LAPORAN PRAKTIKUM

Modul ~~10 11 12~~ 13

MULTI LINKED LIST

Disusun Oleh:

Adhiansyah Muhammad Pradana Farawowan - 2211104038

S1SE-07-02

Asisten Praktikum:

Aldi Putra

Andini Nur Hidayah

Dosen:

Wahyu Andi Saputra, S.Pd., M.Eng.

PROGRAM STUDI S1 REKAYASAN PERANGKAT LUNAK

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM PURWOKERTO

2024

A. Tujuan

Laporan praktikum ini memiliki tujuan di bawah berikut.

1. Memperkenalkan konsep *multi linked list*, sebagai salah satu struktur data
2. Memahami dan mengelola cara kerja *multi linked list*
3. Mengimplementasikan *multi linked list* dalam bahasa C++

B. Landasan Teori

Multi linked list adalah struktur data seperti daftar berantai (*linked list*) tetapi setiap simpul juga menunjuk pada sebuah daftar berantai lain. Operasi-operasinya sama seperti daftar berantai; Menambahkan, mencari, dan mengurangi simpul.

1. Menambahkan simpul
   Ide dasarnya adalah sebagai berikut.
   1. Buat simpul `x`
   2. Isi propertinya `x`, seperti data dan alamat untuk menunjuk ke simpul selanjutnya (biasanya diisi *null*, bukan tidak terisi)
   3. Periksa apakah simpul pertama dari sebuah daftar (*head*) ada isinya atau tidak
      1. Jika terisi, telusuri tiap simpul sampai pada simpul terakhir (*tail*), kemudian jadikan `x` sebagai simpul terakhir
      2. Jika tidak, referensikan `x` sebagai simpul pertama

2. Mencari data
   Ide dasarnya adalah kita mencari data yang unik satu sama lain dengan mencocokkannya antara yang dicari dengan tiap simpul.
   1. Telusur daftar berantai, dimulai dari simpul pertama (*head*)
   2. Periksa apakah tiap data simpul (direpresentasikan dengan properti data) sama dengan data yang dicari
      1. Jika ada, kembalikan nilainya
      2. Jika tidak, lanjut sampai berakhir
3. Menghapus data
   Ide dasarnya mirip seperti mencari data, tapi kita perlu tahu teknik menghapus untuk struktur data linear satu arah.
   1. Telusur daftar berantai, dimulai dari simpul pertama (*head*)
   2. Periksa apakah *simpul selanjutnya* dari simpul yang ditelusuri (direpresentasikan dengan properti data, selanjutnya kita sebut `prev`) sama dengan data yang ingin dihapus
      1. Jika ada
         1. Buat variabel sementara `temp`
         2. Isi `temp` dengan referensi ke data yang ingin dihapus
         3. Arahkan simpul selanjutnya dari `prev` ke simpul selanjutnya dari data yang ingin dihapus
         4. Hapus `temp`
      2. Jika tidak, lanjut sampai berakhir

C. Bimbingan (*guided*)

Bimbingan hari ini adalah mengimplementasikan sebuah *multi linked list*, dengan modifikasi sendiri.

