LAPORAN PRAKTIKUM
MODUL 13
MULTI LINKED LIST

Nama :

Muhammad Rifqi Al Baqi Ananta (2311104005)

Dosen :

Yudha Islami Sulistya, S.Kom., M.Cs.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSTY PURWOKERTO

2024

I. GUIDED

1. Soal 1

Code:

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

// Kelas untuk menyimpan data proyek
class Proyek {
public:
    string nama_proyek;
    int durasi;
    Proyek* next;

    Proyek(string nama, int durasi) {
        this->nama_proyek = nama;
        this->durasi = durasi;
        this->next = nullptr;
    }
};

// Kelas untuk menyimpan data pegawai
class Pegawai {
public:
    string nama;
    string id_pegawai;
    Proyek* proyek_head;
    Pegawai* next;

    Pegawai(string nama, string id_pegawai) {
        this->nama = nama;
        this->id_pegawai = id_pegawai;
        this->proyek_head = nullptr;
        this->next = nullptr;
    }
};

// Kelas untuk manajemen data pegawai dan proyek
class ManajemenData {
public:
    Pegawai* head;

    ManajemenData() {
        head = nullptr;
    }

    // Menambahkan pegawai baru ke dalam list
    void tambah_pegawai(string nama, string id_pegawai) {
        Pegawai* new_pegawai = new Pegawai(nama, id_pegawai);
        if (!head) {
            head = new_pegawai;
        } else {
            Pegawai* temp = head;
            while (temp->next) {
                temp = temp->next;
            }
            temp->next = new_pegawai;
        }
    }

    // Menambahkan proyek baru untuk pegawai tertentu
    void tambah_proyek(string id_pegawai, string nama_proyek, int durasi) {
        Pegawai* temp = head;
        while (temp) {
            if (temp->id_pegawai == id_pegawai) {
                Proyek* new_proyek = new Proyek(nama_proyek, durasi);
                if (!temp->proyek_head) {
                    temp->proyek_head = new_proyek;
                } else {
                    Proyek* proyek_temp = temp->proyek_head;
                    while (proyek_temp->next) {
                        proyek_temp = proyek_temp->next;
                    }
                    proyek_temp->next = new_proyek;
                }
                return;
            }
            temp = temp->next;
        }
        cout << "Pegawai dengan ID " << id_pegawai << " tidak ditemukan.\n";
    }

    // Menghapus proyek berdasarkan nama proyek untuk pegawai tertentu
    void hapus_proyek(string id_pegawai, string nama_proyek) {
        Pegawai* temp = head;
        while (temp) {
            if (temp->id_pegawai == id_pegawai) {
                Proyek* proyek_temp = temp->proyek_head;
                Proyek* prev = nullptr;
                while (proyek_temp) {
                    if (proyek_temp->nama_proyek == nama_proyek) {
                        if (prev) {
                            prev->next = proyek_temp->next;
                        } else {
                            temp->proyek_head = proyek_temp->next;
                        }
                        delete proyek_temp;
                        return;
                    }
                    prev = proyek_temp;
                    proyek_temp = proyek_temp->next;
                }
                cout << "Proyek dengan nama " << nama_proyek << " tidak ditemukan.\n";
                return;
            }
            temp = temp->next;
        }
        cout << "Pegawai dengan ID " << id_pegawai << " tidak ditemukan.\n";
    }

    // Menampilkan data pegawai dan proyek mereka
    void tampilkan_data() {
        Pegawai* temp = head;
        while (temp) {
            cout << "Pegawai: " << temp->nama << " (ID: " << temp->id_pegawai << ")\n";
            Proyek* proyek_temp = temp->proyek_head;
            if (proyek_temp) {
                cout << "Proyek yang dikelola:\n";
                while (proyek_temp) {
                    cout << "  - " << proyek_temp->nama_proyek << " (Durasi: " << proyek_temp->durasi << " bulan)\n";
                    proyek_temp = proyek_temp->next;
                }
            } else {
                cout << "  Tidak ada proyek.\n";
            }
            cout << endl;
            temp = temp->next;
        }
    }
};

// Program utama
int main() {
    ManajemenData manajemen;

    // 1. Menambahkan pegawai
    manajemen.tambah_pegawai("Andi", "P001");
    manajemen.tambah_pegawai("Budi", "P002");
    manajemen.tambah_pegawai("Citra", "P003");

    // 2. Menambahkan proyek
    manajemen.tambah_proyek("P001", "Aplikasi Mobile", 12);
    manajemen.tambah_proyek("P002", "Sistem Akuntansi", 8);
    manajemen.tambah_proyek("P003", "E-commerce", 10);

    // 3. Menambahkan proyek baru
    manajemen.tambah_proyek("P001", "Analisis Data", 6);

    // 4. Menghapus proyek "Aplikasi Mobile" dari Andi
    manajemen.hapus_proyek("P001", "Aplikasi Mobile");

    // 5. Menampilkan data pegawai dan proyek mereka
    manajemen.tampilkan_data();

    return 0;
}
```

Output:

```
Pegawai: Andi (ID: P001)
Proyek yang dikelola:
  - Analisis Data (Durasi: 6 bulan)

