uptime: 2 hours, 17 mins
Packages: 2888 (dpkg)
Shell: zsh 5.9
Resolution: 1920x964
DE: Xfce 4.18
WM: Xfwm
WM Theme: Kali-Dark
Theme: Kali-Dark [GTK2]
Icons: Flat-Remix-Blue-Dark [GTK2]
Terminal: xterm
CPU: 11th Gen Intel i7-11650T (4) @ 2.80GHz
GPU: 60-02.0 VMware SVGA II Adapter
Memory: 2968MiB / 11651MiB

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

```

GUIDED 3

Sourcecode

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

using namespace std;

// Struktur untuk node karyawan
struct EmployeeNode {
    string name; // Nama karyawan
    EmployeeNode* next; // Pointer ke karyawan berikutnya
    EmployeeNode* subordinate; // Pointer ke subordinate pertama
};
```



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

// Kelas untuk Multi-Linked List Karyawan
class EmployeeList {
private:
    EmployeeNode* head; // Pointer ke kepala list

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

    // Menambahkan karyawan (induk)
    void addEmployee(string name) {
        EmployeeNode* newEmployee = new EmployeeNode(name);
        newEmployee->next = head; // Menyambungkan ke karyawan sebelumnya
        head = newEmployee; // Memperbarui head
    }

    // Menambahkan subordinate ke karyawan tertentu
    void addSubordinate(string managerName, string subordinateName) {
        EmployeeNode* manager = head;
        while (manager != nullptr && manager->name != managerName) {
            manager = manager->next;
        }
        if (manager != nullptr) { // Jika manajer ditemukan
            EmployeeNode* newSubordinate = new
EmployeeNode(subordinateName);
            newSubordinate->next = manager->subordinate; // Menyambungkan ke
subordinate sebelumnya
            manager->subordinate = newSubordinate; // Memperbarui subordinate
        } else {
            cout << "Manager not found!" << endl;
        }
    }

    // Menghapus karyawan (induk)
    void deleteEmployee(string name) {
        EmployeeNode** current = &head;
        while (*current != nullptr && (*current)->name != name) {
            current = &((*current)->next);
        }

        if (*current != nullptr) { // Jika karyawan ditemukan
            EmployeeNode* toDelete = *current;
            *current = (*current)->next;

            // Hapus semua subordinate dari node ini
            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;
}
}

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

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

        if (*currentSub != nullptr) { // Jika subordinate ditemukan
            EmployeeNode* toDelete = *currentSub;
            *currentSub = (*currentSub)->next; // Menghapus dari list

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

// Menampilkan daftar karyawan dan subordinate mereka
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() {
    // Destructur untuk membersihkan memori
    while (head != nullptr) {
        EmployeeNode* temp = head;
        head = head->next;

        // Hapus semua subordinate dari node ini
        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");

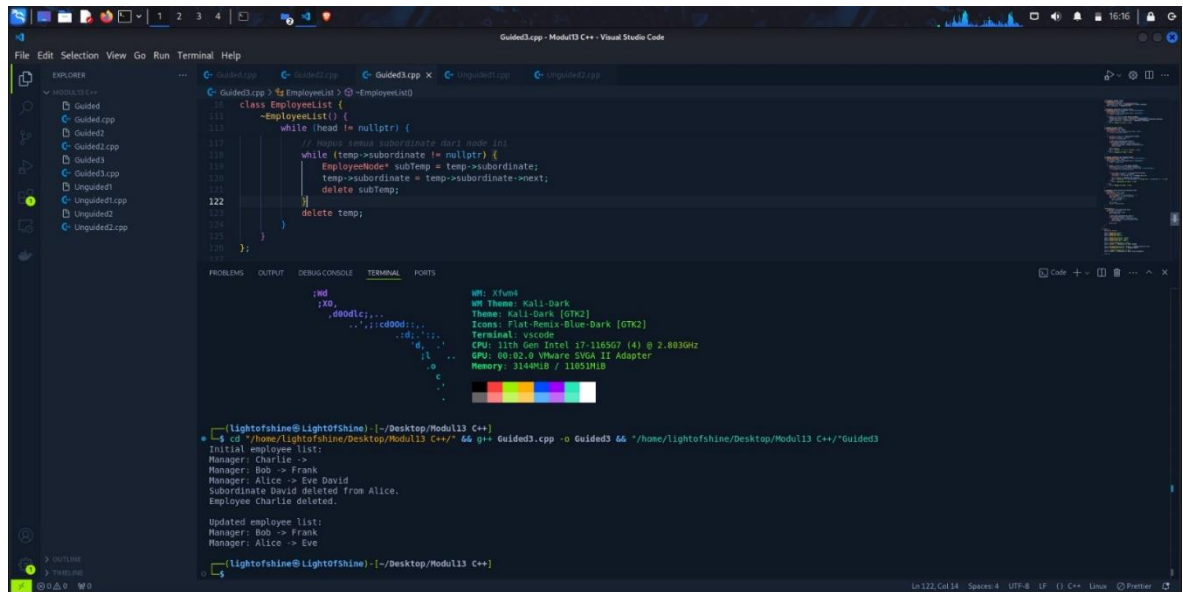
    cout << "Initial employee list:" << endl;
    empList.display(); // Menampilkan isi daftar karyawan