guided_1.cpp

```
#include <iostream>
#include <string>

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 add_parent(int data)
    {
        Node *new_node = new Node(data);
        new_node->next = head;
        head = new_node;
    }

    void add_child(int parentData, int childData)
    {
        Node *parent = head;
        while (parent != nullptr && parent->data != parentData)
        {
            parent = parent->next;
        }
        if (parent != nullptr)
        {
            Node *new_child = new Node(childData);
            new_child->next = parent->child;
            parent->child = new_child;
        }
        else
        {
            std::cout << "Parent " << parentData << " not found!" << '\n';
        }
    }

    void print_all()
    {
        Node *current = head;
        while (current != nullptr)
        {
            std::cout << "Parent: " << current->data << " -> ";
            Node *child = current->child;
            while (child != nullptr)
            {
                std::cout << child->data << " ";
                child = child->next;
            }
```

```
            std::cout << '\n';
            current = current->next;
        }
    }

    // Destructor for mutli linked list
    ~MultiLinkedList()
    {
        while (head != nullptr)
        {
            Node *temp = head;
            head = head->next;

            while (temp->child != nullptr)
            {
                Node *to_be_removed = temp->child;
                temp->child = temp->child->next;
                delete to_be_removed;
            }
            delete temp;
        }
    }
};

int main()
{
    MultiLinkedList the_list;

    the_list.add_parent(3);
    the_list.add_parent(5);
    the_list.add_parent(7);

    the_list.add_child(3, 415);
    the_list.add_child(3, 415);
    the_list.add_child(5, 899);
    the_list.add_child(5, 899);
    the_list.add_child(7, 721);
    the_list.add_child(1, 721);
    the_list.print_all();

    return 0;
}
```

Output dari guided_1.cpp

```
                                                    >a.exe
Parent 1 not found!
Parent: 7 -> 721
Parent: 5 -> 899 899
Parent: 3 -> 415 415
```

guided_2.cpp

```
#include <iostream>
#include <string>

struct SimpleEmployee
{
    std::string name;
    SimpleEmployee *next;
    SimpleEmployee *subordinate;

    SimpleEmployee(std::string employee_name)
    {
        name = employee_name;
        next = nullptr;
        subordinate = nullptr;
    }
};
```

```
class SimpleEmployeeList
{
private:
    SimpleEmployee *head;

public:
    SimpleEmployeeList()
    {
        head = nullptr;
    }

    void add_employee(std::string name)
    {
        SimpleEmployee *new_employee = new SimpleEmployee(name);
        new_employee->next = head;
        head = new_employee;
    }

    void add_subordinate(std::string manager_name, std::string subordinate_name)
    {
        SimpleEmployee *manager = head;
        while (manager != nullptr && manager->name != manager_name)
        {
            manager = manager->next;
        }
        if (manager != nullptr)
        {
            SimpleEmployee *new_subordinate = new SimpleEmployee(subordinate_name);
            new_subordinate->next = manager->subordinate;
            manager->subordinate = new_subordinate;
        }
        else
        {
            std::cout << "Manager not found!" << '\n';
        }
    }

    void print_all()
    {
        SimpleEmployee *current = head;
        while (current != nullptr)
        {
            std::cout << "Manager: " << current->name << " -> ";
            SimpleEmployee *sub = current->subordinate;
            while (sub != nullptr)
            {
                std::cout << sub->name << " ";
                sub = sub->next;
            }
            std::cout << '\n';
            current = current->next;
        }
    }

    ~SimpleEmployeeList()
    {
        while (head != nullptr)
        {
            SimpleEmployee *temp = head;
            head = head->next;

            while (temp->subordinate != nullptr)
            {
                SimpleEmployee *subTemp = temp->subordinate;
                temp->subordinate = temp->subordinate->next;

                delete subTemp;
            }

            delete temp;
        }
    }
};
```

```
int main()
{
    SimpleEmployeeList the_list;

    the_list.add_employee("Shadow");
    the_list.add_employee("Oriana");
    the_list.add_employee("Midgar");

    the_list.add_subordinate("Shadow", "Nu");
    the_list.add_subordinate("Shadow", "Victoria");
    the_list.add_subordinate("Oriana", "Rose");

    the_list.add_subordinate("Midgar", "Alexia");
    the_list.add_subordinate("Midgar", "Iris");

    the_list.print_all();

    return 0;
}
```

