LAPORAN PRAKTIKUM STRUKTUR DATA
MODUL 13
MULTI LINKED LIST

Disusun Oleh:

Dwi Candra Pratama 2211104035/SE07-02

Asisten Praktikum:

Aldi Putra Hamalsyah

Andini Nur Hidayah

Dosen Pengampu:

Wahyu Andi Saputra

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA TELKOM UNIVERSITY PURWOKERTO

2024

A. Guided

TUJUAN PRAKTIKUM

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

Multi Linked List

Multi List merupakan sekumpulan list yang berbeda yang memiliki suatu keterhubungan satu sama lain. Tiap elemen dalam multi link list dapat membentuk list sendiri.Biasanya ada yang bersifat sebagai list induk dan list anak.

Operasi Dasar

1. Insert (Penambahan)

A. Insert Anak

Tujuan: Menambahkan elemen baru ke dalam list anak tertentu.

Langkah Utama:

1. Tentukan elemen induk tempat elemen anak akan ditambahkan.
2. Tambahkan elemen anak ke akhir list anak terkait.

Contoh code:

```
      void insertLastAnak(listanak &Lanak, address_anak
      P) {
          if (!Lanak.first) {
              Lanak.first = P;
              Lanak.last = P;
          } else {
              Lanak.last->next = P;
              P->prev = Lanak.last;
              Lanak.last = P;
          }
}
```

B. Insert Induk

Mirip dengan operasi insert pada Single, Double, atau Circular Linked List.
Elemen baru ditambahkan di awal, tengah, atau akhir list induk sesuai kebutuhan.

2. Delete (Penghapusan)

A. Delete Anak

Tujuan: Menghapus elemen anak tertentu dari list anak yang dimiliki oleh elemen induk tertentu.

Langkah Utama:

1. Cari elemen anak yang sesuai dengan kriteria.
2. Hapus elemen tersebut dari list anak.

Contoh Kode:

```
      void delLastAnak(listanak &Lanak, address_anak &P)
      {
          if (!Lanak.first) return; // Jika list kosong
          P = Lanak.last;
          if (Lanak.first == Lanak.last) { // Jika hanya
      ada satu elemen
              Lanak.first = nullptr;
              Lanak.last = nullptr;
          } else {
              Lanak.last = P->prev;
              Lanak.last->next = nullptr;
          }
          delete P;
}
```

B. Delete Induk
Tujuan: Menghapus elemen induk dan seluruh list anak yang dimilikinya.
Langkah Utama:

1. Hapus semua elemen anak dari list anak terkait.
2. Hapus elemen induk dari list induk.

3. Traversal (Penelusuran)

Traversal dilakukan untuk menampilkan semua elemen di dalam list, baik elemen induk maupun anak.
Langkah Utama:

1. Iterasi melalui setiap elemen induk.
2. Untuk setiap elemen induk, iterasi melalui list anaknya.

Contoh code:

```
      void printInfo(listinduk L) {
          address P = L.first;
          while (P != nullptr) {
              cout << "Info Induk: " << P->info << endl;
              printInfoAnak(P->lanak); // Tampilkan
      elemen anak
              P = P->next;
          }
}
```

B. 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.

SourceCode:

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

// Struktur untuk proyek
struct Project {
    string projectName;
    int duration; // dalam bulan
    Project* next;

    Project(string name, int dur) : projectName(name), duration(dur), next(nullptr) {}
};

// Struktur untuk pegawai
struct Employee {
    string name;
    string id;
    Project* projectHead; // Head pointer untuk proyek
    Employee* next;

    Employee(string empName, string empId) : name(empName), id(empId),
projectHead(nullptr), next(nullptr) {}
};

// Tambahkan pegawai ke linked list pegawai
void addEmployee(Employee*& head, string name, string id) {
    Employee* newEmployee = new Employee(name, id);
    if (!head) {
        head = newEmployee;
        return;
    }
    Employee* temp = head;
    while (temp->next) {
        temp = temp->next;
    }
    temp->next = newEmployee;
```

```
}

// Tambahkan proyek ke pegawai tertentu
void addProject(Employee* employee, string projectName, int duration) {
    if (!employee) return;
    Project* newProject = new Project(projectName, duration);
    if (!employee->projectHead) {
        employee->projectHead = newProject;
        return;
    }
    Project* temp = employee->projectHead;
    while (temp->next) {
        temp = temp->next;
    }
    temp->next = newProject;
}

// Hapus proyek dari pegawai tertentu
void removeProject(Employee* employee, string projectName) {
    if (!employee || !employee->projectHead) return;

    Project* temp = employee->projectHead;
    Project* prev = nullptr;

    // Jika proyek pertama yang harus dihapus
    if (temp->projectName == projectName) {
        employee->projectHead = temp->next;
        delete temp;
        return;
    }

    // Cari proyek yang sesuai
    while (temp && temp->projectName != projectName) {
        prev = temp;
        temp = temp->next;
    }

    // Jika proyek ditemukan
    if (temp) {
        prev->next = temp->next;
        delete temp;
    }
}

// Tampilkan data pegawai dan proyek
void displayEmployees(Employee* head) {
    Employee* temp = head;
    while (temp) {
        cout << "Pegawai: " << temp->name << " (ID: " << temp->id << ")\n";
        Project* projectTemp = temp->projectHead;
```

```
        while (projectTemp) {
            cout << "  - Proyek: " << projectTemp->projectName << ", Durasi: " <<
projectTemp->duration << " bulan\n";
            projectTemp = projectTemp->next;
        }
        temp = temp->next;
        cout << endl;
    }
}

int main() {
    Employee* employeeHead = nullptr;