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

    cout << "\nUpdated employee list:" << endl;
    empList.display(); // Menampilkan isi daftar setelah penghapusan

    return 0;
}
```

Output :



```

Guided3.cpp - Modul13 C++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help

class EmployeeList {
    ~EmployeeList() {
        while (head != nullptr) {
            // Hapus semua subordinate dari node ini
            while (temp->subordinate != nullptr) {
                EmployeeNode* subTemp = temp->subordinate;
                temp->subordinate = temp->subordinate->next;
                delete subTemp;
            }
            delete temp;
        }
    }
};

// lightofshine@lightofshine: ~/Desktop/Modul13 C++
$ cd ~/home/lightofshine/Desktop/Modul13 C++ && g++ Guided3.cpp -o Guided3 && ./Guided3
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
  
```

D. UNGUIDED

1. Manajemen Data Pegawai dan Proyek

Buatlah program menggunakan Multi Linked List untuk menyimpan data pegawai dan proyek yang dikelola setiap pegawai.

- Setiap pegawai memiliki data: Nama Pegawai dan ID Pegawai.
- Setiap proyek memiliki data: Nama Proyek** dan **Durasi (bulan).

Instruksi:

1. Masukkan data pegawai berikut:

- Pegawai 1: Nama = "Andi", ID = "P001".
- Pegawai 2: Nama = "Budi", ID = "P002".
- Pegawai 3: Nama = "Citra", ID = "P003".

2. Tambahkan proyek ke pegawai:

- Proyek 1: Nama = "Aplikasi Mobile", Durasi = 12 bulan (Untuk Andi).
- Proyek 2: Nama = "Sistem Akuntansi", Durasi = 8 bulan (Untuk Budi).
- Proyek 3: Nama = "E-commerce", Durasi = 10 bulan (Untuk Citra).

3. Tambahkan proyek baru:

- Proyek 4: Nama = "Analisis Data", Durasi = 6 bulan (Untuk Andi).

4. Hapus proyek "Aplikasi Mobile" dari Andi.

5. Tampilkan data pegawai dan proyek mereka.

Jawab :

Source Code :

```

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

struct Proyek {
    string namaProyek;
  
```

```
    int durasi;
    Proyek* next;
};

struct Pegawai {
    string namaPegawai;
    string idPegawai;
    Proyek* proyekHead;
    Pegawai* next;
};

class MultiLinkedList {
private:
    Pegawai* head;

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

    void tambahPegawai(string nama, string id) {
        Pegawai* newPegawai = new Pegawai{ nama, id, nullptr, nullptr };
        if (!head) {
            head = newPegawai;
        } else {
            Pegawai* temp = head;
            while (temp->next) temp = temp->next;
            temp->next = newPegawai;
        }
    }

    void tambahProyek(string id, string namaProyek, int durasi) {
        Pegawai* temp = head;
        while (temp && temp->idPegawai != id) temp = temp->next;
        if (temp) {
            Proyek* newProyek = new Proyek{ namaProyek, durasi, nullptr };
            if (!temp->proyekHead) {
                temp->proyekHead = newProyek;
            } else {
                Proyek* pTemp = temp->proyekHead;
                while (pTemp->next) pTemp = pTemp->next;
                pTemp->next = newProyek;
            }
        }
    }

    void hapusProyek(string id, string namaProyek) {
        Pegawai* temp = head;
        while (temp && temp->idPegawai != id) temp = temp->next;
        if (temp) {
            Proyek* pTemp = temp->proyekHead;
            Proyek* prev = nullptr;
```