Output dari guided_2.cpp

```
C:\Users\Ampf\LabProgram\Kode\StrukDat_Sem5\10_MLL>a.exe
Manager: Midgar -> Iris Alexia
Manager: Oriana -> Rose
Manager: Shadow -> Victoria Nu
```

guided_3.cpp

```
#include <iostream>
#include <string>

struct SimpleEmployee
{
    std::string name;
    SimpleEmployee *next;
    SimpleEmployee *subordinate;

    SimpleEmployee(std::string employee_name)
    {
        name = employee_name;
        next = nullptr;
        subordinate = nullptr;
    }
};

class SimpleEmployeeList
{
private:
    SimpleEmployee *head;

public:
    SimpleEmployeeList()
    {
        head = nullptr;
    }

    void add_employee(std::string name)
    {
        SimpleEmployee *new_employee = new SimpleEmployee(name);
        new_employee->next = head;
        head = new_employee;
    }

    void add_subordinate(std::string manager_name, std::string subordinate_name)
    {
        SimpleEmployee *manager = head;
        while (manager != nullptr && manager->name != manager_name)
        {
            manager = manager->next;
```

```
        }
        if (manager != nullptr)
        {
            SimpleEmployee *new_subordinate = new SimpleEmployee(subordinate_name);
            new_subordinate->next = manager->subordinate;
            manager->subordinate = new_subordinate;
        }
        else
        {
            std::cout << "Manager not found!" << '\n';
        }
    }

    void print_all()
    {
        SimpleEmployee *current = head;
        while (current != nullptr)
        {
            std::cout << "Manager: " << current->name << " -> ";
            SimpleEmployee *sub = current->subordinate;
            while (sub != nullptr)
            {
                std::cout << sub->name << " ";
                sub = sub->next;
            }
            std::cout << '\n';
            current = current->next;
        }
    }

    ~SimpleEmployeeList()
    {
        while (head != nullptr)
        {
            SimpleEmployee *temp = head;
            head = head->next;

            while (temp->subordinate != nullptr)
            {
                SimpleEmployee *subTemp = temp->subordinate;
                temp->subordinate = temp->subordinate->next;

                delete subTemp;
            }

            delete temp;
        }
    }
};

int main()
{
    SimpleEmployeeList the_list;

    the_list.add_employee("Shadow");
    the_list.add_employee("Oriana");
    the_list.add_employee("Midgar");

    the_list.add_subordinate("Shadow", "Nu");
    the_list.add_subordinate("Shadow", "Victoria");
    the_list.add_subordinate("Oriana", "Rose");

    the_list.add_subordinate("Midgar", "Alexia");
    the_list.add_subordinate("Midgar", "Iris");

    the_list.print_all();

    return 0;
}
```

Output dari guided_3.cpp

```
[redacted]>a.exe
Initial employee list:
Manager: Diethard ->
Manager: Schneizel -> Cornelia,
Manager: Lelouch -> Ohgi, Kallen,
Subordinate Kallen deleted from Lelouch
Employee Diethard deleted.
Updated employee list:
Manager: Schneizel -> Cornelia,
Manager: Lelouch -> Ohgi,
```

D. Tugas mandiri (*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.

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.

