LAPORAN PRAKTIKUM

PERTEMUAN 13

13_MULTILINKLISTED

Nama :

Ilham Lii Assidaq (2311104068)

Dosen :

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

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

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

II. TOOL

Dev C++

III. DASAR TEORI

Struktur data "Multi Linked List" sangat bermanfaat untuk representasi hierarkis, seperti organisasi data karyawan beserta sub-data lainnya. Konsep ini memungkinkan berbagai daftar (list) saling terhubung, di mana elemen dalam satu daftar dapat menjadi induk bagi elemen dalam daftar lain. Operasi dasar pada Multi Linked List meliputi penambahan (insert) dan penghapusan (delete) elemen pada daftar induk maupun anak-anak, serta fungsi-fungsi tambahan lainnya. Untuk mengelola struktur yang dinamis ini dengan efektif, implementasinya sering kali menggunakan pointer.

IV. GUIDED

1. GUIDED 1

```
#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;
}
```

2. GUIDED2

```
#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;
}
```

3. GUIDED 3

```
#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;
}
```

V. UNGUIDED

1. UNGUIDED1

```
#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 = NULL;

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

void tambahProyek(string idPegawai, string namaProyek, int durasi) {
    Pegawai* currPegawai = headPegawai;
    while (currPegawai != NULL) {
        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 != NULL) {
```

```
        if (currPegawai->idPegawai == idPegawai) {
            Proyek* currProyek = currPegawai->headProyek;
            Proyek* prevProyek = NULL;
            while (currProyek != NULL) {
                if (currProyek->namaProyek == namaProyek) {
                    if (prevProyek == NULL) {
                        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 != NULL) {
        cout << "Pegawai : " << currPegawai->namaPegawai << " (ID: " << currPegawai->idPegawai << ")\n";
        Proyek* currProyek = currPegawai->headProyek;
        while (currProyek != NULL) {
            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;
  }

```

UNGUIDED 2

```
#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);

    return 0;
}
```

VI. KESIMPULAN

Kesimpulan yang saya dapatkan adalah bahwa struktur data ini memungkinkan pengelolaan data yang kompleks dengan hubungan hierarkis antara elemen-elemen dalam daftar 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 memerlukan pemahaman yang mendalam tentang manipulasi pointer dan struktur dinamis.