```

    while (pTemp && pTemp->namaProyek != namaProyek) {
        prev = pTemp;
        pTemp = pTemp->next;
    }
    if (pTemp) {
        if (!prev) {
            temp->proyekHead = pTemp->next;
        } else {
            prev->next = pTemp->next;
        }
        delete pTemp;
    }
}

void tampilkanData() {
    Pegawai* temp = head;
    while (temp) {
        cout << "Pegawai: " << temp->namaPegawai << " (" << temp->idPegawai << ")\n";
        Proyek* pTemp = temp->proyekHead;
        while (pTemp) {
            cout << " - Proyek: " << pTemp->namaProyek << ", Durasi: " << pTemp->durasi
            << " bulan\n";
            pTemp = pTemp->next;
        }
        temp = temp->next;
    }
}

int main() {
    MultiLinkedList mll;

    mll.tambahPegawai("Andi", "P001");
    mll.tambahPegawai("Budi", "P002");
    mll.tambahPegawai("Citra", "P003");

    mll.tambahProyek("P001", "Aplikasi Mobile", 12);
    mll.tambahProyek("P002", "Sistem Akuntansi", 8);
    mll.tambahProyek("P003", "E-commerce", 10);
    mll.tambahProyek("P001", "Analisis Data", 6);

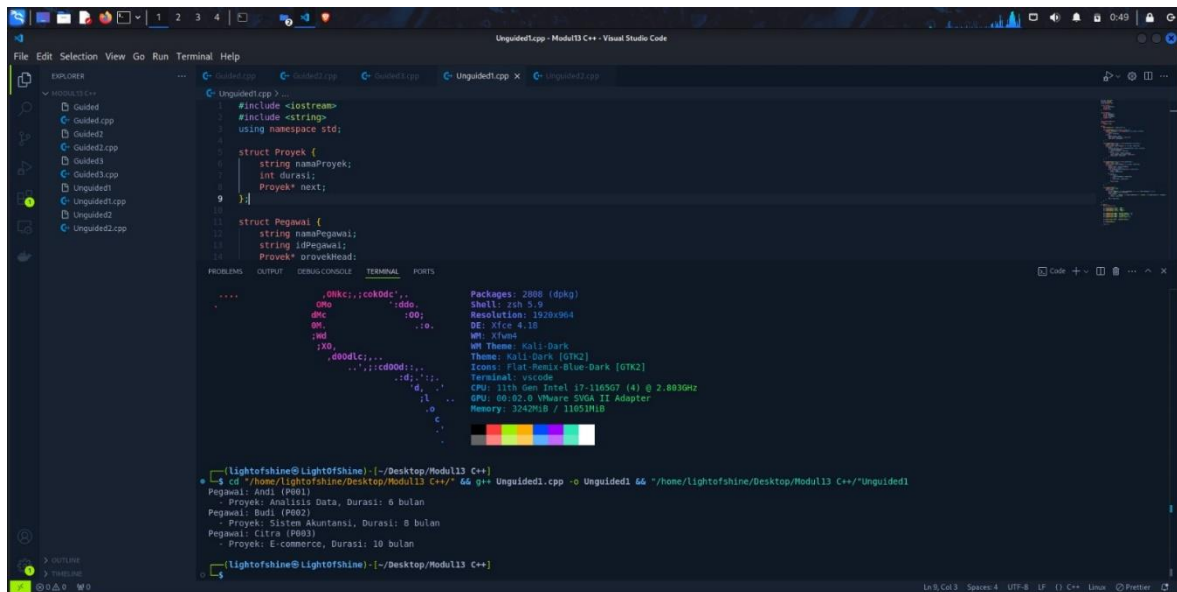
    mll.hapusProyek("P001", "Aplikasi Mobile");

    mll.tampilkanData();

    return 0;
}

```

Output :