Kode

unguided_1.cpp

```
#include <iostream>
#include <string>

struct Proyek
{
    std::string nama;
    int durasi; // Durasi dalam bulan
    Proyek *selanjutnya;
};

struct Pegawai
{
    std::string nama;
    std::string id;
    Proyek *proyek;
    Pegawai *selanjutnya;
};

Pegawai *awal_simpul = nullptr;

void tambah_pegawai(std::string nama, std::string id)
{
    Pegawai *baru = new Pegawai;
    baru->nama = nama;
    baru->id = id;
    baru->proyek = nullptr;
    baru->selanjutnya = nullptr;

    if (awal_simpul == nullptr)
    {
        awal_simpul = baru;
    }
    else
    {
        Pegawai *telusur_pegawai = awal_simpul;
        while (telusur_pegawai->selanjutnya != nullptr)
        {
            telusur_pegawai = telusur_pegawai->selanjutnya;
        }
        telusur_pegawai->selanjutnya = baru;
    }
}

void tambah_proyek_baru(std::string id_pegawai, std::string nama_proyek_baru, int
durasi_proyek)
{
    Pegawai *telusur_pegawai = awal_simpul;
    while (telusur_pegawai != nullptr)
    {
        if (telusur_pegawai->id == id_pegawai)
        {
            Proyek *proyek_baru = new Proyek;
            proyek_baru->nama = nama_proyek_baru;
```

```
            proyek_baru->durasi = durasi_proyek;
            proyek_baru->selanjutnya = nullptr;

            if (telusur_pegawai->proyek == nullptr)
            {
                telusur_pegawai->proyek = proyek_baru;
            }
            else
            {
                Proyek *telusur_proyek = telusur_pegawai->proyek;
                while (telusur_proyek->selanjutnya != nullptr)
                {
                    telusur_proyek = telusur_proyek->selanjutnya;
                }
                telusur_proyek->selanjutnya = proyek_baru;
            }
            return;
        }
        telusur_pegawai = telusur_pegawai->selanjutnya;
    }
}

void hapus_proyek(std::string id_pegawai, std::string nama_proyek)
{
    Pegawai *telusur_pegawai = awal_simpul;
    while (telusur_pegawai != nullptr)
    {
        if (telusur_pegawai->id == id_pegawai)
        {
            Proyek *telusur_proyek = telusur_pegawai->proyek;
            while (telusur_proyek->selanjutnya != nullptr)
            {
                if (telusur_proyek->nama == nama_proyek)
                {
                    Proyek *proyek_akan_dihapus = telusur_proyek;
                    telusur_pegawai->proyek = telusur_proyek->selanjutnya;

                    delete proyek_akan_dihapus;
                }

                telusur_proyek = telusur_proyek->selanjutnya;
            }
        }

        telusur_pegawai = telusur_pegawai->selanjutnya;
    }
}

void tampilkan_data()
{
    Pegawai *p = awal_simpul;
    while (p != nullptr)
    {
        std::cout << p->nama << ": ";
        Proyek *proyek_pegawai = p->proyek;
        while (proyek_pegawai != nullptr)
        {
            std::cout << proyek_pegawai->nama << " -> ";
            proyek_pegawai = proyek_pegawai->selanjutnya;
        }

        std::cout << "NULLPTR" << '\n';
        p = p->selanjutnya;
    }
}

int main()
{
    tambah_pegawai("Andi", "P001");
    tambah_pegawai("Budi", "P002");
    tambah_pegawai("Citra", "P003");

    tambah_proyek_baru("P001", "Aplikasi Mobile", 12);
    tambah_proyek_baru("P002", "Sistem Akuntansi", 8);
```

```
    tambah_proyek_baru("P003", "E-commerce", 10);

    tambah_proyek_baru("P001", "Analisis Data", 6);

    hapus_proyek("P001", "Aplikasi Mobile");

    tampilkan_data();

    return 0;
}
```

Output dari guided_1.cpp

```
                                                    >a.exe
Andi: Analisis Data -> NULLPTR
Budi: Sistem Akuntansi -> NULLPTR
Citra: E-commerce -> NULLPTR
```

