LAPORAN PRAKTIKUM

Modul 13

“Multi Linked List”

Disusun Oleh:

Marvel Sanjaya Setiawan (2311104053)

SE-07-02

Dosen :

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

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

PURWOKERTO

2024

1. **Tujuan**
    - Memahami penggunaan Multi Linked List.
    - Mengimplementasikan Multi Linked List dalam studi kasus.
2. **Landasan Teori**

    Multi Linked List

    Multi Linked List adalah kumpulan list yang saling terhubung. Terdiri dari list induk dan list anak. Contoh: List pegawai (induk) memiliki elemen anak berupa list anak-anaknya.

    Operasi Utama:

    1. Insert Anak
        - Menambahkan elemen anak pada list anak yang terkait dengan list induk tertentu.
        - Langkah: Cari elemen induk, lalu tambahkan elemen anak di posisi terakhir.
    2. Delete Anak
        - Menghapus elemen anak dengan memastikan induknya.
        - Jika elemen induk dihapus, seluruh anaknya juga dihapus.
    3. Insert Induk dan Delete Induk
        - Sama dengan operasi pada single, double, atau circular linked list.

    Struktur Data yang Digunakan:

    1. Elemen List Anak : terdiri dari info, next, dan prev.
    2. List Anak : menyimpan pointer ke elemen pertama (first) dan terakhir (last).
    3. Elemen List Induk : terdiri dari info, list anak (lanak), next, dan prev.
    4. List Induk : menyimpan pointer ke elemen pertama (first) dan terakhir (last).

3. Guided

```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;
}
```

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

Cara Kerja:

1. Node: Menyimpan data, pointer ke node berikutnya (next), dan pointer ke anak node (child).
2. MultiLinkedList: Mengelola linked list dengan pointer ke head node (head).
3. addParent: Menambahkan node parent di awal linked list.
4. addChild: Mencari parent berdasarkan data, lalu menambahkan node anak di awal daftar anak.
5. display: Menampilkan semua node parent dan anak-anaknya.
6. Destruktor: Menghapus semua node dari memori.
7. main: Membuat objek MultiLinkedList, menambahkan beberapa parent dan anak, lalu menampilkan hasilnya.Fungsi Utama: Inisialisasi pohon, membuat node root dan anak-anaknya, melakukan traversal, menampilkan node kiri/kanan, dan menghapus node tertentu.

Telkom University Purwokerto

```cpp
#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;
}

```

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

Cara Kerja:

1. Node: Menyimpan nama karyawan, pointer ke node berikutnya (next), dan pointer ke node anak (subordinate).
2. EmployeeList: Mengelola linked list dengan pointer ke head node (head).
3. addEmployee: Menambahkan node karyawan di awal linked list.
4. addSubordinate: Mencari manajer berdasarkan nama, lalu menambahkan node anak di awal daftar anak.
5. display: Menampilkan semua node manajer dan anak-anaknya.
6. Destruktor: Menghapus semua node dari memori.
7. main: Membuat objek EmployeeList, menambahkan beberapa karyawan dan anak, lalu menampilkan hasilnya.

Telkom University Purwokerto

```cpp
#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

    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;
}
```

