Tugas Pendahuluan Modul 13

STRUKTUR DATA - Ganjil 2024/2025

Multi Linked List

Ketentuan Tugas Pendahuluan

1. Tugas Pendahuluan dikerjakan secara Individu.
2. TP ini bersifat WAJIB, tidak mengerjakan = PENGURANGAN POIN JURNAL / TES ASESMEN.
3. Hanya MENGUMPULKAN tetapi TIDAK MENGERJAKAN = PENGURANGAN POIN JURNAL / TES ASESMEN.
4. Deadline pengumpulan TP Modul 2 adalah Senin, 30 September 2024 pukul 07.30 WIB.
5. TIDAK ADA TOLERANSI KETERLAMBATAN, TERLAMBAT ATAU TIDAK MENGUMPULKAN TP MAKA DIANGGAP TIDAK MENGERJAKAN.
6. DILARANG PLAGIAT (PLAGIAT = E).
7. Kerjakan TP dengan jelas agar dapat dimengerti.
8. Codingan diupload di Github dan upload Laporan di Lab menggunakan format PDF dengan ketentuan: TP_MOD_[XX]_NIM_NAMA.pdf

CP (WA):

- Andini (082243700965)
- Imelda (082135374187)

SELAMAT MENGERJAKAN^^

LAPORAN PRAKTIKUM
PERTEMUAN 13
MULTI LINKED LIST

Nama :

Tsaqif Hisyam Saputra (2311104024)

Dosen :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

A. Tujuan
    - Memahami Penggunaan Multi Linked List.
    - Mengimplementasikan Multi Linked List dalam beberapa studi kasus.
B. Tools
   Visual Studio Code dengan C++ Extensions Pack & Codeblocks

C. Latihan – Unguided

Manajemen Data Pegawai dan Proyek

1. Struktur Data

```
struct Project {
    string projectName;
    int duration;
    Project* next;
};

struct Employee {
    string name;
    string id;
    Project* projectHead;
    Employee* next;
};
```

- **Struct Project**: Digunakan untuk menyimpan data proyek individual, termasuk nama proyek dan durasinya. **next** digunakan untuk membuat linked list dari proyek.
- **Struct Employee**: Digunakan untuk menyimpan data pegawai individual, termasuk nama, ID, dan daftar proyek yang dikelola (melalui pointer **projectHead**).

2. Kelas EmployeeProjectManager

a. Deklarasi Awal

```
class EmployeeProjectManager {
private:
    Employee* head;

public:
    EmployeeProjectManager() : head(nullptr) {}
```

- **Employee* head**:
  Pointer ke pegawai pertama dalam daftar.
- **Konstruktor**:Menginisialisasi **head** dengan **nullptr**, menandakan bahwa daftar pegawai awalnya kosong.

b. Menambahkan Pegawai

```
void addEmployee(const string& name, const string& id) {
    Employee* newEmployee = new Employee{name, id, nullptr, head};
    head = newEmployee;
}
```

- Membuat node **Employee** baru dengan nama, ID, dan menghubungkannya ke daftar pegawai.
- Pointer **head** diarahkan ke pegawai baru, sehingga pegawai baru menjadi kepala daftar (linked list).

c. Menambahkan Proyek ke Pegawai

```
void addProjectToEmployee(const string& employeeId, const string& projectName, int duration) {
    Employee* emp = findEmployee(employeeId);
    if (emp) {
        Project* newProject = new Project{projectName, duration, emp->projectHead};
        emp->projectHead = newProject;
    } else {
        cout << "Employee with ID " << employeeId << " not found.\n";
    }
}
```

- **Pencarian Pegawai**:
  Menggunakan **findEmployee** untuk mencari pegawai berdasarkan ID.
- **Tambah Proyek**:
  Jika pegawai ditemukan, proyek baru dibuat dan ditambahkan di awal daftar proyek pegawai tersebut.