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

Pegawai: Citra (ID: P003)
Proyek yang dikelola:
  - E-commerce (Durasi: 10 bulan)
```

2. Soal 2
   Code:

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

// Kelas untuk menyimpan data buku
class Buku {
public:
    string judul_buku;
    string tanggal_pengembalian;
    Buku* next;

    Buku(string judul, string tanggal) {
        this->judul_buku = judul;
        this->tanggal_pengembalian = tanggal;
        this->next = nullptr;
    }
};

// Kelas untuk menyimpan data anggota
class Anggota {
public:
    string nama;
    string id_anggota;
    Buku* buku_head;
    Anggota* next;

    Anggota(string nama, string id_anggota) {
        this->nama = nama;
        this->id_anggota = id_anggota;
        this->buku_head = nullptr;
        this->next = nullptr;
    }
};

// Kelas untuk manajemen data anggota dan buku
class ManajemenPerpustakaan {
public:
    Anggota* head;

    ManajemenPerpustakaan() {
        head = nullptr;
    }

    // Menambahkan anggota baru ke dalam list
    void tambah_anggota(string nama, string id_anggota) {
        Anggota* new_anggota = new Anggota(nama, id_anggota);
        if (!head) {
            head = new_anggota;
        } else {
            Anggota* temp = head;
            while (temp->next) {
                temp = temp->next;
            }
            temp->next = new_anggota;
        }
    }

    // Menambahkan buku yang dipinjam oleh anggota tertentu
    void tambah_buku(string id_anggota, string judul_buku, string tanggal_pengembalian) {
        Anggota* temp = head;
        while (temp) {
            if (temp->id_anggota == id_anggota) {
                Buku* new_buku = new Buku(judul_buku, tanggal_pengembalian);
                if (!temp->buku_head) {
                    temp->buku_head = new_buku;
                } else {
                    Buku* buku_temp = temp->buku_head;
                    while (buku_temp->next) {
                        buku_temp = buku_temp->next;
                    }
                    buku_temp->next = new_buku;
                }
                return;
            }
            temp = temp->next;
        }
        cout << "Anggota dengan ID " << id_anggota << " tidak ditemukan.\n";
    }

    // Menghapus anggota beserta buku yang dipinjam
    void hapus_anggota(string id_anggota) {
        Anggota* temp = head;
        Anggota* prev = nullptr;

        while (temp) {
            if (temp->id_anggota == id_anggota) {
                if (prev) {
                    prev->next = temp->next;
                } else {
                    head = temp->next;
                }

                // Menghapus semua buku yang dipinjam oleh anggota ini
                Buku* buku_temp = temp->buku_head;
                while (buku_temp) {
                    Buku* buku_next = buku_temp->next;
                    delete buku_temp;
                    buku_temp = buku_next;
                }

                delete temp;
                return;
            }
            prev = temp;
            temp = temp->next;
        }
        cout << "Anggota dengan ID " << id_anggota << " tidak ditemukan.\n";
    }

    // Menampilkan seluruh data anggota dan buku yang dipinjam
    void tampilkan_data() {
        Anggota* temp = head;
        while (temp) {
            cout << "Anggota: " << temp->nama << " (ID: " << temp->id_anggota << ")\n";
            Buku* buku_temp = temp->buku_head;
            if (buku_temp) {
                cout << "Buku yang dipinjam:\n";
                while (buku_temp) {
                    cout << "  - " << buku_temp->judul_buku << " (Pengembalian: " << buku_temp->tanggal_pengembalian << ")\n";
                    buku_temp = buku_temp->next;
                }
            } else {
                cout << "  Tidak ada buku yang dipinjam.\n";
            }
            cout << endl;
            temp = temp->next;
        }
    }
};

// Program utama
int main() {
    ManajemenPerpustakaan perpustakaan;

    // 1. Menambahkan anggota
    perpustakaan.tambah_anggota("Rani", "A001");
    perpustakaan.tambah_anggota("Dito", "A002");
    perpustakaan.tambah_anggota("Vina", "A003");

    // 2. Menambahkan buku yang dipinjam
    perpustakaan.tambah_buku("A001", "Pemrograman C++", "01/12/2024");
    perpustakaan.tambah_buku("A002", "Algoritma Pemrograman", "15/12/2024");

    // 3. Menambahkan buku baru
    perpustakaan.tambah_buku("A001", "Struktur Data", "10/12/2024");

    // 4. Menghapus anggota Dito beserta buku yang dipinjam
    perpustakaan.hapus_anggota("A002");

    // 5. Menampilkan seluruh data anggota dan buku yang dipinjam
    perpustakaan.tampilkan_data();

    return 0;
}
```

Output:

```
Anggota: Rani (ID: A001)
Buku yang dipinjam:
  - Pemrograman C++ (Pengembalian: 01/12/2024)
  - Struktur Data (Pengembalian: 10/12/2024)

Anggota: Vina (ID: A003)
  Tidak ada buku yang dipinjam.
```

II. PENJELASAN

Pada praktikum ini, kami mengimplementasikan Multi Linked List untuk membangun sistem manajemen buku perpustakaan yang dapat menyimpan data anggota dan buku yang dipinjam. Dalam sistem ini, setiap anggota perpustakaan memiliki data berupa nama dan ID anggota, yang terhubung dengan linked list buku yang mereka pinjam. Setiap buku berisi informasi mengenai judul dan tanggal pengembalian, serta terhubung dengan anggota yang meminjamnya. Penggunaan Multi Linked List memungkinkan fleksibilitas dalam mengelola data, seperti menambah atau menghapus anggota dan buku secara efisien. Selain itu, struktur ini memungkinkan relasi yang jelas antara anggota dan buku yang dipinjam, mempermudah pengelolaan serta pemeliharaan data anggota perpustakaan dan koleksi bukunya. Implementasi ini berhasil menunjukkan bagaimana linked list dapat digunakan untuk merepresentasikan data yang saling terkait dengan cara yang efisien dan mudah dikelola.