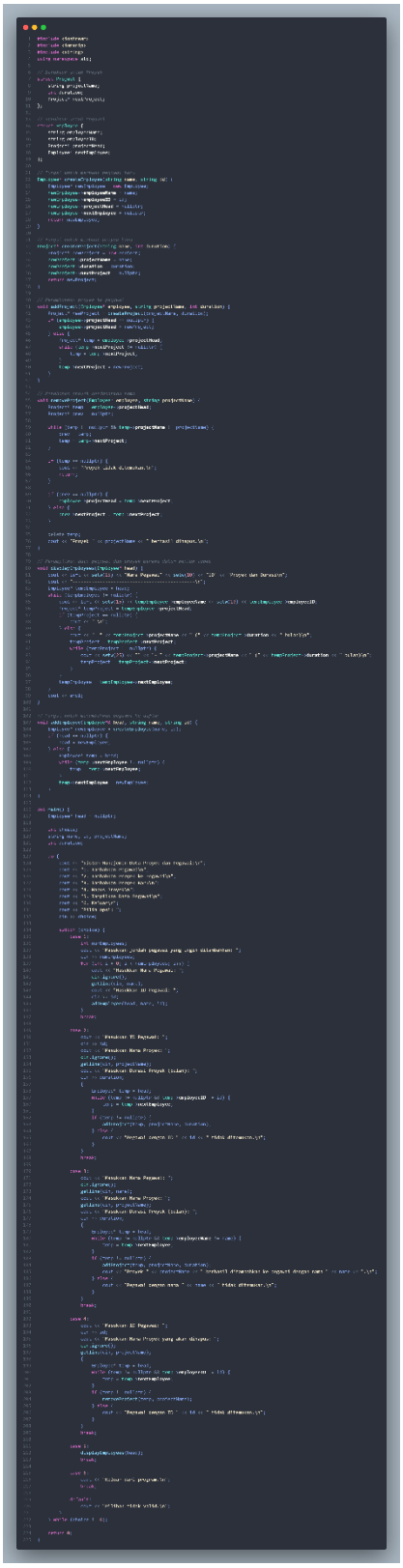
```
Initial employee list:
Manager: Charlie ->
Manager: Bob -> Frank
Manager: Alice -> Eve David
Subordinate David deleted from Alice.
Employee Charlie deleted.

Updated employee list:
Manager: Bob -> Frank
Manager: Alice -> Eve
```

Cara Kerja:

1. Node: Menyimpan nama karyawan, pointer ke node berikutnya (next), dan pointer ke anak node (subordinate).
2. EmployeeList: Mengelola linked list dengan pointer ke head node (head).
3. addEmployee: Menambahkan node karyawan di awal linked list.
4. addSubordinate: Mencari manajer berdasarkan nama, lalu menambahkan node anak di awal daftar anak.
5. deleteEmployee: Menghapus karyawan dan semua anaknya dari linked list.
6. deleteSubordinate: Menghapus anak karyawan tertentu dari linked list.
7. display: Menampilkan semua karyawan dan anak-anak mereka.
8. Destruktor: Menghapus semua node dari memori.
9. main: Membuat objek EmployeeList, menambahkan karyawan dan anak, menghapus beberapa, lalu menampilkan hasilnya.

Telkom University Purwokerto

4. Unguided



Telkom University Purwokerto

```
Sistem Manajemen Data Pegawai dan Proyek:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 1
Masukkan jumlah pegawai yang ingin ditambahkan: 3
Masukkan Nama Pegawai: Andi
Masukkan ID Pegawai: P001
Masukkan Nama Pegawai: Budi
Masukkan ID Pegawai: P002
Masukkan Nama Pegawai: Citra
Masukkan ID Pegawai: P003
Sistem Manajemen Data Pegawai dan Proyek:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 5
Nama Pegawai   ID        Proyek dan Durasi
-----------------------------------------
Andi           P001      -
Budi           P002      -
Citra          P003      -
```

```
Sistem Manajemen Data Pegawai dan Proyek:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 2
Masukkan ID Pegawai: P001
Masukkan Nama Proyek: Aplikasi Mobile
Masukkan Durasi Proyek (bulan): 12
Sistem Manajemen Data Pegawai dan Proyek:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 2
Masukkan ID Pegawai: P002
Masukkan Nama Proyek: Sistem Akuntansi
Masukkan Durasi Proyek (bulan): 8
Sistem Manajemen Data Pegawai dan Proyek:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 2
Masukkan ID Pegawai: P003
Masukkan Nama Proyek: E-Commerence
Masukkan Durasi Proyek (bulan): 10
Sistem Manajemen Data Pegawai dan Proyek:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 5
Nama Pegawai   ID        Proyek dan Durasi
-----------------------------------------
Andi           P001      - Aplikasi Mobile (12 bulan)
Budi           P002      - Sistem Akuntansi (8 bulan)
Citra          P003      - E-Commerence (10 bulan)
```

```
Sistem Manajemen Data Proyek dan Pegawai:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 3
Masukkan Nama Pegawai: Andi
Masukkan Nama Proyek: Analisis Data
Masukkan Durasi Proyek (bulan): 6
Proyek Analisis Data berhasil ditambahkan ke pegawai dengan nama Andi.
Sistem Manajemen Data Proyek dan Pegawai:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 5
Nama Pegawai   ID        Proyek dan Durasi
---------------------------------------------
Andi           P001      - Aplikasi Mobile (12 bulan)
                         - Analisis Data (6 bulan)
Budi           P002      - Sistem Akuntansi (8 bulan)
Citra          P003      - E-commerce (10 bulan)
```

```
Sistem Manajemen Data Proyek dan Pegawai:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 4
Masukkan ID Pegawai: P001
Masukkan Nama Proyek yang akan dihapus: Analisis Data
Proyek Analisis Data berhasil dihapus.
Sistem Manajemen Data Proyek dan Pegawai:
1. Tambahkan Pegawai
2. Tambahkan Proyek ke Pegawai
3. Tambahkan Proyek Baru
4. Hapus Proyek
5. Tampilkan Data Pegawai
6. Keluar
Pilih opsi: 5
Nama Pegawai   ID        Proyek dan Durasi
---------------------------------------------
Andi           P001      - Aplikasi Mobile (12 bulan)
Budi           P002      - Sistem Akuntansi (8 bulan)
Citra          P003      - E-commerce (10 bulan)
```