d. Menambahkan Proyek dari Pegawai

```
void removeProjectFromEmployee(const string& employeeId, const string& projectName) {
    Employee* emp = findEmployee(employeeId);
    if (emp) {
        Project* prev = nullptr;
        Project* current = emp->projectHead;
        while (current) {
            if (current->projectName == projectName) {
                if (prev) {
                    prev->next = current->next;
                } else {
                    emp->projectHead = current->next;
                }
                delete current;
                cout << "Project " << projectName << " removed from employee " << emp->name << ".\n";
                return;
            }
            prev = current;
            current = current->next;
        }
        cout << "Project " << projectName << " not found for employee " << emp->name << ".\n";
    } else {
        cout << "Employee with ID " << employeeId << " not found.\n";
    }
}
```

- **Cari Proyek**:
  Mencari proyek berdasarkan nama dalam daftar proyek pegawai.
- **Hapus Proyek**:
  1. Jika proyek ditemukan, node proyek dihapus dari linked list.
  2. Mengatur ulang pointer agar daftar tetap terhubung.

e. Menampilkan Semua Data

```
void displayAll() {
    Employee* emp = head;
    while (emp) {
        cout << "Employee Name: " << emp->name << ", ID: " << emp->id << "\n";
        Project* proj = emp->projectHead;
        if (proj) {
            cout << "  Projects:\n";
            while (proj) {
                cout << "    - " << proj->projectName << " (Duration: " << proj->duration << " months)\n";
                proj = proj->next;
            }
        } else {
            cout << "  No projects assigned.\n";
        }
        emp = emp->next;
    }
}
```

- Fungsi mencetak data semua pegawai dan daftar proyek mereka.
- Jika seorang pegawai tidak memiliki proyek, pesan “No projects assigned” ditampilkan.

f. Fungsi Pencarian Pegawai

```
private:
    Employee* findEmployee(const string& id) {
        Employee* current = head;
        while (current) {
            if (current->id == id) {
                return current;
            }
            current = current->next;
        }
        return nullptr;
    }
};
```

- Fungsi mencari pegawai berdasarkan ID dalam daftar pegawai.
- Jika ditemukan, menegembalikan pointer ke pegwai tersebut. Jika tidak, mengembalikan **nullptr**

3. Fungsi main

```
int main() {
    EmployeeProjectManager manager;

    // Adding employees
    manager.addEmployee("Andi", "P001");
    manager.addEmployee("Budi", "P002");
    manager.addEmployee("Citra", "P003");

    // Adding projects
    manager.addProjectToEmployee("P001", "Aplikasi Mobile", 12);
    manager.addProjectToEmployee("P002", "Sistem Akuntansi", 8);
    manager.addProjectToEmployee("P003", "E-commerce", 10);

    // Adding a new project to Andi
    manager.addProjectToEmployee("P001", "Analisis Data", 6);

    // Removing a project from Andi
    manager.removeProjectFromEmployee("P001", "Aplikasi Mobile");

    // Display all employees and their projects
    manager.displayAll();

    return 0;
}
```

- Dalam main ditambhkan pegawai dan proyek sesuai instruksinya. Menambahkan proyek baru untuk Andi, dan menghapus proyek “Aplikasi Mobile” dari Andi. Fungsi juga menampilkan semua pegawai dan proyek mereka.

OUTPUT:

```
Project Aplikasi Mobile removed from employee Andi.
Employee Name: Citra, ID: P003
  Projects:
    - E-commerce (Duration: 10 months)
Employee Name: Budi, ID: P002
  Projects:
    - Sistem Akuntansi (Duration: 8 months)
Employee Name: Andi, ID: P001
  Projects:
    - Analisis Data (Duration: 6 months)
```

Manajemen Buku Perpustakaan

1. **Struktur Data**
   a. **Node Buku**

```
struct Book {
    string title;
    string returnDate;
    Book* next;
};
```

- Menyimpan informasi tentang sebuah buku yang dipinja, termasuk judul buku, tanggal pengembalian, dan pointer ke buku berikutnya dalam daftar buku yang dipinjam.

   b. **Node Anggota**

```
struct Member {
    string name;
    string id;
    Book* bookHead;
    Member* next;
};
```

- Menyimpan informasi tentang seorang anggota perpustakaan.
- Memiliki pointer ke daftar buku yang sedang dipinjam oleh anggota tersebut

2. **Kelas LibrarySystem**
   a. **Deklarasi Awal**

```
class LibraryManager {
private:
    Member* head;

public:
    LibraryManager() : head(nullptr) {}
```

