LAPORAN PRAKTIKUM

PERTEMUAN 13

MULTI LINKED LIST

Nama :

Salman Alfarisi (2311104036)

Dosen :

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

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

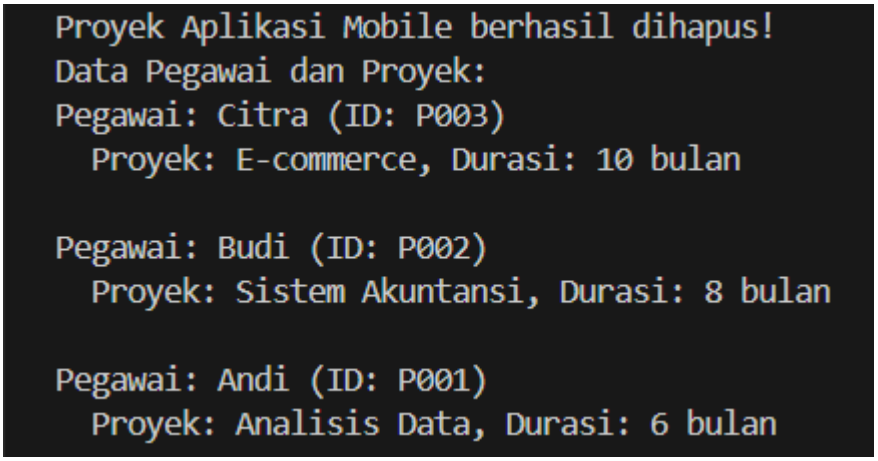
2024

1. LATIHAN 1

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


struct Proyek {
    string nama_proyek;
    int durasi;
};


struct Pegawai {
    string nama_pegawai;
    string id_pegawai;
};


struct ElmProyek {
    Proyek info;
    ElmProyek* next;
};


struct ElmPegawai {
    Pegawai info;
    ElmPegawai* next;
    ElmProyek* firstProyek;
};


struct List {
    ElmPegawai* first;
};


void createList(List &L) {
    L.first = nullptr;
}


ElmPegawai* allocatePegawai(const Pegawai &data) {
    ElmPegawai* P = new ElmPegawai;
    P->info = data;
    P->next = nullptr;
    P->firstProyek = nullptr;
    return P;
}


ElmProyek* allocateProyek(const Proyek &data) {
    ElmProyek* P = new ElmProyek;
    P->info = data;
    P->next = nullptr;
    return P;
}


void insertPegawai(List &L, ElmPegawai* P) {
    if (L.first == nullptr) {
        L.first = P;
    } else {
        P->next = L.first;
        L.first = P;
    }
}


void insertProyek(ElmPegawai* pegawai, ElmProyek* proyek) {
    if (pegawai->firstProyek == nullptr) {
        pegawai->firstProyek = proyek;
    } else {
        ElmProyek* Q = pegawai->firstProyek;
        while (Q->next != nullptr) {
            Q = Q->next;
        }
        Q->next = proyek;
    }
}

```

```cpp
void removeProyek(ElmPegawai* pegawai, const string &nama_proyek) {
    if (pegawai->firstProyek == nullptr) return;

    ElmProyek* P = pegawai->firstProyek;
    ElmProyek* prev = nullptr;

    while (P != nullptr) {
        if (P->info.nama_proyek == nama_proyek) {
            if (prev == nullptr) {
                pegawai->firstProyek = P->next;
            } else {
                prev->next = P->next;
            }
            delete P;
            cout << "Proyek " << nama_proyek << " berhasil dihapus!\n";
            return;
        }
        prev = P;
        P = P->next;
    }
    cout << "Proyek " << nama_proyek << " tidak ditemukan!\n";
}


void printList(const List &L) {
    ElmPegawai* P = L.first;
    while (P != nullptr) {
        cout << "Pegawai: " << P->info.nama_pegawai << " (ID: " << P->info.id_pegawai << ")\n";
        ElmProyek* Q = P->firstProyek;
        while (Q != nullptr) {
            cout << "  Proyek: " << Q->info.nama_proyek << ", Durasi: " << Q->info.durasi << " bulan\n";
            Q = Q->next;
        }
        cout << endl;
        P = P->next;
    }
}

int main() {
    List L;
    createList(L);