Cara Kerja:

1. Node: Menyimpan nama proyek, durasi, dan pointer ke proyek berikutnya (nextProject).

2. Employee: Menyimpan nama pegawai, ID, pointer ke proyek pertama (projectHead), dan pointer ke pegawai berikutnya (nextEmployee).

3. createEmployee: Membuat pegawai baru dengan nama dan ID yang diberikan.

4. createProject: Membuat proyek baru dengan nama dan durasi yang diberikan.
5. addEmployee: Menambahkan pegawai ke daftar pegawai (head).
6. addProject: Menambahkan proyek ke pegawai yang ditentukan.
7. removeProject: Menghapus proyek dari pegawai berdasarkan nama proyek.
8. displayEmployees: Menampilkan data pegawai dan proyek mereka dalam bentuk tabel.
9. main: Mengelola input pengguna untuk menambah pegawai, menambah proyek, menghapus proyek, dan menampilkan data pegawai.

Telkom University Purwokerto

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

// Struktur untuk Buku
struct Book {
    string title;
    string returnDate;
    Book* nextBook;
};

// Struktur untuk Anggota
struct Member {
    string memberName;
    string memberID;
    Book* bookHead;
    Member* nextMember;
};

// Fungsi untuk membuat anggota baru
Member* createMember(string name, string id) {
    Member* newMember = new Member;
    newMember->memberName = name;
    newMember->memberID = id;
    newMember->bookHead = nullptr;
    newMember->nextMember = nullptr;
    return newMember;
}

// Fungsi untuk membuat buku baru
Book* createBook(string title, string returnDate) {
    Book* newBook = new Book;
    newBook->title = title;
    newBook->returnDate = returnDate;
    newBook->nextBook = nullptr;
    return newBook;
}

// Menambahkan buku ke anggota
void addBook(Member* member, string title, string returnDate) {
    Book* newBook = createBook(title, returnDate);
    if (member->bookHead == nullptr) {
        member->bookHead = newBook;
    } else {
        Book* temp = member->bookHead;
        while (temp->nextBook != nullptr) {
            temp = temp->nextBook;
        }
        temp->nextBook = newBook;
    }
}

// Menghapus anggota beserta buku yang dipinjam
void removeMember(Member*& head, string id) {
    Member* temp = head;
    Member* prev = nullptr;

    while (temp != nullptr && temp->memberID != id) {
        prev = temp;
        temp = temp->nextMember;
    }

    if (temp == nullptr) {
        cout << "Anggota dengan ID " << id << " tidak ditemukan.\n";
        return;
    }

    if (prev == nullptr) {
        head = temp->nextMember;
    } else {
        prev->nextMember = temp->nextMember;
    }

    // Hapus semua buku yang dipinjam anggota
    Book* bookTemp = temp->bookHead;
    while (bookTemp != nullptr) {
        Book* toDelete = bookTemp;
        bookTemp = bookTemp->nextBook;
        delete toDelete;
    }

    delete temp;
    cout << "Anggota dengan ID " << id << " beserta buku yang dipinjam berhasil dihapus.\n";
}