unguided_2.cpp

```
#include <iostream>
#include <string>

struct Buku
{
    std::string judul;
    std::string tanggal_kembali;
    Buku *selanjutnya;
};

struct AnggotaPerpus
{
    std::string nama;
    std::string id;
    Buku *buku_dipinjam;
    AnggotaPerpus *selanjutnya;
};

AnggotaPerpus *entri_depan = nullptr;

void tambah_anggota_perpus(std::string nama, std::string id)
{
    AnggotaPerpus *anggota_baru = new AnggotaPerpus;
    anggota_baru->nama = nama;
    anggota_baru->id = id;
    anggota_baru->buku_dipinjam = nullptr;
    anggota_baru->selanjutnya = nullptr;

    if (entri_depan == nullptr)
    {
        entri_depan = anggota_baru;
    }
    else
    {
        AnggotaPerpus *telusur_anggota = entri_depan;
        while (telusur_anggota->selanjutnya != nullptr)
        {
            telusur_anggota = telusur_anggota->selanjutnya;
        }
        telusur_anggota->selanjutnya = anggota_baru;
    }
}

void tambah_buku(std::string id_anggota, std::string judul, std::string tanggal_kembali)
{
    AnggotaPerpus *telusur_anggota = entri_depan;
    while (telusur_anggota != nullptr)
    {
        if (telusur_anggota->id == id_anggota)
        {
            Buku *buku_baru = new Buku;
```

```
            buku_baru->judul = judul;
            buku_baru->tanggal_kembali = tanggal_kembali;
            buku_baru->selanjutnya = nullptr;

            if (telusur_anggota->buku_dipinjam == nullptr)
            {
                telusur_anggota->buku_dipinjam = buku_baru;
            }
            else
            {
                Buku *telusur_buku = telusur_anggota->buku_dipinjam;
                while (telusur_buku->selanjutnya != nullptr)
                {
                    telusur_buku = telusur_buku->selanjutnya;
                }
                telusur_buku->selanjutnya = buku_baru;
            }
            return;
        }
        telusur_anggota = telusur_anggota->selanjutnya;
    }
}

void hapus_seluruh_data(std::string id_anggota)
{
    AnggotaPerpus *telusur_anggota = entri_depan;
    AnggotaPerpus *anggota_sebelumnya = nullptr;
    while (telusur_anggota != nullptr)
    {
        if (telusur_anggota->id == id_anggota)
        {
            Buku *telusur_buku = telusur_anggota->buku_dipinjam;
            while (telusur_buku != nullptr)
            {
                Buku *buku_akan_dihapus = telusur_buku;
                telusur_buku = telusur_buku->selanjutnya;
                delete buku_akan_dihapus;
            }

            if (anggota_sebelumnya != nullptr)
            {
                anggota_sebelumnya->selanjutnya = telusur_anggota->selanjutnya;
            }
            else
            {
                entri_depan = telusur_anggota->selanjutnya;
            }
            delete telusur_anggota;
            break;
        }
        anggota_sebelumnya = telusur_anggota;
        telusur_anggota = telusur_anggota->selanjutnya;
    }
}

// Kode ini sebaiknya tidak disentuh
void tampilkan_semua_anggota()
{
    AnggotaPerpus *daftar_anggota = entri_depan;
    while (daftar_anggota != nullptr)
    {
        std::cout << daftar_anggota->nama << " (" << daftar_anggota->id << ") = ";

        Buku *daftar_buku_dipinjam = daftar_anggota->buku_dipinjam;
        if (daftar_buku_dipinjam == nullptr)
        {
            // pass
        }
        else
        {
            while (daftar_buku_dipinjam != nullptr)
            {
                std::cout << daftar_buku_dipinjam->judul << " <" << daftar_buku_dipinjam-
>tanggal_kembali << "> -> ";
```

```
                daftar_buku_dipinjam = daftar_buku_dipinjam->selanjutnya;
            }
        }

        std::cout << "NULLPTR" << '\n';
        daftar_anggota = daftar_anggota->selanjutnya;
    }
}

int main()
{

    tambah_anggota_perpus("Rani", "A001");
    tambah_anggota_perpus("Dito", "A002");
    tambah_anggota_perpus("Vina", "A003");

    tambah_buku("A001", "Pemrograman C++", "01/12/2024");
    tambah_buku("A002", "Algoritma Pemrograman", "15/12/2024");
    tambah_buku("A001", "Struktur Data", "10/12/2024");

    hapus_seluruh_data("A002");

    tampilkan_semua_anggota();

    return 0;
}
```

Output dari guided_2.cpp

```
>a.exe
Rani (A001) = Pemrograman C++ <01/12/2024> -> Struktur Data <10/12/2024> -> NULLPTR
Vina (A003) = NULLPTR
```