    Pegawai p1 = {"Andi", "P001"};
    Pegawai p2 = {"Budi", "P002"};
    Pegawai p3 = {"Citra", "P003"};


    ElmPegawai* pegawai1 = allocatePegawai(p1);
    ElmPegawai* pegawai2 = allocatePegawai(p2);
    ElmPegawai* pegawai3 = allocatePegawai(p3);

    insertPegawai(L, pegawai1);
    insertPegawai(L, pegawai2);
    insertPegawai(L, pegawai3);


    Proyek proj1 = {"Aplikasi Mobile", 12};
    Proyek proj2 = {"Sistem Akuntansi", 8};
    Proyek proj3 = {"E-commerce", 10};
    Proyek proj4 = {"Analisis Data", 6};


    insertProyek(pegawai1, allocateProyek(proj1));
    insertProyek(pegawai2, allocateProyek(proj2));
    insertProyek(pegawai3, allocateProyek(proj3));
    insertProyek(pegawai1, allocateProyek(proj4));

    removeProyek(pegawai1, "Aplikasi Mobile");

    cout << "Data Pegawai dan Proyek:\n";
    printList(L);

    return 0;
}
```

OUTPUT



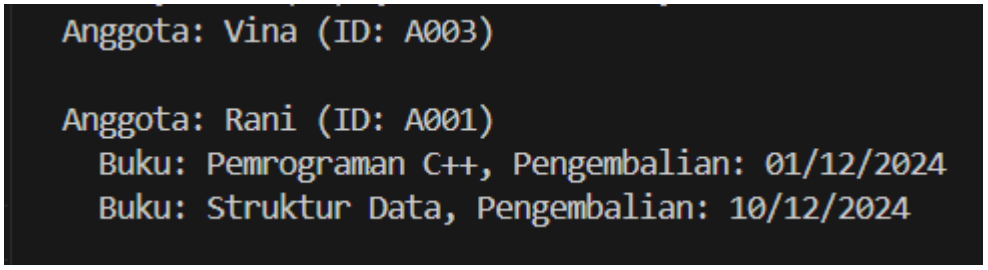
Struktur Multi-Linked List digunakan dalam program untuk mengelola data dan proyek individu yang bekerja di dalamnya. Setiap pegawai memiliki data seperti nama dan ID pegawai yang disimpan dalam node ElmEmployee. Setiap node pegawai juga memiliki pointer firstProject, yang menunjuk ke daftar proyek yang sedang dikerjakan oleh pegawai tersebut. Proyek ini memiliki informasi seperti nama proyek dan durasi proyek (dalam bulan), yang disimpan dalam node ElmProject. Struktur ini memungkinkan setiap pegawai memiliki daftar proyek mereka sendiri. Dengan menggunakan program, Anda dapat menambah pegawai baru (menambahkan Pegawai), menambah proyek ke daftar pegawai (menambahkan Proyek), menghapus pegawai dengan proyek mereka (hapus Pegawai), dan menampilkan data semua pegawai dan proyek yang mereka kelola (printList). Program ini membantu mengelola informasi proyek secara terstruktur dengan mengatur data hierarkis pekerja dan proyek.

2. LATIHAN 2

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

struct Buku {
    string judulBuku;
    string tanggalPengembalian;
};

struct Anggota {
    string namaAnggota;
    string idAnggota;
};

struct ElmBuku {
    Buku info;
    ElmBuku* next;
};

struct ElmAnggota {
    Anggota info;
    ElmBuku* firstBuku;
    ElmAnggota* next;
};

struct List {
    ElmAnggota* first;
};

void createList(List &L) {
    L.first = nullptr;
}

ElmAnggota* allocateAnggota(const Anggota &data) {
    ElmAnggota* P = new ElmAnggota;
    P->info = data;
    P->next = nullptr;
    P->firstBuku = nullptr;
    return P;
}

ElmBuku* allocateBuku(const Buku &data) {
    ElmBuku* P = new ElmBuku;
    P->info = data;
    P->next = nullptr;
    return P;
}

void insertAnggota(List &L, ElmAnggota* P) {
    if (L.first == nullptr) {
        L.first = P;
    } else {
        P->next = L.first;
        L.first = P;
    }
}

void insertBuku(ElmAnggota* anggota, ElmBuku* buku) {
    if (anggota->firstBuku == nullptr) {