```

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

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

struct Pegawai {
    string namaPegawai;
    string idPegawai;
    Proyek* proyekHead;
};

int main(int argc, char* argv[]) {
    if (argc < 2) {
        cout << "Usage: ./program <command> <data>" << endl;
        return 1;
    }

    string command = argv[1];
    string data = argv[2];

    if (command == "add") {
        // Add new data
    } else if (command == "delete") {
        // Delete data
    } else if (command == "update") {
        // Update data
    } else if (command == "display") {
        // Display all data
    } else if (command == "search") {
        // Search data
    } else {
        cout << "Invalid command" << endl;
        return 1;
    }

    return 0;
}

```

2. Sistem Manajemen Buku Perpustakaan

Gunakan Multi Linked List untuk menyimpan data anggota perpustakaan dan daftar buku yang dipinjam.

- Setiap anggota memiliki data: Nama Anggota dan ID Anggota.
- Setiap buku memiliki data: Judul Buku dan Tanggal Pengembalian.

Instruksi:

- Masukkan data anggota berikut:
 - Anggota 1: Nama = "Rani", ID = "A001".
 - Anggota 2: Nama = "Dito", ID = "A002".
 - Anggota 3: Nama = "Vina", ID = "A003".
- Tambahkan buku yang dipinjam:
 - Buku 1: Judul = "Pemrograman C++", Pengembalian = "01/12/2024" (Untuk Rani).
 - Buku 2: Judul = "Algoritma Pemrograman", Pengembalian = "15/12/2024" (Untuk Dito).
- Tambahkan buku baru:
 - Buku 3: Judul = "Struktur Data", Pengembalian = "10/12/2024" (Untuk Rani).
- Hapus anggota Dito beserta buku yang dipinjam.
- Tampilkan seluruh data anggota dan buku yang dipinjam.

Jawab :

Sourcecode :

```

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

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

```

```
};

struct Anggota {
    string namaAnggota;
    string idAnggota;
    Buku* bukuHead;
    Anggota* next;
};

class MultiLinkedList {
private:
    Anggota* head;

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

    void tambahAnggota(string nama, string id) {
        Anggota* newAnggota = new Anggota{nama, id, nullptr, nullptr};
        if (!head) {
            head = newAnggota;
        } else {
            Anggota* temp = head;
            while (temp->next) temp = temp->next;
            temp->next = newAnggota;
        }
    }

    void tambahBuku(string id, string judulBuku, string tanggalPengembalian) {
        Anggota* temp = head;
        while (temp && temp->idAnggota != id) temp = temp->next;
        if (temp) {
            Buku* newBuku = new Buku{judulBuku, tanggalPengembalian, nullptr};
            if (!temp->bukuHead) {
                temp->bukuHead = newBuku;
            } else {
                Buku* bTemp = temp->bukuHead;
                while (bTemp->next) bTemp = bTemp->next;
                bTemp->next = newBuku;
            }
        }
    }

    void hapusAnggota(string id) {
        Anggota* temp = head;
        Anggota* prev = nullptr;
        while (temp && temp->idAnggota != id) {
            prev = temp;
            temp = temp->next;
        }
        if (temp) {
```



```
        if (!prev) {
            head = temp->next;
        } else {
            prev->next = temp->next;
        }

        Buku* bTemp = temp->bukuHead;
        while (bTemp) {
            Buku* toDelete = bTemp;
            bTemp = bTemp->next;
            delete toDelete;
        }
        delete temp;
    }
}

void tampilkanData() {
    Anggota* temp = head;
    while (temp) {
        cout << "Anggota: " << temp->namaAnggota << " (" << temp->idAnggota << ")\n";
        Buku* bTemp = temp->bukuHead;
        while (bTemp) {
            cout << " - Buku: " << bTemp->judulBuku << ", Pengembalian: " << bTemp-
            >tanggalPengembalian << "\n";
            bTemp = bTemp->next;
        }
        temp = temp->next;
    }
}

};

int main() {
    MultiLinkedList mll;

    mll.tambahAnggota("Rani", "A001");
    mll.tambahAnggota("Dito", "A002");
    mll.tambahAnggota("Vina", "A003");