- Mengelola daftar anggota perpustakaan dengan pointer **head.**
- Konstruktor menginisialisasi sistem dengan daftar kosong (**head** = **nulllptr**).

b. Menambahkan Anggota

```
void addMember(const string& name, const string& id) {
    Member* newMember = new Member{name, id, nullptr, head};
    head = newMember;
}
```

- Menambahkan anggota baru ke sistem perpustakaan.
- Node anggota baru ditambahkan di awal daftar anggota.

c. Menambahkan Buku ke Anggota

```
void addBookToMember(const string& memberId, const string& title, const string& returnDate) {
    Member* member = findMember(memberId);
    if (member) {
        Book* newBook = new Book{title, returnDate, member->bookHead};
        member->bookHead = newBook;
    } else {
        cout << "Member with ID " << memberId << " not found.\n";
    }
}
```

- Menambahkan buku ke daftar buku yang dipinjam oleh seorang anggota.
- Mencari anggota berdasarkan ID menggunakn fungsi **findMember**.

d. Menghapus Anggota

```
void removeMember(const string& memberId) {
    Member* prev = nullptr;
    Member* current = head;
    while (current) {
        if (current->id == memberId) {
            if (prev) {
                prev->next = current->next;
            } else {
                head = current->next;
            }

            Book* book = current->bookHead;
            while (book) {
                Book* temp = book;
                book = book->next;
                delete temp;
            }

            delete current;
            cout << "Member with ID " << memberId << " and their books removed.\n";
            return;
        }
        prev = current;
        current = current->next;
    }
    cout << "Member with ID " << memberId << " not found.\n";
}
```

- Menghapus seorang anggota berdasarkan ID.
- Jika anggota ditemukan, semua buku yang dipinjam anggota tersebut juga dihapus menggunakan fungsi **clearBooks.**

e. Menghapus Semua Buku

```
void clearBooks(Book* bookHead) {
    while (bookHead) {
        Book* temp = bookHead;
        bookHead = bookHead->next;
        delete temp;
    }
}
```

- Membersihkan semua buku yang dipinjam oleh seorang anggota
- Membebaskan memori yang dialokasikan untuk setiap node buku

f. Menampilkan Semua Data

```
void displayAll() {
    Member* member = head;
    while (member) {
        cout << "Member Name: " << member->name << ", ID: " << member->id << "\n";
        Book* book = member->bookHead;
        if (book) {
            cout << "  Borrowed Books:\n";
            while (book) {
                cout << "    - " << book->title << " (Return Date: " << book->returnDate << ")\n";
                book = book->next;
            }
        } else {
            cout << "  No borrowed books.\n";
        }
        member = member->next;
    }
}
```

- Menampilkan semua data anggota dan buku yang dipinjam.
- Jika anggota tidak meminjam buku, ditampilkan pesan “No Borrowed Books”.

g. Fungsi Pencarian Anggota

```
private:
    Member* findMember(const string& id) {
        Member* current = head;
        while (current) {
            if (current->id == id) {
                return current;
            }
            current = current->next;
        }
        return nullptr;
    }
};
```

- Mencari anggota berdasarkan ID.
- Mengembalikan pointer ke anggota jika ditemukan, atau **nullptr** jika tidak ditemukan.

3. Fungsi main

```
int main() {
    LibraryManager library;

    // Adding members
    library.addMember("Rani", "A001");
    library.addMember("Dito", "A002");
    library.addMember("Vina", "A003");

    // Adding borrowed books
    library.addBookToMember("A001", "Pemrograman C++", "01/12/2024");
    library.addBookToMember("A002", "Algoritma Pemrograman", "15/12/2024");

    // Adding a new book to Rani
    library.addBookToMember("A001", "Struktur Data", "10/12/2024");

    // Removing member Dito and their books
    library.removeMember("A002");

    // Display all members and their borrowed books
    library.displayAll();

    return 0;
}
```

- Menambahkan tiga anggota perpustakaan (**Rani**, **Dito**, **Vina**).
- Menambahkan buku ke daftar buku yang dipinjam oleh **Rani** dan **Dito**.
- Menghapus anggota **Dito** dan semua buku yang dipinjamnya.
- Menampilkan semua data anggota beserta buku-buku yang dipinjam.

Semoga Selalu diberi kemudahan^^