        anggota->firstBuku = buku;
    } else {
        ElmBuku* Q = anggota->firstBuku;
        while (Q->next != nullptr) {
            Q = Q->next;
        }
        Q->next = buku;
    }
}
```

```cpp
void removeAnggota(List &L, const string &idAnggota) {
    ElmAnggota* P = L.first;
    ElmAnggota* prev = nullptr;

    while (P != nullptr) {
        if (P->info.idAnggota == idAnggota) {
            ElmBuku* buku = P->firstBuku;
            while (buku != nullptr) {
                ElmBuku* temp = buku;
                buku = buku->next;
                delete temp;
            }

            if (prev == nullptr) {
                L.first = P->next;
            } else {
                prev->next = P->next;
            }
            delete P;
            return;
        }
        prev = P;
        P = P->next;
    }
}

void printList(const List &L) {
    ElmAnggota* P = L.first;
    while (P != nullptr) {
        cout << "Anggota: " << P->info.namaAnggota << " (ID: " << P->info.idAnggota << ")\n";
        ElmBuku* Q = P->firstBuku;
        while (Q != nullptr) {
            cout << "  Buku: " << Q->info.judulBuku << ", Pengembalian: " << Q->info.tanggalPengembalian << "\n";
            Q = Q->next;
        }
        cout << endl;
        P = P->next;
    }
}

int main() {
    List L;
    createList(L);

    Anggota a1 = {"Rani", "A001"};
    Anggota a2 = {"Dito", "A002"};
    Anggota a3 = {"Vina", "A003"};

    ElmAnggota* anggota1 = allocateAnggota(a1);
    ElmAnggota* anggota2 = allocateAnggota(a2);
    ElmAnggota* anggota3 = allocateAnggota(a3);

    insertAnggota(L, anggota1);
    insertAnggota(L, anggota2);
    insertAnggota(L, anggota3);

    Buku b1 = {"Pemrograman C++", "01/12/2024"};
    Buku b2 = {"Algoritma Pemrograman", "15/12/2024"};
    Buku b3 = {"Struktur Data", "10/12/2024"};

    insertBuku(anggota1, allocateBuku(b1));
    insertBuku(anggota2, allocateBuku(b2));
    insertBuku(anggota1, allocateBuku(b3));

    removeAnggota(L, "A002");

    printList(L);

    return 0;
}
```

OUTPUT



Data anggota perpustakaan dan daftar buku yang mereka pinjam disimpan dalam program melalui struktur Multi-Linked List. Dalam node ElmAnggota, informasi seperti nama anggota dan ID anggota disimpan. Node ini juga memiliki pointer firstBook yang mengarah ke daftar buku yang dipinjam oleh anggota.

Buku mengandung data seperti judul buku dan tanggal pengembalian, yang disimpan dalam node ElmBuku. Dengan struktur ini, setiap anggota memiliki daftar buku sendiri, yang diimplementasikan sebagai daftar terhubung (linked list) yang berbeda. Program dapat menambah anggota (insertAnggota), menambah buku ke daftar anggota (insertBuku), menghapus anggota dan semua buku yang dipinjam (removeAnggota), dan menampilkan data semua anggota dan buku dalam daftar. Program ini memungkinkan manajemen data perpustakaan yang efisien dan hierarkis.