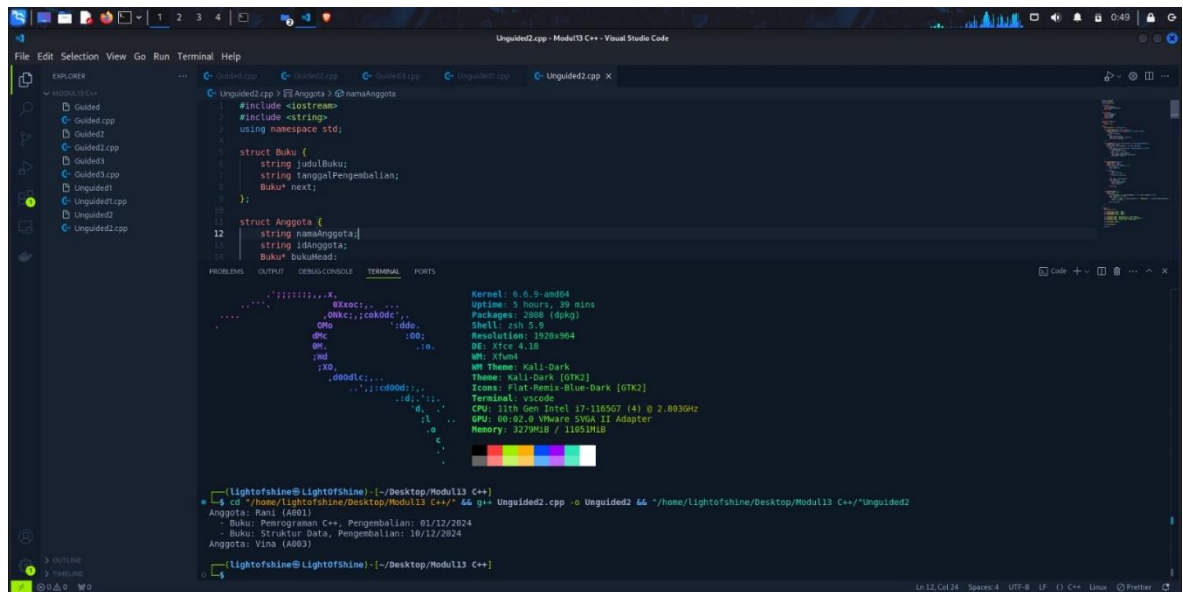
    mll.tambahBuku("A001", "Pemrograman C++", "01/12/2024");
    mll.tambahBuku("A002", "Algoritma Pemrograman", "15/12/2024");
    mll.tambahBuku("A001", "Struktur Data", "10/12/2024");

    mll.hapusAnggota("A002");

    mll.tampilkanData();

    return 0;
}
```

Output :



```

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

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

struct Anggota {
    string namaAnggota;
    string idAnggota;
    Buku* bukuHead;
};

int main() {
    // ... (code for creating and linking nodes) ...

    // ... (code for displaying the list) ...

    return 0;
}
    
```

```

Kernel: 6.6.9-amd64
Uptime: 3 hours, 39 mins
Packages: 2808 (dpkg)
Shell: zsh 5.9
Resolution: 1920x964
DE: MFC 4.18
WM: Xfwm4
WM Theme: kali-Dark
Theme: Kali-Dark [GTK2]
Icons: Flat-Remix-Blue-Dark [GTK2]
Terminal: xcode
CPU: 11th Gen Intel i7-11650T (4) @ 2.803GHz
GPU: 00:02.0 VMware SVGA II Adapter
Memory: 3279MiB / 11551MiB

lightshime@LightOfShime: ~/Desktop/Modul13 C++
$ cd ~/home/LightOfShime/Desktop/Modul13 C++/ && g++ Unguided2.cpp -o Unguided2 && ./Unguided2
Anggota: Rani (A001)
- Buku: Pemrograman C++, Pengembalian: 01/12/2024
- Buku: Struktur Data, Pengembalian: 10/12/2024
Anggota: Vina (A002)
    
```

E. KESIMPULAN

Multi Linked List adalah struktur data yang terdiri dari kumpulan linked list yang saling terhubung, di mana elemen dari list induk dapat menunjuk ke list anak. Struktur ini digunakan untuk mengelola data hierarkis dengan fleksibilitas tinggi, memungkinkan penyisipan, penghapusan, dan pengelompokan data tanpa mengubah struktur utama. Multi Linked List sangat berguna untuk aplikasi dengan relasi majemuk, seperti manajemen pegawai dan proyek atau sistem perpustakaan, meskipun memerlukan pengelolaan memori yang cermat.

