LAPORAN PRAKTIKUM
Modul 13
“Multi Linked List”

Disusun Oleh:
Dewi Atika Muthi -2211104042
SE-07-02

Dosen:
Wahyu Andi Saputra

PROGRAM STUDI S1 SOFTWARE ENGINEERING
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
PURWOKERTO
2024

1. **Tujuan**
    - Memahami konsep Multi Linked List
    - Mengimplementasikan struktur Multi Linked List dalam berbagai kasus
    - Mempraktikkan operasi dasar pada Multi Linked List seperti penambahan, penghapusan, dan traversal

2. **Landasan Teori**

    Multi Linked List adalah struktur data yang memungkinkan setiap elemen dalam list memiliki beberapa pointer tambahan, membentuk hubungan kompleks antar elemen. Karakteristik utama Multi Linked List:
    - Setiap elemen dapat memiliki hubungan dengan elemen lain melalui beberapa pointer
    - Memungkinkan representasi struktur hierarkis atau relasional
    - Dapat membentuk list dalam list (nested list)

    Struktur dasar Multi Linked List biasanya terdiri dari:
    - Pointer utama (head)
    - Pointer tambahan untuk menghubungkan elemen terkait
    - Kemampuan untuk memiliki sub-list atau anak list

    Logika Operasi pada Multi Linked List:

    Operasi Insert

    a. Insert Parent (Menambah Node Utama)
    - Membuat node baru dengan data yang diberikan
    - Jika list kosong, node menjadi head
    - Jika sudah ada node, tambahkan di akhir list utama

    b. Insert Child (Menambah Anak pada Parent)
    - Temukan parent yang sesuai
    - Buat node anak baru
    - Jika parent belum memiliki anak, jadikan anak pertama
    - Jika sudah ada anak, tambahkan di akhir list anak

    Operasi Delete

    a. Delete Parent
    - Hapus node parent dari list utama
    - Secara rekursif hapus semua child yang terkait
    - Bebaskan memori node yang dihapus

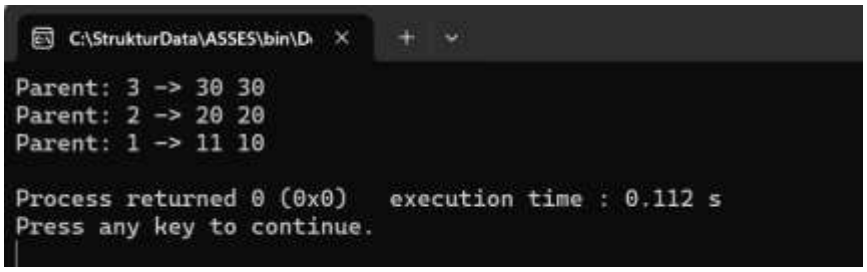
    b. Delete Child
    - Temukan parent dari child yang akan dihapus
    - Hapus child dari list anak parent
    - Bebaskan memori node child

3. Guided

1) Data 1 (insert parent n child)

```
#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:



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

Process returned 0 (0x0)   execution time : 0.112 s
Press any key to continue.
```

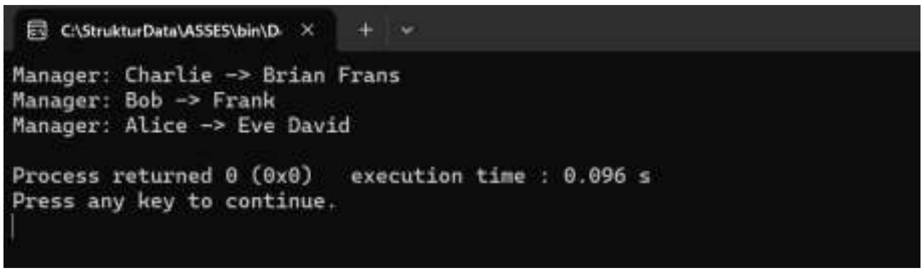
Penjelasan Program:

- Dapat menambahkan parent (1, 2, 3) dengan fungsi `addParent`
- Dapat menambahkan child (anak) ke parent tertentu ([10, 11 > 1], [20, 20 > 2], [30, 30 > 3]) dengan fungsi `addChild`
- Menampilkan struktur hierarkis list dengan fungsi `display`

2) Data 2 (insert parent n child)

```
#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:



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

Process returned 0 (0x0)   execution time : 0.096 s
Press any key to continue.
```

