LAPORAN PRAKTIKUM

PERTEMUAN 13

Nama :

Razhendriya Vania Ramadhan Suganjarsarwat (2311104048)

Dosen :

WAHYU ANDI SAPUTRA

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TUJUAN

Membuat program dengan **Multi Linked List** untuk mengelola data dengan struktur kompleks seperti data pegawai dan proyek, serta data anggota perpustakaan dan buku yang dipinjam. Tujuannya adalah memahami penerapan struktur data linked list dalam manajemen data bertingkat.

II. TOOL

Vscode

III. DASAR TEORI

Multi Linked List adalah struktur data yang menggabungkan konsep linked list untuk menyimpan data bertingkat. Misalnya:

- **Parent Node menyimpan data utama, seperti pegawai atau anggota.**
- **Child Node menyimpan data terkait, seperti proyek atau buku yang dipinjam.**

IV. GUIDED

```cpp
#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;
}
```

Multi Linked List

- **Tujuan: Menyimpan data induk (parent) dan anak (child).**
- **Fitur:**
    - **Tambah parent.**
    - **Tambah child ke parent tertentu.**
    - **Tampilkan parent dan daftar child.**

```cpp
#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;
}
```

Employee Management

- **Tujuan: Mengelola karyawan (manager) dan subordinate.**
- **Fitur:**
    - **Tambah karyawan (manager).**
    - **Tambah subordinate ke manager.**
    - **Tampilkan manager dan subordinate mereka.**

Employee Management + Delete

- **Tujuan: Versi lanjutan program 2 dengan fitur hapus.**
- **Fitur:**
  - **Tambah karyawan dan subordinate.**
  - **Hapus subordinate dari manager.**
  - **Hapus karyawan beserta subordinate-nya.**
  - **Tampilkan hasil perubahan.**

V. UNGUIDED

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

struct ProjectNode {
    string name;
    int duration;
    ProjectNode* next;

    ProjectNode(string name, int duration) : name(name), duration(duration), next(nullptr) {}
};

struct EmployeeNode {
    string name, id;
    ProjectNode* projects;
    EmployeeNode* next;

    EmployeeNode(string name, string id) : name(name), id(id), projects(nullptr), next(nullptr) {}
};

class EmployeeProjectList {
private:
    EmployeeNode* head;

public:
    EmployeeProjectList() : head(nullptr) {}

    void addEmployee(string name, string id) {
        EmployeeNode* newEmp = new EmployeeNode(name, id);
        newEmp->next = head;
        head = newEmp;
    }

    void addProject(string empId, string projName, int duration) {
        EmployeeNode* emp = head;
        while (emp && emp->id != empId) emp = emp->next;
        if (emp) {
            ProjectNode* newProj = new ProjectNode(projName, duration);
            newProj->next = emp->projects;
            emp->projects = newProj;
        }
    }

    void deleteProject(string empId, string projName) {
        EmployeeNode* emp = head;
        while (emp && emp->id != empId) emp = emp->next;
        if (emp) {
            ProjectNode** curr = &emp->projects;
            while (*curr && (*curr)->name != projName) curr = &(*curr)->next;
            if (*curr) {
                ProjectNode* toDelete = *curr;
                *curr = (*curr)->next;
                delete toDelete;
            }
        }
    }

    void display() {
        EmployeeNode* emp = head;
        while (emp) {
            cout << "Pegawai: " << emp->name << " (ID: " << emp->id << ")\n";
            ProjectNode* proj = emp->projects;
            while (proj) {
                cout << "  Proyek: " << proj->name << " - " << proj->duration << " bulan\n";
                proj = proj->next;
            }
            emp = emp->next;
        }
    }
};

int main() {
    EmployeeProjectList list;
    // Masukkan data pegawai
    list.addEmployee("Andi", "P001");
    list.addEmployee("Budi", "P002");
    list.addEmployee("Citra", "P003");

    // Tambahkan proyek
    list.addProject("P001", "Aplikasi Mobile", 12);
    list.addProject("P002", "Sistem Akuntansi", 8);
    list.addProject("P003", "E-commerce", 10);
    list.addProject("P001", "Analisis Data", 6);

    // Hapus proyek Aplikasi Mobile
    list.deleteProject("P001", "Aplikasi Mobile");

    // Tampilkan data pegawai dan proyek
    list.display();
    return 0;
}
```

Manajemen Data Pegawai dan Proyek

Program ini menggunakan **Multi Linked List** untuk:

1. Menyimpan data pegawai (nama dan ID).
2. Menyimpan proyek yang dikelola setiap pegawai (nama proyek dan durasi).

Fitur Utama:

- Tambah pegawai dan proyek.
- Hapus proyek tertentu dari pegawai.
- Tampilkan semua data pegawai beserta proyek mereka.

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

struct BookNode {
    string title, returnDate;
    BookNode* next;

    BookNode(string t, string d) : title(t), returnDate(d), next(nullptr) {}
};

struct MemberNode {
    string name, id;
    BookNode* books;
    MemberNode* next;

    MemberNode(string n, string i) : name(n), id(i), books(nullptr), next(nullptr) {}
};

class LibrarySystem {
private:
    MemberNode* head;

public:
    LibrarySystem() : head(nullptr) {}

    void addMember(string name, string id) {
        MemberNode* newMem = new MemberNode(name, id);
        newMem->next = head;
        head = newMem;
    }

    void addBook(string memId, string title, string returnDate) {
        MemberNode* mem = head;
        while (mem && mem->id != memId) mem = mem->next;
        if (mem) {
            BookNode* newBook = new BookNode(title, returnDate);
            newBook->next = mem->books;
            mem->books = newBook;
        }
    }

    void deleteMember(string memId) {
        MemberNode** curr = &head;
        while (*curr && (*curr)->id != memId) curr = &(*curr)->next;
        if (*curr) {
            BookNode* book = (*curr)->books;
            while (book) {
                BookNode* toDelete = book;
                book = book->next;
                delete toDelete;
            }
            MemberNode* toDelete = *curr;
            *curr = (*curr)->next;
            delete toDelete;
        }
    }

    void display() {
        MemberNode* mem = head;
        while (mem) {
            cout << "Anggota: " << mem->name << " (ID: " << mem->id << ")\n";
            BookNode* book = mem->books;
            while (book) {
                cout << "  Buku: " << book->title << " - Pengembalian: " << book->returnDate << "\n";
                book = book->next;
            }
            mem = mem->next;
        }
    }
};

int main() {
    LibrarySystem lib;

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

    // Tambahkan buku
    lib.addBook("A001", "Pemrograman C++", "01/12/2024");
    lib.addBook("A002", "Algoritma Pemrograman", "15/12/2024");
    lib.addBook("A001", "Struktur Data", "10/12/2024");

    // Hapus anggota Dito
    lib.deleteMember("A002");

    // Tampilkan data anggota dan buku
    lib.display();
    return 0;
}
```

Sistem Manajemen Buku Perpustakaan

Program ini juga menggunakan **Multi Linked List** untuk:

1. Menyimpan data anggota perpustakaan (nama dan ID).
2. Menyimpan buku yang dipinjam setiap anggota (judul buku dan tanggal pengembalian).

Fitur Utama:

- Tambah anggota dan buku yang dipinjam.
- Hapus anggota beserta buku yang dipinjam.
- Tampilkan semua data anggota beserta buku yang dipinjam.

VI. KESIMPULAN

Multi Linked List sangat efektif untuk mengelola data bertingkat seperti data pegawai-proyek dan anggota-buku perpustakaan. Struktur ini mendukung penambahan, penghapusan, dan pencarian data dengan fleksibilitas tinggi.