// Menampilkan data anggota dan buku yang dipinjam
void displayMembers(Member* head) {
    Member* tempMember = head;
    int bookNumber = 1;
    while (tempMember != nullptr) {
        Book* tempBook = tempMember->bookHead;
        while (tempBook != nullptr) {
            cout << "- Buku " << bookNumber << ": Judul = \"" << tempBook->title << "\", Pengembalian = \"" << tempBook->returnDate << "\" (Untuk " << tempMember->memberName << ").\n";
            tempBook = tempBook->nextBook;
            bookNumber++;
        }
        tempMember = tempMember->nextMember;
    }
}

int main() {
    Member* head = nullptr;

    // Menambahkan data anggota
    head = createMember("Rani", "A001");
    head->nextMember = createMember("Dito", "A002");
    head->nextMember->nextMember = createMember("Vina", "A003");

    cout << "Data Awal Anggota Perpustakaan:\n\n";
    cout << "Anggota 1: Nama = \"Rani\", ID = \"A001\".\n";
    cout << "Anggota 2: Nama = \"Dito\", ID = \"A002\".\n";
    cout << "Anggota 3: Nama = \"Vina\", ID = \"A003\".\n\n";

    cout << "Setelah menambahkan buku yang dipinjam:\n\n";

    // Menambahkan buku yang dipinjam
    addBook(head, "Pemrograman C++", "01/12/2024");
    addBook(head->nextMember, "Algoritma Pemrograman", "15/12/2024");
    displayMembers(head);

    // Menambahkan buku baru untuk Rani
    cout << "\nSetelah menambahkan buku baru:\n\n";
    addBook(head, "Struktur Data", "10/12/2024");
    displayMembers(head);

    // Menghapus anggota Dito beserta buku yang dipinjam
    cout << "\nSetelah menghapus anggota Dito:\n\n";
    removeMember(head, "A002");
    displayMembers(head);

    return 0;
}
```

```
Data Awal Anggota Perpustakaan:

Anggota 1: Nama = "Rani", ID = "A001".
Anggota 2: Nama = "Dito", ID = "A002".
Anggota 3: Nama = "Vina", ID = "A003".

Setelah menambahkan 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)

Setelah menambahkan buku baru:

- Buku 1: Judul = "Pemrograman C++", Pengembalian = "01/12/2024" (Untuk Rani).
- Buku 2: Judul = "Struktur Data", Pengembalian = "10/12/2024" (Untuk Rani).
- Buku 3: Judul = "Algoritma Pemrograman", Pengembalian = "15/12/2024" (Untuk Dito)

Setelah menghapus anggota Dito:

Anggota dengan ID A002 beserta buku yang dipinjam berhasil dihapus.
- Buku 1: Judul = "Pemrograman C++", Pengembalian = "01/12/2024" (Untuk Rani).
- Buku 2: Judul = "Struktur Data", Pengembalian = "10/12/2024" (Untuk Rani).
```

Cara Kerja:

1. Node: Menyimpan nama proyek, durasi, dan pointer ke proyek berikutnya (nextProject).
2. Employee: Menyimpan nama pegawai, ID, pointer ke proyek pertama (projectHead), dan pointer ke pegawai berikutnya (nextEmployee).
3. createEmployee: Membuat pegawai baru dengan nama dan ID yang diberikan.
4. createProject: Membuat proyek baru dengan nama dan durasi yang diberikan.
5. addEmployee: Menambahkan pegawai ke daftar pegawai (head).
6. addProject: Menambahkan proyek ke pegawai yang ditentukan.
7. removeProject: Menghapus proyek dari pegawai berdasarkan nama proyek.
8. displayEmployees: Menampilkan data pegawai dan proyek mereka dalam bentuk tabel.
9. main: Mengelola input pengguna untuk menambah pegawai, menambah proyek, menghapus proyek, dan menampilkan data pegawai.

5. **Kesimpulan**

Multi Linked List mengelola data hierarkis secara efektif dengan operasi penambahan, penghapusan, dan penampilan data induk dan anak. Ini memudahkan manajemen data terstruktur dan traversal elemen yang terhubung. Destruktor membersihkan memori untuk mencegah kebocoran. Implementasi ini cocok untuk berbagai studi kasus.