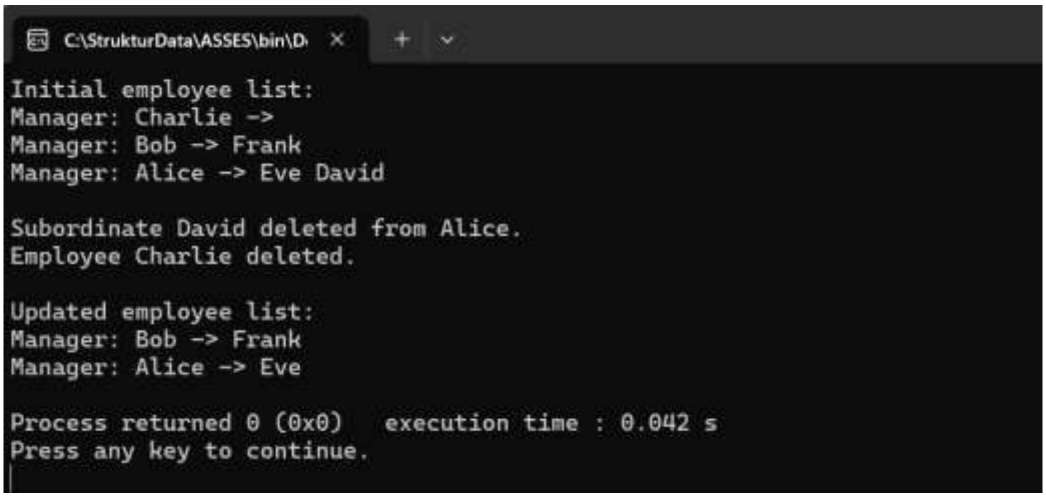
Penjelasan progam:

- Representasi struktur organisasi
- Menambah manager dan bawahannya
- Menampilkan hierarki organisasi

3) Data 3 (insert parent n child implement delete)

```
#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() {
        // Destructor 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

    cout << endl;

    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:



```
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 : 0.042 s
Press any key to continue.
```

Penjelasan program:

- Kemampuan menghapus karyawan (manager: Charlie)
- Menghapus bawahan spesifik (David dari Alice)
- Manajemen memori dengan destructor

4. Unguided

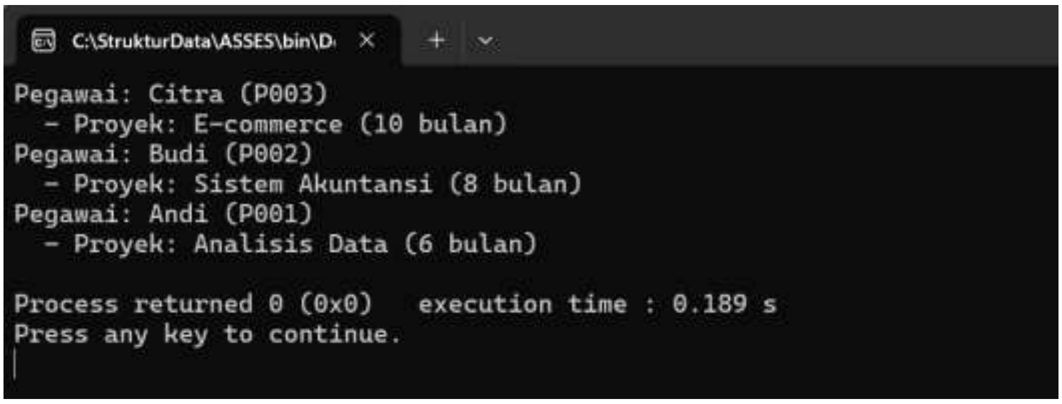
1) Study Case 1: Manajemen Data Pegawai dan Proyek

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

Source code:

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

struct Project {
    string name;
    int duration;
    Project* next;
};

struct Employee {
    string name;
    string id;
    Project* projects;
    Employee* next;
};

Employee* head = nullptr;

void addEmployee(string name, string id) {
    Employee* newEmployee = new Employee{name, id, nullptr, head};
    head = newEmployee;
}

void addProject(string empId, string projectName, int duration) {
    Employee* temp = head;
    while (temp && temp->id != empId) {
        temp = temp->next;
    }
    if (temp) {
        Project* newProject = new Project{projectName, duration, temp-
    >projects};
        temp->projects = newProject;
    }
}

void removeProject(string empId, string projectName) {
    Employee* temp = head;
    while (temp && temp->id != empId) {
        temp = temp->next;
    }
    if (temp) {
        Project* curr = temp->projects;
        Project* prev = nullptr;
        while (curr && curr->name != projectName) {
            prev = curr;
            curr = curr->next;
        }
        if (curr) {
            if (prev) {
                prev->next = curr->next;
            } else {
                temp->projects = curr->next;
            }
            delete curr;
        }
    }
}

void displayData() {
    Employee* emp = head;
    while (emp) {
        cout << "Pegawai: " << emp->name << " (" << emp->id << ")\n";
        Project* proj = emp->projects;
        while (proj) {
            cout << "  - Proyek: " << proj->name << " (" << proj->duration << "
    bulan)\n";
            proj = proj->next;
        }
        emp = emp->next;
    }
}

int main() {
    addEmployee("Andi", "P001");
    addEmployee("Budi", "P002");
    addEmployee("Citra", "P003");

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

    removeProject("P001", "Aplikasi Mobile");

    displayData();
    return 0;
}
```

Output:



```
C:\StrukturData\ASSES\bin\D

Pegawai: Citra (P003)
  - Proyek: E-commerce (10 bulan)
Pegawai: Budi (P002)
  - Proyek: Sistem Akuntansi (8 bulan)
Pegawai: Andi (P001)
  - Proyek: Analisis Data (6 bulan)

Process returned 0 (0x0)   execution time : 0.189 s
Press any key to continue.
```

Penjelasan program:

- Program memulai dengan menambahkan tiga pegawai: Andi, Budi, dan Citra, dengan fungsi `addEmploye`
- Kemudian, proyek ditambahkan ke pegawai masing-masing, sesuai dengan instruksi, dengan fungsi `addProject`
- Proyek tambahan juga dimasukkan untuk Andi, dan proyek tertentu dihapus dari daftar proyeknya dengan fungsi `removeProject`
- Untuk mencetak seluruh data pegawai beserta proyek mereka, menggunakan fungsi `displayData`.