    // 1. Masukkan data pegawai
    addEmployee(employeeHead, "Andi", "P001");
    addEmployee(employeeHead, "Budi", "P002");
    addEmployee(employeeHead, "Citra", "P003");

    // 2. Tambahkan proyek ke pegawai
    addProject(employeeHead, "Aplikasi Mobile", 12); // Proyek untuk Andi
    addProject(employeeHead->next, "Sistem Akuntansi", 8); // Proyek untuk Budi
    addProject(employeeHead->next->next, "E-commerce", 10); // Proyek untuk Citra

    // 3. Tambahkan proyek baru
    addProject(employeeHead, "Analisis Data", 6); // Proyek baru untuk Andi

    // 4. Hapus proyek "Aplikasi Mobile" dari Andi
    removeProject(employeeHead, "Aplikasi Mobile");

    // 5. Tampilkan data pegawai dan proyek mereka
    displayEmployees(employeeHead);

    return 0;
}
```

OutPut

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 10\UNguided\output> & .\'Unguided1.exe'
Pegawai: Andi (ID: P001)
  - Proyek: Analisis Data, Durasi: 6 bulan

Pegawai: Budi (ID: P002)
  - Proyek: Sistem Akuntansi, Durasi: 8 bulan

Pegawai: Citra (ID: P003)
  - Proyek: E-commerce, Durasi: 10 bulan

PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 10\UNguided\output>
```

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

Source Code:

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

// Struktur untuk buku
struct Book {
    string title;
    string returnDate; // Tanggal pengembalian
    Book* next;

    Book(string t, string rd) : title(t), returnDate(rd), next(nullptr) {}
};

// Struktur untuk anggota
struct Member {
    string name;
    string id;
    Book* bookHead; // Head pointer untuk buku
    Member* next;

    Member(string mName, string mId) : name(mName), id(mId), bookHead(nullptr),
next(nullptr) {}
};

// Tambahkan anggota ke linked list anggota
void addMember(Member*& head, string name, string id) {
    Member* newMember = new Member(name, id);
    if (!head) {
        head = newMember;
        return;
    }
    Member* temp = head;
    while (temp->next) {
```

```
        temp = temp->next;
    }
    temp->next = newMember;
}

// Tambahkan buku ke anggota tertentu
void addBook(Member* member, string title, string returnDate) {
    if (!member) return;
    Book* newBook = new Book(title, returnDate);
    if (!member->bookHead) {
        member->bookHead = newBook;
        return;
    }
    Book* temp = member->bookHead;
    while (temp->next) {
        temp = temp->next;
    }
    temp->next = newBook;
}

// Hapus anggota beserta buku yang dipinjam
void removeMember(Member*& head, string memberId) {
    if (!head) return;

    Member* temp = head;
    Member* prev = nullptr;

    // Jika anggota pertama yang harus dihapus
    if (temp->id == memberId) {
        head = temp->next;

        // Hapus semua buku yang dipinjam anggota
        while (temp->bookHead) {
            Book* bookTemp = temp->bookHead;
            temp->bookHead = bookTemp->next;
            delete bookTemp;
        }
        delete temp;
        return;
    }

    // Cari anggota yang sesuai
    while (temp && temp->id != memberId) {
        prev = temp;
        temp = temp->next;
    }

    // Jika anggota ditemukan
    if (temp) {
        prev->next = temp->next;
```

```
        // Hapus semua buku yang dipinjam anggota
        while (temp->bookHead) {
            Book* bookTemp = temp->bookHead;
            temp->bookHead = bookTemp->next;
            delete bookTemp;
        }
        delete temp;
    }
}

// Tampilkan data anggota dan buku yang dipinjam
void displayMembers(Member* head) {
    Member* temp = head;
    while (temp) {
        cout << "Anggota: " << temp->name << " (ID: " << temp->id << ")\n";
        Book* bookTemp = temp->bookHead;
        while (bookTemp) {
            cout << "  - Buku: " << bookTemp->title << ", Tanggal Pengembalian: " <<
bookTemp->returnDate << "\n";
            bookTemp = bookTemp->next;
        }
        temp = temp->next;
        cout << endl;
    }
}

int main() {
    Member* memberHead = nullptr;

    // 1. Masukkan data anggota
    addMember(memberHead, "Rani", "A001");
    addMember(memberHead, "Dito", "A002");
    addMember(memberHead, "Vina", "A003");

    // 2. Tambahkan buku yang dipinjam
    addBook(memberHead, "Pemrograman C++", "01/12/2024"); // Buku untuk Rani
    addBook(memberHead->next, "Algoritma Pemrograman", "15/12/2024"); // Buku
untuk Dito

    // 3. Tambahkan buku baru
    addBook(memberHead, "Struktur Data", "10/12/2024"); // Buku baru untuk Rani

    // 4. Hapus anggota Dito beserta buku yang dipinjam
    removeMember(memberHead, "A002");

    // 5. Tampilkan seluruh data anggota dan buku yang dipinjam
    displayMembers(memberHead);

    return 0;
```

```
}
```

OutPut

```
PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 10\UNguided\output> & .\'Unguided2.exe'
Anggota: Rani (ID: A001)
  - Buku: Pemrograman C++, Tanggal Pengembalian: 01/12/2024
  - Buku: Struktur Data, Tanggal Pengembalian: 10/12/2024

Anggota: Vina (ID: A003)

PS D:\Semester5\StrukturData\PraktikumStrukturData\Pertemuan 10\UNguided\output>
```