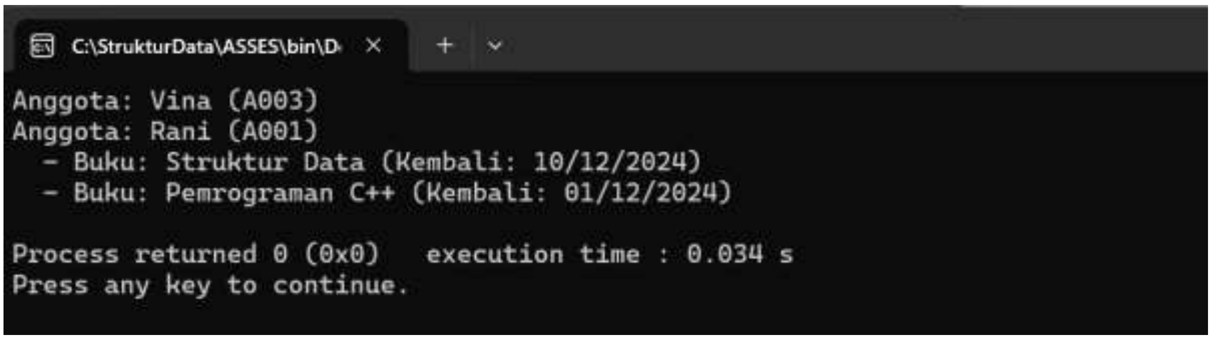
2) **Study Case 2: Sistem Manajemen Buku Perpustakaan**

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

Source code:

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

struct Book {
    string title;
    string returnDate;
    Book* next;
};

struct Member {
    string name;
    string id;
    Book* books;
    Member* next;
};

Member* memberHead = nullptr;

void addMember(string name, string id) {
    Member* newMember = new Member{name, id, nullptr, memberHead};
    memberHead = newMember;
}

void addBook(string memberId, string bookTitle, string returnDate) {
    Member* temp = memberHead;
    while (temp && temp->id != memberId) {
        temp = temp->next;
    }
    if (temp) {
        Book* newBook = new Book{bookTitle, returnDate, temp->books};
        temp->books = newBook;
    }
}

void removeMember(string memberId) {
    Member* curr = memberHead;
    Member* prev = nullptr;
    while (curr && curr->id != memberId) {
        prev = curr;
        curr = curr->next;
    }
    if (curr) {
        if (prev) {
            prev->next = curr->next;
        } else {
            memberHead = curr->next;
        }
        while (curr->books) {
            Book* toDelete = curr->books;
            curr->books = curr->books->next;
            delete toDelete;
        }
        delete curr;
    }
}

void displayMembers() {
    Member* member = memberHead;
    while (member) {
        cout << "Anggota: " << member->name << " (" << member->id << ")" <<
    endl;
        Book* book = member->books;
        while (book) {
            cout << "  - Buku: " << book->title << " (Kembali: " << book-
    >returnDate << ")" << endl;
            book = book->next;
        }
        member = member->next;
    }
}

int main() {
    addMember("Rani", "A001");
    addMember("Dito", "A002");
    addMember("Vina", "A003");

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

    removeMember("A002");

    displayMembers();
    return 0;
}
```

Output:



```
Anggota: Vina (A003)
Anggota: Rani (A001)
  - Buku: Struktur Data (Kembali: 10/12/2024)
  - Buku: Pemrograman C++ (Kembali: 01/12/2024)

Process returned 0 (0x0)   execution time : 0.034 s
Press any key to continue.
```

Penjelasan program:

- Program dimulai dengan menambahkan tiga anggota: Rani, Dito, dan Vina dengan fungsi `addMember`
- Buku-buku yang dipinjam oleh anggota dimasukkan sesuai dengan instruksi, seperti buku "Pemrograman C++" untuk Rani dan "Algoritma Pemrograman" untuk Dito dengan fungsi `addBook`
- Kemudian, anggota Dito dihapus dari daftar beserta semua buku yang dipinjamnya dengan fungsi `removeMember`
- Akhirnya, program mencetak seluruh data anggota perpustakaan beserta daftar buku yang mereka pinjam dengana fungsi `displayMemebers`.

5. Kesimpulan

Praktikum ini memberikan pemahaman mendalam tentang Multi Linked List, mengeksplorasi konsep, implementasi, dan aplikasi praktisnya dalam struktur data yang kompleks.